TELECOM PARIS
école nationale supérieure des télécommunications

Thèse

présentée pour obtenir le grade de docteur

de l'Ecole Nationale Supérieure des Télécommunications

Spécialité : Informatique et Réseaux

Walid BAGGA

Cryptographie à Base de Politiques: Théorie et Applications

soutenue le 08 Décembre 2006 devant le jury composé de

Rapporteurs	**Giovanni Di Crescenzo**	**(Senior Scientist, Telcordia Technologies)**
	Marc Girault	**(Expert Émérite, France Telecom)**
Examinateurs	**Yves Deswarte**	**(Directeur de Recherche, LAAS)**
	Gene Tsudik	**(Professeur, University of California Irvine)**
	Pascal Urien	**(Professeur, Telecom Paris)**
Directeur de Thèse	**Refik Molva**	**(Professeur, Institut Eurecom)**

Ecole Nationale Supérieure des Télécommunications

TELECOM PARIS
école nationale supérieure des télécommunications

Ph.D. Thesis

Ecole Nationale Supérieure des Télécommunications

Computer Science and Networks

Walid BAGGA

Policy-Based Cryptography: Theory and Applications

Defense date: December, 8th 2006

Committee in charge:

Reporters	**Giovanni Di Crescenzo**	**(Senior Scientist, Telcordia Technologies)**
	Marc Girault	**(Expert Émérite, France Telecom)**
Examiners	**Yves Deswarte**	**(Directeur de Recherche, LAAS)**
	Gene Tsudik	**(Professor, University of California Irvine)**
	Pascal Urien	**(Professor, Telecom Paris)**
Advisor	**Refik Molva**	**(Professor, Institut Eurecom)**

Ecole Nationale Supérieure des Télécommunications

À toutes celles et tous ceux qui ont contribué de près ou de loin à l'accomplissement de ce travail, je dis, tout simplement et du fond du coeur, merci!

... à mes parents.

Résumé en Français

Entre ce que je pense, ce que je veux dire, ce que je crois dire, ce que je dis,
ce que vous avez envie d'entendre, ce que vous entendez, ce que vous comprenez...
Il y a dix possibilités qu'on ait des difficultés à communiquer. Mais essayons quand même...
- Bernard Werber -

La cryptographie est une discipline du domaine de la sécurité de l'information et des communications qui permet à travers des primitives mathématiques de fournir un ensemble de services de sécurité telles que la confidentialité, l'intégrité, l'authenticité et la non répudiation. La contribution principale de cette thèse de doctorat est la formalisation du concept de 'cryptographie à base de politiques'. Ce concept permet de combiner la cryptographie avec la notion de politiques. Une politique est typiquement définie à travers une combinaison de règles ou de conditions devant être respectées pour atteindre un objectif de sécurité pre-défini. Ainsi, une primitive cryptographique à base de politiques peut être simplement vue comme une primitive cryptographique qui fournit un ou plusieurs services de sécurité tout en donnant la preuve de la conformité avec une politique spécifique.

Ce résumé en français est organisé comme suit: nous commençons dans l'introduction par fournir les motivations justifiant le concept de cryptographie à base de politiques et décrire l'organisation de ce rapport tout en soulignant nos différentes contributions dans cette thèse. Nous décrivons ensuite aussi bien les aspects théoriques que pratiques de trois primitives cryptographiques à base de politiques développées au cours de cette thèse. Une conclusion générale pour notre travail de recherche est donnée à la fin de ce résumé.

Introduction

L'approche standard en cryptographie

En 1976, Diffie et Hellman ont présenté le concept de cryptographie asymétrique; un concept considéré comme le développement le plus important dans l'histoire de la cryptography moderne [63]. L'objectif premier de leur concept était de surmonter un problème inhérent à la cryptographie symétrique; celui de la distribution de clés. En effet, aussi connu sous le nom de cryptographie à clé publique, leur concept permet de concevoir des schémas de chiffrement qui n'exigent pas l'échange de clés secrètes. En outre, il permet de concevoir une variété de primitives cryptographiques plus sophistiquées les unes que les autres fournissant des services de sécurité autres que la confidentialité. La plus significative de ces primitives est sans doute la signature numérique. Une description plus détaillée des primitives de chiffrement et de signature à clés publiques est donnée ci-dessous:

- **chiffrement à clé publique**: cette primitive est principalement utilisée pour assurer la confidentialité. Un schéma de chiffrement à clé publique permet de chiffrer un message en utilisant une clé publique tel que seulement une entité ayant accès à la clé privée correspondante puisse déchiffrer le message. En 1978, le premier schéma pratique de chiffrement à clé publique a été présenté par Rivest, Shamir et Adleman [130]. Plus connu sous le nom de RSA, sa sécurité est basée sur la difficulté de factoriser de grands nombres entiers. En 1985, un autre schéma pratique de chiffrement à clé publique basé sur la difficulté du problème du logarithme discret a été proposé par El Gamal [82].

- **signature numérique**: cette primitive cryptographique permet de founir les services d'intégrité, d'authenticité et de non répudiation. Un schéma de signature numérique permet de générer une signature sur un message telle que la signature soit valide par rapport à une clé publique si et seulement si elle ait été produite en utilisant la clé privée correspondante. En 1991, la première norme internationale pour les signatures numériques a été adoptée (ISO/IEC9796). Elle est basée sur le schéma de chiffrement à clé publique RSA. Deux ans plus tard, le gouvernement des États-Unis a adopté la norme de signature numérique, désignée sous le nom du DSA, comme mécanisme standard de signature basé sur le schéma de chiffrement à clé publique d'El Gamal (PAP 186).

Dans son utilisation standard, une clé publique apparaît comme une chaîne binaire aléatoire telle qu'il n'y ait rien à son sujet indiquant à qui elle appartient. Ainsi, une des questions auxquelles sont confrontés les schémas standards de chiffrement et de signature à clé publique est le déploiement et la gestion d'infrastructures assurant l'authenticité des clés publiques utilisées. Sommairement, on doit établir une méthode qui fournit une assurance à l'entité qui chiffre un message au sujet du rapport entre la clé publique utilisée et l'identité de l'entité légitime ayant accès à la clé privée correspondante.

Actuellement, la méthode la plus populaire pour l'authentification des clés publiques consiste à utiliser les certificats de clés publiques; originalement formalisés par Kohnfelder dans [102]. Un certificat de clé publique est simplement la signature d'une entité de confiance, appelée l'autorité de certification, sur une affirmation qui lie une clé publique à l'identité de l'entité légitime ayant accès à la clé privée correspondante. En signant l'affirmation, l'autorité de

certification atteste que la clé publique contenue dans l'affirmation appartient à l'entité dont l'identité est celle indiquée par l'affirmation.

Dans cette approche largement adoptée, on suppose qu'une entité qui envoie un message chiffré à une autre entité connaisse au préalable l'identité de l'entité pour laquelle le message est destiné. Avant de chiffrer un message en utilisant une certaine clé publique, une entité doit ainsi obtenir un certificat valide de clé publique fournissant l'assurance du rapport entre la clé publique et l'identité du destinataire légitime du message. De manière analogue, on suppose qu'une entité recevant une signature sur un message connaisse l'identité de l'entité qui a produit la signature et a l'assurance, à travers la certification de clé publique, de la relation entre l'identité du signataire et la clé publique selon lesquelles la signature est considérée comme valide. Une infrastructure de sécurité dont les services sont mis en oeuvre pour gérer l'utilisation des certificats de clés publiques s'appelle l'infrastructure de clé publique, souvent dénotée par PKI. Une alternative élégante à la certification de cés publiques est offerte par la cryptographie à base d'identités.

Cryptographie à base d'identités

Le concept de cryptographie à base d'identités a été formulé par Shamir en 1984 [137]. Son objectif premier était d'éliminer le besoin de certificats pour l'authentification de clés publiques en utilisant l'identité d'une entité comme sa clé publique. La clé privée correspondante est produite par une autorité de confiance, appelée le générateur de clés privées. Elle est transmise à l'entité en question par un canal sécurisé. Un schéma de chiffrement à base d'identités permet ainsi de chiffrer un message par rapport à une identité de telle manière que seulement une entité dont l'identité est celle par rapport à laquelle le message a été chiffré puisse déchiffrer le message. De manière analogue, un schéma de signature à base d'identités permet de produire une signature sur un message de telle manière que la signature soit valide seulement par rapport à l'identité de l'entité qui l'a produite.

Des solutions efficaces pour des schémas de signature à base d'identités ont été rapidement concus [71, 69], tandis que la conception de schémas de chiffrement à base d'identités s'est avérée plus compliquée. En effet, aucun schéma de chiffrement à base d'identités proposé entre 1984 et 2000 n'était entièrement satisfaisant en termes de sécurité et d'efficacité. En 2001, Boneh et Franklin ont réussi, en utilisant les couplages bilinéaires sur des courbes elliptiques, à développer un schéma de chiffrement à base d'identités qui soit à la fois pratique et dont la sécurité soit prouvée [38]. Depuis lors, plusieurs publications sur le chiffrement et la signature à base d'identités ont été proposés. En outre, les couplages bilinéaires ont émergé comme une primitive mathématique puissante dont les propriétés permettent de concevoir des schémas cryptographiques ne pouvant pas être réalisés en utilisant les primitives standards [22].

La notion d'identité est clairement centrale aux primitives cryptographiques à base d'identités. Dans le cas du chiffrement à base d'identiés, l'accès à un message chiffré est seulement permis à l'entité dont l'identité est celle par rapport à laquelle le message a été chiffré, tandis que dans le cas de la signature à base d'identités, la validité de la signature sur un message est définie par rapport à l'identité du signataire. Autrement dit, la légitimité d'une entité autorisée à accéder à un message confidentiel et la valeur (en terme de niveau de confiance) d'une signature produite par une entité sont principalement basées sur son identité. Généralement, l'identité n'est pas

suffisante pour l'autorisation et l'établissement de confiance, particulièrement dans le contexte des environnements ouverts à grande échelle tels que l'Internet, où les interactions se produisent souvent entre des entités sans connaissance préalable les unes des autres. Une approche de plus en plus populaire pour déterminer le niveau de confiance des entités communicantes consiste à utiliser des politiques remplies par des certificats numériques.

Autorisation et établissement de confiance à base de politiques

Avec la popularité croissante de l'Internet, les environnements de communication ouverts à grande échelle deviennent de plus en plus répandus. Les caractéristiques de tels environnements rendent les identités des entités communicantes non appropriées ou insuffisantes pour l'autorisation et l'établissement de confiance. En effet, les interactions se produisent souvent entre des entités de différents domaines de sécurité sans connaissance préalable les une des autres. L'identité d'une entité définie dans un certain domaine est sans signification à une autre entité appartenant à un domaine différent et le niveau de confiance à attribuer ne peut ainsi pas être fondé sur l'identité. Des affirmations sur l'entité, dont la validité excède généralement les frontières du domaine de sécurité auquel appartient de l'entité, sont certainement plus appropriées. Dans cette thèse, le terme 'affirmation' signifie un ensemble d'attestations, où chaque attestation peut être un attribut (par exemple le rôle de l'entité dans une organisation), une propriété (par exemple l'entité est un expert dans certains domaines), une autorisation (par exemple l'entité est autorisée à avoir accès à certaines ressources ou à effectuer certaines actions), *etc*. En plus d'un ensemble d'attestations, une affirmation peut contenir des informations suplémentaires additionnelles telles que l'identité de l'entité ou la période de validité de l'affirmation.

Une approche de plus en plus populaire consiste à exprimer les conditions d'autorisation et d'établissement de confiance par des politiques remplies par des certificats numériques (credentials). Une politique se compose principalement d'une combinaison logique de conditions, où chaque condition est remplie par un certificat spécifique. Chaque certificat représente la signature d'une entité de confiance spécifique, appelée l'émetteur du certificat, sur une certaine affirmation au sujet d'une entité, appelée le propriétaire du certificat. Un certificat est généré en utilisant la clé privée de son émetteur et sa validité peut être vérifiée en utilisant la clé publique correspondante. La validité du certificat fournit l'assurance que son propriétaire remplit l'affirmation signée.

Puisque l'identité du destinataire d'un message chiffré n'est pas appropriée pour décider s'il devrait être autorisé à avoir accès au message, un schéma de chiffrement doit être accompagné d'un mécanisme prouvant la conformité du destinataire avec la politique d'autorisation associée au message. De manière analogue, comme l'identité du signataire d'un message n'est pas suffisante pour déterminer le niveau de confiance à lui attribuer, un schéma de signature doit être supporté par un mechanisme prouvant la confirmité du signataire avec une politique d'établissement de confiance définie par le vérificateur de la validité de la signature. Une entité est conforme à une politique si et seulement si elle possède un ensemble de certificats remplissant la combinaison logique de conditions spécifiées par la politique. L'approche classique pour obtenir une preuve de la conformité d'une entité avec une politique consiste en trois étapes: 1) obtenir un ensemble de certificats de l'entité, 2) vérifier la validité de chaque certificat par rapport à la clé publique de son émetteur, 3) vérifier que l'ensemble des certificats remplie la

combinaison logique de conditions spécifiées par la politique.

Pour synthétiser, la notion de politique est centrale à l'autorisation et à l'établissement de confiance, alors que la notion d'identité, centrales aux primitives cryptographiques à base d'identités est obsolète. Il serait ainsi logique et intéressant de songer à développer des primitives cryptographiques à base de politiques plutôt qu'à base d'identités. Au lieu de chiffrer un message par rapport à l'identité de l'entité à laquelle le message est destiné, le message est chiffré par rapport à la politique qui doit être accomplie par le destinataire afin qu'il soit autorisé à avoir accès au message. De manière similaire, au lieu de vérifier la validité d'une signature par rapport à l'identité de l'entité qui l'a produite, la validité de la signature est vérifiée par rapport à la politique qui doit être accomplie par le signataire afin que la signature atteigne un niveau de confiance acceptable. C'est l'idée principale derrière le concept de la cryptographie à base de politiques développé au cours de cette thèse.

Cryptographie à base de politiques

Dans cette thèse, nous formalisons le concept de 'cryptographie à base de politiques' en définissant deux primitives: chiffrement à base de politiques et signature à base de politiques. Un schéma de chiffrement à base de politiques permet de chiffrer un message par rapport à une politique de telle sorte que seule une entité qui est conforme avec la politique puisse déchiffrer le message. Notre primitive de chiffrement à base de politiques permet ainsi de réaliser à la fois les services de confidentialité et d'autorisation. De manière analogue, un schéma de signature à base de politiques permet de signer un message par rapport à une politique de telle sorte que la signature soit considérée comme valide si et seulement si elle ait été produite par une entité qui est conforme avec la politique en question. Notre primitive de signature à base de politiques permet ainsi de à la fois les services d'intégrité, de non répudiation et d'établissement de confiance.

Dans notre concept de cryptographie à base de politiques, une politique se compose de conjonctions et de disjonctions de conditions, où chaque condition est remplie par un certificat spécifique. Contrairement à l'approche classique où une entité doit révéler ses certificats afin de prouver sa conformité avec une politique d'autorisation ou d'établissement de confiance, les certificats dans la cryptographie à base de politiques sont privés puisqu'ils sont utilisés comme des clés de déchiffrement et de signature. En effet, un ensemble de certificats remplissant la combinaison logique de conditions spécifiées par une politique est exigé pour déchiffrer un message qui est chiffré par rapport à la politique, alors qu'un ensemble de certificats remplissant la combinaison logique de conditions spécifiées par une politique est exigé pour produire d'une signature valide par rapport à la politique.

Notre objectif dans cette thèse est triple:

- Définir formellement les primitives de chiffrement et de signature à base de politiques et concevoir des réalisations concrètes de ces deux primitives. Compte tenu du fait qu'une politique est formalisée comme une expression booléenne monotone, le défit principal est celui de trouver une méthode élégante pour convertir les opérateurs logiques spécifiés par la politique en opérations mathématiques dans les algorithmes de chiffrement/déchiffrement et de signature/vérification de signature.

- Définir formellement des modèles de sécurité adaptés à nos primitives cryptographiques à base de politiques et prouver la sécurité des schémas proposés dans le cadre de ces modèles. L'idée ici est de considérer des notions de sécurité bien étudiées dans la littérature liées aux schémas de chiffrement et de signature et de les adapter pour tenir compte des spécificités de nos primitives cryptographiques.

- Valider l'utilité du concept proposé de cryptographie à base de politiques en montrons comment nos primitives cryptographiques peuvent être utilisées pour résoudre des problèmes pratiques bien établis dans différents contextes, tout en améliorant les solutions existantes.

Organisation et contributions de la thèse

Ce rapport de thèse est organisé comme suit:

- Dans Chapitre 0, nous passons en revue les connaissances de base liées, d'une part, à la cryptographie standard et au concept de la sécurité prouvée et, d'autre part, aux couplages bilinéaires sur des courbes elliptiques et à leurs applications récentes en cryptographie. Le but de ce chapitre n'est naturellement pas de re-écrire ce qui peut être facilement trouvé dans plusieurs références dans la littérature. Au lieu de cela, il vise à présenter les différentes notions et notations cryptographiques qui seront utilisées dans la suite du rapport.

  Après avoir lu Chapitre 0, le lecteur peut ensuite lire soit Chapitre-1 puis Chapitre 2 qui, tous deux, traitent le chiffrement à base de politiques, soit lire Chapitre 3 qui traite la signature à base de politiques.

- Dans Chapitre 1, nous étudions la primitive de chiffrement à base de politiques. Le chapitre se compose de deux parties indépendantes mais complémentaires. Dans la première partie, nous étudions formellement le chiffrement à base de politiques, alors que dans la deuxième partie nous illustrons ses applications et ses propriétés particulières dans trois contextes différents. Après avoir formellement défini la primitive de chiffrement à base de politiques, nous présentons un nouveau modèle de sécurité associé qui tient en compte les spécificités de la cryptographie à base de politiques. Puis, nous présentons un schéma élégant et relativement efficace de chiffrement à base de politiques, dont la sécurité est prouvée dans le modèle de l'oracle aléatoire. L'application la plus intuitive du chiffrement à base de politiques est dans le cadre du contrôle d'accès. Dans ce contexte, nous présentons un framework basé sur notre primitive de chiffrement pour le contrôle d'accès aux documents XML. Notre solution surclasse les solutions existantes dans plusieurs domaines et notamment celui de la gestion des clés de chiffrement. Notre deuxième application du chiffrement à base de politiques est dans le contexte du renforcement des politiques de protection de la vie privée. Dans ce cadre, nous montrons comment notre primitive de chiffrement peut être employée pour la réalisation du paradigme dit: 'sticky policy paradigm'. En troisième lieu, nous montrons comment notre primitive peut être utilisée dans le contexte de l'établissement des communautés ad-hoc en respectant un principe fondamental de protection de la vie privée, celui de la minimisation des données.

- Dans Chapitre 2, nous traitons une propriété inhérente à la primitive de chiffrement à base de politiques qui est celle de son exposition, dans certains contextes, à des attaques de collusions: en plus d'une entité légitime, n'importe quel groupe d'entités pouvant former, en joignant leurs certificats, un groupe de certificats remplissant une politique par rapport à laquelle un message a été chiffré, peut déchiffrer le message. Comme alternative, nous présentons une variante de la primitive de chiffrement à base de politiques, appelée chiffrement à clé publique à base de politiques. Après avoir formellement défini la nouvelle primitive et le modèle de sécurité correspondant, nous présentons une réalisation concrète utilisant les couplages bilinéaires et prouvons sa sécurité dans le modèle de l'oracle aléatoire. Puis, nous montrons comment le chiffrement à clé publique à base de politiques peut être utilisé pour résoudre le problème d'inter-dépendance cyclique de politiques inhérent aux approches standards dans le cadre de la négociation de confiance.

- Dans Chapitre 3, nous étudions la primitive de signature à base de politiques. Comme dans Chapitre 1 et Chapitre 2, ce chapitre se compose à son tour de deux parties complémentaires mais indépendantes. Après avoir formellement défini la primitive de signature à base de politiques et les modèles de sécurité associés, nous présentons une réalisation concrète de cette primitive utilisant les couplages bilinéaires. Comme application de notre primitive, nous présentons une nouvelle forme de certification intermédiaire (de procuration), appelée certification intermédiaire à preuve.

Le travail de recherche effectué par l'auteur a donné lieu à un certain nombre de publications scientifiques dont certaines contiennent les idées principales présentées dans ce rapport de thèse [17, 19, 18, 16, 15], et d'autres contiennent d'autres contributions non liées à la cryptographie à base de politiques [13, 14].

Dans le reste de ce résumé, nous présentons chacune des primitives proposées dans cette thèse.

Chiffrement à base de politiques

Parce que l'identité de l'entité qui reçoit un message chiffré n'est pas suffisante pour décider si cette entité peut être autorisée à avoir accès au message en clair, les schémas de chiffrement à clé publique ou à base d'identités doivent être utilisés en combinaison avec un mécanisme prouvant la conformité du récepteur du message avec la politique d'autorisation définie par le propriétaire du message. Dans ce cadre, l'approche standard est que le propriétaire du message reçoit tout d'abord un ensemble de certificats de la part de l'entité voulant accéder au message. A la réception des certificats, le propriétaire vérifie la validité de chacun par rapport à la clé publique de son émetteur et vérifie ensuite que l'ensemble des certificats reçs remplie la combinaison logique de conditions spécifiées par sa politique d'autorisation associée au message. Selon cette approche, deux mécanismes séparés sont nécessaires pour assurer les services de confidentialité et d'autorisation. Ici, la notion d'identité, qui est centrale à la cryptographie à base d'identités, joue tout simplement le rôle de lien entre les deux mécanismes.

Notre primitive de chiffrement à base de politiques permet de fournir les services de confidentialité et d'autorisation à l'aide d'un seul mécanisme. En effet, un algorithme de chiffrement à base de politique permet de chiffrer un message par rapport à une politique de telle sorte

que seule une entité qui est conforme à la politique en question peut déchiffrer le message. Contrairement à l'approche standard où une entité est obligée de dévoiler ses certificats pour prouver sa conformité avec une politique d'autorisation donnée, les certificats dans notre approche sont privés puisqu'ils sont utilisés comme des clés de déchiffrement. Intuitivement, la notion de politique est pour le chiffrement à base de politiques ce qu'est la notion d'identité pour le chiffrement à base d'identités.

Une illustration de notre primitive de chiffrement à base de politiques est donnée dans Figure 1.

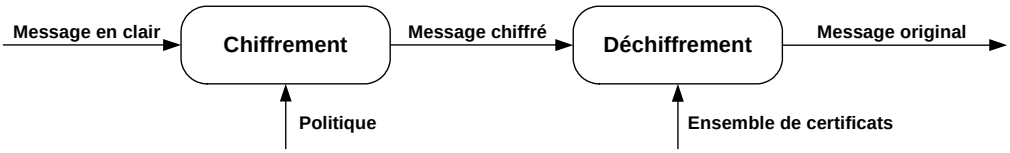


Figure 1: Chiffrement à base de politiques

Une primitive de chiffrement à base de politiques doit respecter les propriétés suivantes:

- **confidentialité à base de politiques**: contrairement au chiffrement à base d'identités où la confidentialité d'un message chiffré est basée sur l'identité de l'entité à laquelle le message est destiné, la confidentialité d'un message chiffré en utilisant un algorithme de chiffrement à base de politiques doit être basée sur la conformité de l'entité recevant le message avec la politique d'autorisation associée au message. Cette première propriété traduit le fait que le chiffrement à base de politiques assure aussi bien la confidentialité que l'autorisation en une seule opération logique (le déchiffrement). Par soucis de normalisation, les politiques considérées dans cette thèse sont formalisées comme des expressions booléennes monotones écrites sous formes standards.

- **émetteurs de certificats multiples**: une politique par rapport à laquelle un message est chiffré doit pouvoir supporter des certificats émis par de multiples autorités de certifications indépendantes, où chaque émetteur représente un domaine de sécurité ou un espace de responsabilité autonome et limité. Ceci est dû au fait qu'une autorité de certification centralisée ne peut pas être responsable de la vérification et la validation de tout type d'affirmations à propos de toutes les entités communicantes. En outre, le propriétaire d'une message peut exiger, par exemple, que pour qu'une affirmation sur une entité soit considérée comme valide, elle doit être certifiée par deux autorités de certification indépendantes. De plus, deux entités peuvent avoir confiance en deux autorités de certifications différentes pour la vérification de la validité de la même affirmation.

- **indépendance de l'émission des certificats**: l'algorithme de chiffrement doit être conçu de telle sorte que le chiffrement d'un message soit indépendant de l'émission des certificats remplissant la politique par rapport à laquelle le message est chiffré. En d'autres termes, il doit être possible qu'un message soit chiffré avant même que l'information requise pour son déchiffrement soit générée. En effet, certains certificats ne peuvent être issus qu'une fois une action ait été accomplie ou un évènement ait eu lieu et ceci ne doit pas affecter le moment auquel un message puisse être chiffré.

- **représentation publique des politiques**: l'information publique requise par l'algorithme de chiffrement à base de politiques doit être facilement dérivable de la représentation publique de la politique par rapport à laquelle un message est chiffré. Cette propriété est intuitivement équivalente à celle requise par le chiffrement à base d'identités où l'information publique utilisée par l'algorithme de chiffrement est facilement dérivable de la représentation publique de l'identité de l'entité à laquelle le message est destiné.
- **performance**: une réalisation concrète du chiffrement à base de politiques peut être effectuée à partir d'un schéma de chiffrement à base d'identités. Ceci est effectué par une approche 'naive' qui consiste à effectuer plusieurs chiffrements pour traduire les disjonctions et des chiffrements type-onion pour traduire les conjonctions, tout en remplaçant les identités par des affirmations. L'efficacité d'une telle approche est clairement non satisfaisante aussi bien du point de vue coût de calculs que du point de vue taille du message chiffré, particulièrement lorsque les politiques par rapport auxquelles les messages sont chiffrés deviennent complexes. Une réalisation concrète du chiffrement à base de politiques doit ainsi être plus efficace que cette approche.
- **sécurité prouvée**: la sécurité d'un schéma de chiffrement à base de politiques doit être prouvée dans le cadre d'un modèle de sécurité bien défini qui prend en compte les spécificités de la cryptographie à base de politiques. Dans cette thèse, nous considérons la sécurité de nos schémas dans le contexte de l'oracle aléatoire.

Dans Chapitre 2, nous formalisons le concept de chiffrement à base de politiques, nous proposons une réalisation concrète de cette primitive remplissant toutes les propriétés décrites ci-dessus et nous montrons l'utilité de notre primitive en décrivant trois applications. Le chapitre est organisé comme suit: dans Section 1.2, nous discutons un certain nombre de primitives cryptographiques avancées liées à notre primitive de chiffrement à base de politiques. Dans Section 1.3, nous présentons notre modèle formel de politiques et la terminologie associée. Puis, nous définissons formellement le chiffrement à base de politiques et présentons un modèle de sécurité associé. Dans Section 1.4, nous décrivons d'abord notre schéma de chiffrement à base de politiques utilisant les couplages bilinéaires sur des courbes elliptiques. Puis, nous discutons son efficacité avant d'analyser sa sécurité dans le modèle de l'oracle aléatoire. Dans Section 1.5, nous présentons un framework pour le contrôle d'accès aux documents XML utilisant notre primitive de chiffrement. Cette application illustre comment notre primitive peut être d'une manière élégante pour protéger les documents à structure arborescente. Dans Section 1.6.3, nous montrons, dans le contexte de la protection des données privées, comment notre primitive de chiffrement peut être utilisée en combinaison avec d'autres outils pour la mise en oeuvre du paradigme dit 'sticky policy paradigm'. Enfin, dans Section 1.7, nous montrons comment notre primitive peut être utilisée pour l'établissement de communautés ad-hoc. Plus précisément, nous nous focalisons sur le principe minimisation des données qui ne peut nullement être accompli en utilisant un mécanisme standard impliquant l'échange de certificats.

Chiffrement à base de politiques sans collusions

La primitive de chiffrement à base de politiques présentée dans Chapitre 2 est basée implicitement sur deux hypothèses: d'abord, les émetteurs de certificats ne sont pas intéressés à ac-

céder aux messages échangés par les utilisateurs. Ensuite, les utilisateurs ne sont pas disposés à partager leurs certificats avec d'autres utilisateurs. Ces deux hypothèses de sécurité sont considérées comme acceptables dans certains contextes, en particulier dans le cadre des communications 1-à-n. Par contre, elles ne peuvent être admises dans d'autres contextes où les contraintes de sécurité sont plus strictes. En effet, dans de tels contextes, deux types d'attaques peuvent survenir:

- **collusion entre émetteurs de certificats**: en plus d'une entité légitime, n'importe quel ensemble d'émetteurs de certificats qui collaborent pour former un ensemble de certificats remplissant la politique peut également déchiffrer le message.

- **collusion entre utilisateurs**: deux ou plusieurs utilisateurs qui peuvent collaborer pour former un ensemble de certificats remplissant une politique avec laquelle aucun n'est en conformité peuvent déchiffrer n'importe quel message chiffré par rapport à la politique.

Pour éviter les collusions entre utilisateurs, une solution consiste à lier systématiquement chaque certificat à un identifiant vérifiable de son propriétaire. La politique par rapport à laquelle un message est chiffré est telle que les différentes affirmations incluent un identifiant vérifiable de l'entité à laquelle le message est destiné. Ici, 'l'identifiant vérifiable' signifie qu'il existe un protocole permettant à l'entité qui chiffre le message (l'expéditeur) de vérifier que l'identifiant indiqué par la politique selon laquelle le message sera chiffré correspond au destinataire prévu. Dans cette perspective, on devrait assumer que chaque identifiant correspond à une entité unique, alors qu'une entité peut avoir plusieurs identifiants. Nous soulignons le fait que la confidentialité des messages chiffrés n'est pas basée sur l'identifiant de l'entité pour laquelle le message est destiné comme dans l'approche classique du chiffrement à base d'identités, mais sur sa conformité avec la politique par rapport à laquelle le message est chiffré. Les identifiants sont juste employés pour assurer l'unicité des certificats émis. Ci-dessous, nous décrivons deux stratégies possibles d'identification:

- **identification par pseudonyme**: les entités communicantes peuvent être identifiées par des pseudonymes (par exemple nom régional, adresse IP, identifiant aléatoire, *etc*). Comme dans beaucoup de systèmes de certification [114, 43, 46], on peut assumer l'existence d'une autorité de pseudonymes qui contrôle l'attribution des pseudonymes aux différentes entités. L'autorité de pseudonymes peut jouer le rôle d'un émetteur de certificats particulier qui émet des certificats de pseudonymes représentant la signature de l'autorité de pseudonymes sur le pseudonyme assigné à une entité. Dans ce cas, l'authenticité d'une entité identifiée par un certain pseudonyme est contrainte par la possession du certificat correspondant au pseudonyme, qui est secrètement gardé par l'éntité. Une approche élégante pour combiner l'authentification avec le chiffrement à base de politiques consiste à ajouter systématiquement une condition remplie par un certificat de pseudonyme du destinataire prévu à la politique selon laquelle un message doit être chiffré.

- **identification par clé**: une autre approche consiste à supposer que chaque entité détient une paire de clés publique/privée aléatoirement généré. Dans ce cas, une entité peut être simplement identifiée par sa clé publique. Ainsi, l'authenticité d'une entité identifiée par une clé publique est contrainte par la possession de la clé privée correspondante. On

assume que la clé privée est de valeur de telle sorte qu'elle n'est jamais révélée par son propriétaire.

Lier chaque certificat à un identifiant vérifiable n'est pas suffisant pour surmonter des collusions potentielles entre les émetteurs de certificats. En effet, afin d'éviter ce genre d'attaques, en plus de l'ensemble de certificats remplissant la politique par rapport à laquelle un message est chiffré, l'algorithme de déchiffrement à base de politiques doit impliquer un élément secret connu seulement par le destinataire légitime du message. Dans cette perspective, une clé privée qui soit secrètement connue par le destinataire légitime peut jouer le rôle d'un tel élément secret.

Nous présentons une primitive de chiffrement à base de politiques sans collusions qui permet de surmonter les imperfections de la primitive originale de chiffrement à base de politiques présentée dans Chapitre 1. La nouvelle primitive, dite chiffrement à clé publique à base de politiques, combine les fonctionnalités du chiffrement original à base de politiques et du chiffrement classique à clé publique. Ainsi, il permet de chiffrer un message par rapport à une politique et une clé publique d'une manière telle que seulement une entité ayant accès non seulement à un ensemble de certificats remplissant la politique mais également à la clé privée associée à la clé publique utilisée peut déchiffrer le message.

Une illustration de notre primitive de chiffrement à clé publique à base de politiques est donnée dans Figure 2

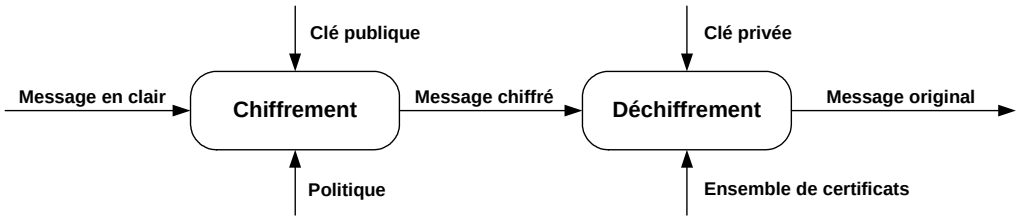


Figure 2: Chiffrement à base de politiques sans collusions

Dans Chapitre 2, nous proposons une réalisation concrète du chiffrement à clé publique à base de politiques utilisant les couplages bilinéaires. Le schéma de chiffrement proposé est intuitivement semblable à celui proposé dans Chapitre 1. Cependant, un soin particulier doit être de rigueur afin de préserver le même niveau de sécurité. Ce chapitre est organisé comme suit: dans Section 2.2, nous discutons l'état de l'art. Dans Section 2.3, nous présentons d'abord notre modèle de politiques et la terminologie associée. Puis, nous définissons formellement notre primitive de chiffrement à clé publique à base de politiques avant de décrire le modèle de sécurité associé. Dans Section 2.4, nous décrivons d'abord notre schéma de chiffrement à clé publique à base de politiques. Puis, nous discutons sa performance avant de prouver sa sécurité dans le cadre du modèle de l'oracle aléatoire. Enfin, dans Section 2.5, nous montrons comment la primitive de chiffrement à clé publique à base de politiques peut être employée dans le contexte de la négociation de confiance automatisée. Principalement, nous suggérons de remplacer le schéma de chiffrement proposé dans [92], qui souffre (au même titre que la primitive originale de chiffrement à base de politiques) de l'exposition aux attaques de collusion, par notre primitive de chiffrement à clé publique à base de politiques.

Signature à base de politiques

L'identité d'une entité n'est généralement pas suffisante pour déterminer le niveau de confiance à attribuer à une signature qu'elle produit, surtout dans le contexte des environnements ouverts à grande échelle comme l'Internet, puisque dans de tels environnements, les interactions se produisent souvent entre les entités de différents domaines de sécurité sans connaissance préalable l'un de l'autre. Comme dans le cas de l'autorisation, une approche de plus en plus populaire consiste à exprimer les conditions d'établissement de confiance par des politiques remplies par des certificats numériques. En conséquence, un schéma de signature standards ou à base d'identités doivent être employés en combinaison avec un mécanisme qui prouve la conformité du signataire avec une politique d'établissement de confiance définie par le vérificateur de la signature. L'approche standard est que le vérificateur de la signature reçoit d'abord un ensemble de certificats du signataire. Puis, il vérifie la validité de chacun des certificats reçus avant de valider le fait que l'ensemble des certificats remplie sa politique. Selon cette approche, au moins deux mécanismes séparés sont exigés pour fournir l'établissement de confiance ainsi que l'intégrité et la non répudiation.

Le but de notre primitive de signature à base de politiques est de réaliser l'intégrité, la non répudiation et l'établissement de confiance en une seule opération logique. Un schéma de signature à base de politiques permet à une entité de produire une signature sur un message par rapport à une clé publique et une politique d'une manière telle que la signature soit valide si et seulement si l'entité possède un ensemble de certificats remplissant la politique ainsi que la clé privée correspondant à la clé publique en question. La validité de la signature fournit ainsi une preuve de la conformité du signataire avec la politique selon laquelle la signature a été produite.

Une illustration de notre primitive de signature à base de politiques est donnée dans Figure 3.

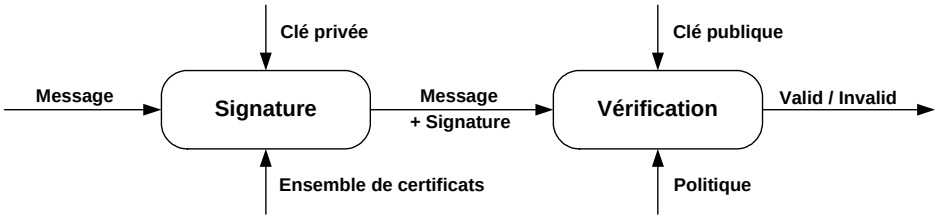


Figure 3: Signature à base de politiques

Une primitive de signature à base de politiques doit respecter les propriétés suivantes:

- **preuve de conformité**: la validité d'une signature produite en utilisant un schéma de signature à base de politiques représente la preuve de conformité du signataire avec la politique selon laquelle la signature a été produite. En d'autres termes, la signature est valide si et seulement si elle a été produite en utilisant un ensemble de certificats remplissant la politique.

- **non répudiation**: la validité d'une signature produite en utilisant un schéma de signature à base de politiques est également définie par rapport à une clé publique. Une signature

valide peut être produite seulement par une entité ayant accès à la clé privée correspondante. Cette condition vise à empêcher une entité de nier des engagements ou des actions précédentes convenus par la procédure de signature.

- **intégrité**: comme pour la signature standard, cette propriété permet de protéger un message signé contre toute modification au cours de sa transmission.
- **ambiguité des certificats**: une signature valide fournit une preuve que le signataire est conforme avec la politique par rapport à laquelle elle a été produite. Dans le cas où la politique se compose de disjonctions de conditions, l'entité de vérification ne devrait pas pouvoir connaître quels certificats spécifiques ont été employés pour produire de la signature.
- **réalisation**: contrairement au chiffrement à base de politiques qui peut être réalisé en utilisant un schéma de chiffrement à base de politiques une réalisation concrète de la primitive de signature à base de politiques ne peut pas être accomplie en utilisant un schéma de signature à base d'identités. En effet, alors que les conjonctions de conditions peuvent être réalisées en utilisant les signatures multiples, des disjonctions ne peuvent pas être réalisées. Intuitivement, la structure de disjonction dans une politique est semblable à la structure d'anneau dans les signatures d'anneaux [131]. Une méthode élégante devrait être trouvée pour adapter efficacement la structure d'anneau au contexte des politiques complexes formalisées en expressions booléennes monotones écrites sous des formes normales.
- **sécurité prouvée**: la sécurité d'un schéma de signature à base de politiques doit être prouvée dans le cadre de modèles de sécurité bien définis qui prennent en compte les propriétés spécifique de la cryptographie à base de politiques.

Dans Chapitre 3, nous formalisons le concept de signature à base de politiques, nous proposons une réalisation concrète de cette primitive et nous prouvons sa sécurité avant de montrer son utilité en décrivant un scénario d'application original. Le reste du chapitre est organisé comme suit: dans Section 3.2 nous discutons l'état de l'art. Dans Section 3.3, nous définissons formellement la primitive de signature à base de politiques et les modèles de sécurité associés. Dans Section 3.4, nous décrivons d'abord notre schéma de signature à base de politiques. Puis, nous discutons ses performances avant de prouver sa sécurité dans le cadre du modèle de l'oracle aléatoire. Enfin, dans Section 3.5, nous présentons une forme originale de certificats, appelée les certificats intermédiaires à preuve, qui peuvent être réalisés en utilisant la primitive de signature à base de politiques.

Conclusion

Pour conclure, dans cette thèse nous avons formalisé le concept original de cryptographie à base de politiques en traitant aussi bien les aspects théoriques que pratiques de ce concept. Notre travail peut être vu sous trois angles. Le premier est que notre concept introduit une nouvelle manière de concevoir les primitives cryptographiques en adéquation avec les besoins réels des environnements de communication. Dans ces derniers, la notion d'identité laisse la place à la

notion de politique pour l'autorisation et l'établissement de confiance. Le second est que notre concept peut être vu comme une généralisation du concept de cryptographie à base d'identités. En effet, une primitive cryptographique à base d'identités est une primitive à base de politiques où les politiques sont réduites à une condition remplies par un certificat d'identité. Le troisième est que nos primitives cryptographiques représentent une nouvelle illustration de l'utilité des couplages bilinéaires dans la conception de primitives cryptographiques. La cryptographie à base de politique ne remplacera pas la cryptographie conventionnelle mais doit être vue, au même titre que la cryptographie à base d'identités comme un bon complément.

Contents

Abstract

‘Identity-based cryptography’ is definitely one of the most popular topics addressed by the cryptographic research community in the last five years. As can be guessed from the name, the notion of ‘identity’ is central to identity-based cryptographic primitives. As for identity-based encryption, access to an encrypted message is only permitted to the entity whose identity is the one according to which the message was encrypted, whereas for identity-based signature, the validity of the signature on a message is defined with respect to the identity of the entity that generated the signature. In general, identity is not sufficient for authorization and trust establishment, especially in the context of large-scale open environments like the Internet, where interactions often occur between parties with no pre-existing familiarity of one another. An increasingly popular approach to determining the trustworthiness of the interacting entities consists in using policies fulfilled by digital credentials.

In this thesis, we present a new concept in cryptography, called ‘policy-based cryptography’, which allows to perform cryptographic operations with respect to policies fulfilled by digital credentials. Intuitively, a policy-based encryption scheme allows to encrypt a message with respect to a policy so that only an entity that is compliant with the policy can decrypt the message. Similarly, a policy-based signature scheme allows to generate a signature on a message with respect to a policy so that the signature is valid if and only if it was generated by an entity that is compliant with the policy. We present three policy-based cryptographic primitives from bilinear pairings over elliptic curves and prove their security under well-defined security models. We further illustrate the usefulness of our concept of policy-based cryptography through the description of application scenarios in the contexts of access control, privacy policy enforcement, establishment of ad-hoc communities, automated trust negotiation and proxy certification.

Introduction

Writers say: "To write a good book, first tell them what you are going to tell them, then tell them, finally tell them what you told them"
- Anonymous -

Cryptography is a branch of information and communication security providing a set of mathematical primitives and mechanisms used to implement a set of security services. The main contribution of this thesis is the formalization of the concept of 'policy-based cryptography'. Our concept allows to combine cryptography with the notion of policy, which is one of the fundamental security notions. A policy is typically defined through a combination of rules or conditions that must be met (or fulfilled) in order to achieve a pre-defined security objective. A policy-based cryptographic primitive can thus be simply viewed as a cryptographic primitive that implements one or more security services, while providing an assurance of the compliance with a specific policy.

In this introductory chapter, we first provide an overview of the standard approach in cryptography, where the notion of 'identity' plays a central role. We then motivate our concept of policy-based cryptography, where the notion of 'identity' paves the way to the notion of 'policy'. Finally, we outline the structure of this manuscript while highlighting the contributions of this thesis.

Standard Cryptography

For many years, cryptography was the exclusive domain of military, diplomatic and governmental secret services. The proliferation of computers and networks in the sixties brought with it a strong demand from the private sector for securing digital information and communications. In the early seventies, the work of IBM on symmetric encryption has led to the adoption of the Data Encryption Standard (DES) as a U.S. Federal Information Processing Standard [123]. This standard is today considered as the most-well known mechanism in the history of modern cryptography. It is widely used for securing electronic commerce for several financial institutions around the world [116]. Informally, a symmetric encryption scheme allows to encrypt a message using a key in way such that only an entity having access to the key is able to decrypt the message. This means that a common key must be shared between an entity that encrypts a message and an entity that is able to decrypt the encrypted message. One of the major issues with symmetric encryption therefore is to find an efficient way to agree upon and exchange the encryption keys securely. This problem is referred to as the key distribution problem.

In 1976, Diffie and Hellman introduced the concept of asymmetric cryptography, considered as the most striking development in the history of modern cryptography [63]. The primary goal was to overcome the key distribution problem inherent to symmetric cryptography. Although the authors did not propose a practical realization of symmetric encryption, their publication has spurred an extensive activity in the cryptographic research community. Also referred to as public-key cryptography, their concept allows to construct encryption schemes that do not require the secure exchange of keys. Furthermore, it allows to construct a variety of more or less sophisticated cryptographic primitives providing security services other than confidentiality, the most significant of which is definitely the digital signature primitive. More details about the public-key encryption and signature primitives are given below:

- **public-key encryption**: this primitive is mainly used to implement the confidentiality security service. A public-key encryption scheme allows to encrypt a message using a public key in such a way that only an entity having access to the corresponding private key can decrypt the message. In 1978, the first practical public-key encryption scheme was presented by Rivest, Shamir and Adleman [130]. Also referred to as RSA, its security is based on the hardness of factoring large integers. In 1985, another practical public-key encryption scheme based on the intractability of the discrete logarithm problem was proposed by El Gamal [82].
- **digital signature**: this primitive is used to implement the integrity, non-repudiation and authenticity security services. A (public-key) signature scheme allows to generate a signature on a message in such a way that the signature is valid with respect to a public key if and only if it was generated using the corresponding private key. In 1991, the first international standard for digital signatures was adopted (ISO/IEC9796). It is based on the RSA public-key encryption scheme. Two years later, the U.S. Government adopted the Digital Signature Standard, referred to as DSS, as a standard signature mechanism based on the El Gamal public-key encryption scheme (FIPS 186).

In its standard setting, a public key appears as random bit string so that there is nothing about it that indicates to whom it belongs. One of the major issues faced by standard public-key encryption and signature schemes is therefore the deployment and management of infrastructures supporting the authenticity of public keys. Basically, one needs to set up a method that provides an assurance to the entity that encrypts the message about the relationship between the public key and a legitimate holder of the corresponding private key.

The most popular method for authenticating public keys consists in using public-key certificates, which were first introduced by Kohnfelder in [102]. A public-key certificate is basically the signature of a trusted entity, called certification authority, on an assertion that binds a public key to identity of the legitimate holder of the corresponding private key. By signing the assertion, the certification authority attests that the public key contained in the assertion belongs to the entity whose identity is the one specified by the assertion. In this widely accepted approach, it is assumed that an entity who sends an encrypted message to another entity knows the identity of the recipient the message is intended for. Before encrypting a message using a certain public key, an entity needs thus to obtain a valid public-key certificate providing the assurance of the relationship between the public key and the identity of the legitimate recipient. Similarly, it is assumed that an entity receiving a signature on a message knows the identity of the entity that generated it and has the assurance, through public-key certification, of the relation between the identity of the signer and the public key according to which the signature is considered as valid. A security infrastructure whose services are implemented to deploy and manage the use of public key certificates is called public-key infrastructure, commonly denoted by PKI. We refer the reader to [2] for a comprehensive description of public-key infrastructures. An elegant alternative to public-key certification is offered by identity-based cryptography.

Identity-Based Cryptography

The concept of identity-based cryptography was first formulated by Shamir in 1984 [137]. Its original goal was to eliminate the need for public-key certificates by using the identity of an entity as its public key. The corresponding private key is generated by a trusted authority called private key generator and is issued to the entity through a secure channel. An identity-based encryption scheme allows thus to encrypt a message with respect to an identity in such a way that only an entity whose identity is the one according to which the message was encrypted can decrypt the message. Similarly, an identity-based signature scheme allows to generate a signature on a message in such a way that the signature is valid only with respect to the identity of the entity that generated it.

Efficient solutions for identity-based signature schemes were quickly found [71, 69], whilst the design of identity-based encryption schemes turned out to be more challenging. In fact, no identity-based encryption scheme proposed between 1984 and 2000 was fully satisfactory in terms of both security and efficiency. In 2001, Boneh and Franklin managed to develop a practical and provably secure identity-based encryption scheme using bilinear pairings over

elliptic curves [38]. Since then, several publications on identity-based encryption and signature schemes have been proposed [71, 69]. Besides, it has emerged that bilinear pairings from elliptic curves represent a very powerful mathematical primitive that can be used to build novel cryptographic schemes achieving properties or performances that could not be achieved using standard cryptographic primitives [22].

The notion of 'identity' is central to the identity-based cryptographic primitives. As for identity-based encryption, access to an encrypted message is only permitted to the entity whose identity is the one according to which the message was encrypted, whereas for identity-based signature, the validity of the signature on a message is defined with respect to the identity of the entity that generated the signature. Said differently, the legitimacy of an entity that is allowed to access a confidential message and the trustworthiness of the signatures it generates are solely based on its identity. In general, identity is not sufficient for authorization and trust establishment, especially in the context of large-scale open environments like the Internet, where interactions often occur between parties with no pre-existing familiarity of one another. An increasingly popular approach to determining the trustworthiness of the interacting entities consists in using policies fulfilled by digital credentials, which we discuss in the following section.

From 'Identity' to 'Policy'

With the growing popularity of the Internet, open large-scale communication environments are becoming increasingly prevalent. The characteristics of such environments make the identities of the communicating entities either not relevant or not sufficient for authorization and trust establishment. Indeed, it is often the case that interactions occur between entities from different security domains without pre-existing knowledge of each other. The identity of an entity defined in some domain is meaningless to a communication partner belonging to a different domain and the trustworthiness can thus not be based on identity. Instead, assertions fulfilled by the entity, whose validity generally exceeds the scope of the security domain of the entity, are definitely more relevant. In this thesis, the term 'assertion' means a set of statements, where each statement can be an attribute (e.g. the role of the entity within an organization), a property (e.g. the entity is an expert on certain topics), an authorization (e.g. the entity is allowed to have access to certain resources or to perform certain actions), *etc*. Together with the set of statements, the assertion may optionally contain additional information such as the identity of the entity or the validity period of the assertion.

An increasingly popular approach consists in expressing authorization and trust requirements through policies fulfilled by digital credentials. A policy consists of a logical combination of conditions, where each condition is fulfilled by a specific digital credential, or simply a credential. Each credential is the signature of a specific trusted entity, called credential issuer, on a certain assertion about an entity, called credential owner. It is generated using the credential issuer's private key and its validity can be verified using the credential issuer's public key. The validity of the credential proves that a trusted entity, namely the credential issuer, certifies

that the credential owner fulfills the signed assertion. Several credential systems with different features and properties are proposed in the literature. Examples are the X.509 attribute certificates [93], SPKI certificates [67], hidden credentials [92], private credentials [44], anonymous credentials [46, 47, 51, 115, 9], *etc.*

Because the identity of the recipient of an encrypted message is not relevant to decide whether it should be authorized to have access to the message, identity-based encryption need to be used subsequently to a mechanism that proves the compliance of the recipient with the authorization policy defined by the owner of the message. Similarly, as the identity of the signer of a message is not sufficient to decide about the trustworthiness of the generated signature, identity-based signature needs to be used in combination with a mechanism that proves the compliance of the signer with a trust establishment policy defined by the verifier of the signature. An entity is compliant with a policy if and only if it has access to a qualified set of credentials for the policy i.e. a set of credentials fulfilling the combination of conditions specified by the policy. The standard approach for getting a proof of the compliance of an entity with a policy consists of three stages: 1) receiving a qualified set of credentials for the policy from the entity, 2) verifying the validity of each of the received credentials, 3) checking that the received set of credentials is effectively a qualified set of credentials for the policy.

The notion of 'policy' is central to authorization and trust establishment, whereas the notion of 'identity' carried by the identity-based cryptographic primitives is obsolete. It would therefore be interesting to develop cryptographic primitives that are based on the former rather than on the latter. Instead of encrypting a message using the identity of the recipient the message is intended for, the message is encrypted with respect to the policy that must be fulfilled by the recipient in order for it to be authorized to have access to the message. Similarly, instead of verifying the validity of a signature with respect to the identity of the entity that generated it, the signature is verified with respect to the policy that must be fulfilled by the signer in order for the signature to be accepted. This is the main idea behind the concept of policy-based cryptography presented in this manuscript.

Policy-Based Cryptography

In this thesis, we formalize the concept of 'policy-based cryptography' through the definition of two primitives: policy-based encryption and policy-based signature. A policy-based encryption scheme allows to encrypt a message with respect to a policy so that only an entity that is compliant with the policy can decrypt the message. The policy-based encryption primitive achieves thus confidentiality and authorization in a logically single step. Similarly, a policy-based signature scheme allows to generate a signature on a message with respect to a policy in such a way that the signature is valid if and only if it was generated by an entity that is compliant with the policy. The policy-based signature primitive achieves integrity, non-repudiation and trust establishment in a single cryptographic operation as well.

In policy-based cryptography, a policy consists of conjunctions and disjunctions of conditions, where each condition is fulfilled by a specific credential. In contrast with the standard approach where an entity needs to disclose its credentials in order to prove its compliance with an authorization or a trust establishment policy, the credentials in policy-based cryptography are private as they are used as inputs to the decryption and signature algorithms. Indeed, a qualified set of credentials for a policy is required to decrypt a message that is encrypted with respect to the policy, while a qualified set of credential for a policy needs to be used in order to generate a valid signature on a message with respect to the policy.

In this thesis, our goal is threefold:

- to provide formal definitions for policy-based encryption and policy-based signature, then to propose concrete implementations of the two primitives. Because a policy is formalized as a monotone Boolean expression, this challenge turns out to be the one of finding an elegant method to convert the conjunctive and disjunctive logical operators into mathematical operations within the encryption/decryption and signature/verification algorithms.
- to formally define the security models related to our policy-based cryptographic primitives, then to prove the security of the proposed schemes under these models. The idea here is to consider well studied security notions related to encryption and signature schemes and to adapt them in order to cope with the particular features of the policy-based cryptographic primitives.
- to validate the usefulness of the proposed concept of policy-based cryptography. In order to achieve this, we show how our policy-based cryptographic primitives can be used to solve well-established problems in different contexts, while improving the solutions found in the literature.

Structure and Contributions

The rest of this manuscript is organized as follows:

- In Chapter 0, we review the relevant background material related, on one hand, to standard public-key cryptography and the concept of provable security and, on the other hand, to bilinear pairings over elliptic curves and their recent applications in cryptography. The goal of this chapter is naturally not to re-write what can be easily found in several references in the literature. Instead, it aims at introducing the different cryptographic notions and notations that will be used in the next chapters.

  The reader can move on either to Chapter 1 then to Chapter 2, which both tackle policy-based encryption, or to Chapter 3, which independently tackles policy-based signature.

- In Chapter 1, we study the policy-based encryption primitive. The chapter consists of two independent but complementary parts. While we formally study the policy-based encryption primitive in the first part, we illustrate its applications and particular properties in three different contexts in the second part. After formally defining the encryption primitive, we present a new model for semantic security against chosen ciphertext attacks that is adapted to the policy-oriented setting. Then, we present an elegant and relatively efficient pairing-based policy-based encryption scheme, the security of which is proved in the random oracle model. The most intuitive application of policy-based encryption is the enforcement of access control policies. In this context, we present a framework for controlling access to published XML documents using our policy-based encryption primitive. Our solution outclasses the existing ones when dealing with key management. Our second application of policy-based encryption is in the context of privacy policy enforcement which is similar but not exactly the same as the enforcement of standard access control policies. In this context, we show how the policy-based encryption primitive can be used to implement the sticky policy paradigm, while underlying its support for cryptographic workflow. Finally, we show how a privacy-enhanced secure establishment of ad-hoc communities can be achieved using our policy-based encryption primitive.

- In Chapter 2, we address the collusion property that is inherent to the original policy-based encryption primitive presented in Chapter 1. In fact, the latter may suffer in certain contexts from the fact that in addition to a legitimate entity, any collusion of entities that are able to pool their credentials to obtain a qualified set of credentials for the policy according to which a message was encrypted, can decrypt the message. As an alternative, we introduce a variant of the policy-based encryption primitive, called policy-based public-key encryption. After formally defining the new primitive and the corresponding security model, we present a concrete implementation from bilinear pairings and prove its security in the random oracle model. Then, we show how the policy-based public-key encryption can be used to solve the cyclic policy interdependency problem inherent to the standard approaches in the context of trust negotiation.

- In Chapter 3, we study the policy-based signature primitive. Like Chapters 1 and 2, this chapter consists of two independent but complementary parts. After formally defining the policy-based signature primitive, we present a model for credential ambiguity and a model for existential unforgeability against chosen message attacks that are both adapted to the policy-oriented setting. As an application of policy-based signature, we present a novel form of proxy certificates, called proof-carrying proxy certificates.

The research work performed by the author resulted in a number of scientific publications some of which contain the main ideas presented in this manuscript [17, 19, 18, 16, 15], and some others contain contributions that are loosely related to policy-based cryptography [13, 14].

Preliminary Topics

Our primary goal in this thesis is to formalize the concept of policy-based cryptography and to implement provably secure policy-based encryption and signature schemes. It is therefore natural to start our work by providing formal definitions for the policy-based cryptographic primitives and the related security models. In our definitions, we extend the well-studied and commonly recognized formalism of public-key cryptography by taking the particular features of policy-based cryptographic primitives into account. Our implementations of policy-based encryption and signature schemes are based on bilinear pairings over elliptic curves, which are mathematical primitives that proved recently to be extremely useful in cryptography. The primary goal of this chapter is to review the different notions and the notational conventions that will be used in the sequel of this manuscript.

This chapter is organized as follows: in Section 0.1, we briefly recall the formal definitions of four basic cryptographic primitives: symmetric encryption, public-key encryption, digital signature and hash function. The goal of this section is to simply introduce the notational conventions that will be used in our definitions of policy-based cryptographic schemes. In Section 0.2, we provide an overview of the foundations of the concept of provable security including the reductionist proof strategy, the random oracle model and the formal security models related to public-key encryption and digital signature schemes. Particularly, we present the Fujisaki-Okamoto transformations and the replay technique which will be referenced in our reductionist security proofs. Finally, in Section 0.3, we provide an overview of bilinear pairings over elliptic curves and their application in cryptography. We define in particular the related Diffie-Hellman problems whose intractability guarantees the security of our policy-based encryption and signature schemes. Moreover, we present pairing-based public-key encryption schemes which will be referenced later in the security proofs of our policy-based encryption schemes.

0.1 Standard Cryptography

There are four basic cryptographic primitives that are widely used and often combined to secure today's digital communications: symmetric-key encryption, public-key encryption, digital signature and hash function. In this section, we formally define each of these primitives. We refer the reader to [116] for more details about the theoretical and practical aspects of these primitives.

0.1.1 Symmetric-Key Encryption

A symmetric-key encryption scheme allows to encrypt a message using a key in way such that only an entity having access to the same key is able to decrypt the message. This primitive allows to implement the confidentiality security service whose goal is preventing unauthorized entities from having access to a sensitive message. A formal definition of symmetric-key encryption schemes is given below:

Definition 0.1 *A **symmetric encryption scheme**, denoted by* E^{sym}*, is specified by four algorithms:* Setup, KeyGen, Encrypt *and* Decrypt, *which are defined as follows:*

- **Setup**. *On input of a security parameter k, this algorithm generates a set $\mathcal{P}$ of public parameters that specifies the different groups and public functions that will be referenced by subsequent algorithms. Furthermore, it specifies a key space $\mathcal{K}^{sym}$, a message space $\mathcal{M}^{sym}$ and a ciphertext space $\mathcal{C}^{sym}$.*
- **KeyGen**$^{\mathsf{sym}}$. *This algorithm generates a random symmetric key $k_s \in \mathcal{K}^{sym}$.*
- **Encrypt**$^{\mathsf{sym}}$. *On input of message $M \in \mathcal{M}^{sym}$ and symmetric key k_s, this algorithm returns a ciphertext $C \in \mathcal{C}^{sym}$.*
- **Decrypt**$^{\mathsf{sym}}$. *On input of a ciphertext $C \in \mathcal{C}^{sym}$ and a symmetric key k_s, this algorithm returns a message $M \in \mathcal{M}^{sym}$.*

Remark 0.1 *A symmetric-key encryption scheme has to satisfy the standard consistency constraint i.e.* $C = \mathsf{Encrypt}_{k_s}^{sym}(M) \Leftrightarrow \mathsf{Decrypt}_{k_s}^{sym}(C) = M$.

0.1.2 Public-Key Encryption

A public-key encryption scheme allows to encrypt a message in a way such that only an entity having access to the corresponding private key is able to decrypt the message. As for symmetric-key encryption, this primitive allows to implement the confidentiality security service while overcoming the key sharing problem inherent to symmetric-key encryption schemes.

Usually combined with public-key certification mechanisms, they additionally allow to ensure the authenticity of the recipient the message is intended for. A formal definition of public-key encryption schemes is given below:

Definition 0.2 *A **public-key encryption scheme**, denoted by* PKE*, is specified by four algorithms:* Setup, KeyGen, Encrypt *and* Decrypt*, which are defined as follows:*

- **Setup**. *On input of a security parameter k, this algorithm generates a set $\mathcal{P}$ of public parameters that specifies the different groups and public functions that will be referenced by subsequent algorithms. Furthermore, it specifies a key space $\mathcal{K}$, a message space $\mathcal{M}$ and a ciphertext space $\mathcal{C}$.*
- **KeyGen**. *This algorithm generates a pair of keys $(sk, pk) \in \mathcal{K}$, where sk is a private key and pk is the corresponding public key.*
- **Encrypt**. *On input of message $M \in \mathcal{M}$ and public key pk, this algorithm returns a ciphertext $C \in \mathcal{C}$.*
- **Decrypt**. *On input of a ciphertext $C \in \mathcal{C}$ and a private key sk, this algorithm returns a message $M \in \mathcal{M}$ or $\perp$ (for '*error*').*

Remark 0.2 *A public-key encryption scheme has to satisfy the standard consistency constraint i.e.* $C = \mathsf{Encrypt}_{pk}(M) \Leftrightarrow \mathsf{Decrypt}_{sk}(C) = M$.

0.1.3 Digital Signature

A digital signature scheme allows to generate a signature on a message using a private key in way such that the signature is valid only with respect to the public key associated to the private key used to generate it. Often combined with public-key certification mechanisms, this primitive allows to ensure the authenticity of the signer of the message. Furthermore, it allows to implement the non-repudiation security service, whose goal it preventing an entity from denying previous commitments or actions. A formal definition of digital signature schemes is given below:

Definition 0.3 *A **digital signature scheme**, denoted by* DS*, is specified by four algorithms:* Setup, KeyGen, Sign *and* Verify*, which are defined as follows:*

- **Setup**. *On input of a security parameter k, this algorithm generates a set $\mathcal{P}$ of public parameters that specifies the different parameters, groups and public functions that will be referenced by subsequent algorithms. Furthermore, it specifies a key space $\mathcal{K}$, a message space $\mathcal{M}$ and a signature space $\mathcal{S}$.*

- **KeyGen**. *This algorithm generates a pair of keys $(sk, pk) \in \mathcal{K}$, where sk is a private key and pk is the corresponding public key.*
- **Sign**. *On input of message $M \in \mathcal{M}$ and private key sk, this algorithm returns a signature $\sigma = \mathsf{Sign}_{sk}(M) \in \mathcal{S}$.*
- **Verify**. *On input of a message M, signature σ and a public key pk, this algorithm returns either $\top$ (for 'valid') or $\perp$ (for 'invalid').*

Remark 0.3 *A digital signature scheme has to satisfy the standard consistency constraint i.e.* $\sigma = \mathsf{Sign}_{sk}(M) \Leftrightarrow \mathsf{Verify}_{pk}(M, \sigma) = \top$.

0.1.4 Hash Function

A hash function is a mathematical primitive that, given a variable-length input string, called pre-image, converts it to a fixed-length output string, called hash value. Informally, it allows to fingerprint a message i.e. produce a value that indicates whether a candidate pre-image is likely to be the same as the real pre-image. This primitive allows to implement the integrity security service, which prevents from an accidental or malicious manipulation (insertion, deletion, substitution) of information during its transmission. A formal definition of hash functions is given below:

Definition 0.4 *A **hash function** $H : \{0,1\}^* \rightarrow \{0,1\}^n$ is an efficiently computable algorithm that maps an input x of arbitrary finite bit-length, to an output $H(x)$ of fixed bit-length n.*

*A hash function is a **collision resistant hash function** (strong one-way hash function) if it satisfies the following properties:*

1. *Preimage resistance: for all $y \in \{0,1\}^n$, it is computationally infeasible to find an element $x \in \{0,1\}^*$ such that $y = H(x)$*
2. *2^{nd}-Preimage resistance: for all $x \in \{0,1\}^*$, it is computationally infeasible to find $x' \neq x$ such that $H(x') = H(x)$*
3. *Collision resistance: it is computationally infeasible to find $x \neq x' \in \{0,1\}^*$ such that $H(x) = H(x')$*

*A hash function that satisfies the two first properties but do not satisfy the third one is called a **one-way hash function** (weak one-way hash function).*

Remark 0.4 *The above definition of hash functions although might seem informal suffices for this thesis. We refer to [132] for a comprehensive discussion on hash functions and the related security notions.*

Remark 0.5 *A hash function is usually designed to act as a compression function. It can also be designed to map elements of one group to elements of another group. In practice, however, mapping between groups is difficult and may require mapping to some intermediate set and using some deterministic encoding operations to map to and from the groups. Such a construction was shown for a hash function that maps the set of binary strings* $\{0,1\}^*$ *into a group* $\mathbb{G}_1^*$ *of points over an elliptic curve in [38, 39]. Further details about elliptic curve algebra are given in Section 0.3.*

Now that we have formally defined the four basic cryptographic primitives, we discuss in the following section the related formal security notions.

0.2 Provable Security

The fact that a cryptographic scheme withstood cryptanalytic attacks for a long time has been considered as a kind of security validation for many years. Because several schemes have taken a long time before being totally broken, the lack of cryptanalytic attacks or the provision of intuitive or heuristic security arguments are not considered as security validations anymore. A more convincing approach, first introduced by Goldwasser and Micali in [85], consists in proving the security of cryptographic schemes in the context of complexity theory. In this approach, a cryptographic scheme is associated to a well-studied hard problem, called the underlying problem, and the security proof, also referred to as reductionist security proof, consists in showing that if one can break the cryptographic scheme, then one can efficiently solve the underlying problem. This approach is referred to as the provable security paradigm.

In the following section, we first provide a brief overview of the concept of reductionist security proofs and the random oracle model under which we prove the security of our policy-based cryptographic schemes. Then, we define the security notions related to standard encryption and signature schemes and the related proof techniques.

0.2.1 Reductionist Security Proofs

Before describing the principles of the provable security paradigm, we first provide the formal definitions of the notions of *negligible function* and *polynomial time algorithm*.

Definition 0.5 *A real function* f *is a* ***negligible function*** *if for every integer* $c \geq 0$ *there exists an integer* $k_c > 0$ *such that for all* $k > k_c$, $f(k) < \frac{1}{k^c}$.

Definition 0.6 *Let* f *and g be functions of parameter k. We write* $f(k) = O(g(k))$ *if there exists a positive constant c and a positive integer* k_0 *such that, for all* $k \geq k_0$, $0 \leq f(k) \leq c.g(k)$.

Definition 0.7 *A **polynomial time algorithm** is an algorithm whose worst-case running time function is of the form $O(k^c)$, where k is the input size and c is a constant.*

The process that is followed to prove the security of a given scheme under the provable security paradigm consists of the following three stages:

1. **Cryptographic scheme**: provide a formal definition of the considered scheme i.e. provide a detailed description of the different algorithms that compose the scheme and the associated parameters and security requirements.

2. **Security model**: state the security model under which the considered scheme is required to be secure. This statement results in the definition of intractability for an adversarial goal under a specific attack scenario. Concretely, the security model is defined in terms of a game played between the adversary, who is modeled as a polynomial time algorithm, and the challenger, who controls a set of oracles simulating the different components of the considered scheme. The game consists of a set of query-response interactions that allow the adversary to get the information it wants to acquire in order to perform its attack. At the end of the interactions, if the adversary wins the game, then it succeeds in achieving its adversarial goal.

3. **Reductionist proof**: provide a reductionist security proof that shows that the adversary can be transformed to an algorithm that solves a problem, called the underlying problem, known to be computationally hard to solve. It does so by simulating the adversary's attack environment. The simulation should be performed in way such that the adversary cannot distinguish, with a non-negligible probability, the simulated environment from a real world environment. The success probability of solving the hard computational problem can be related to that of the adversary. The very assumption that the computational problem is hard (in the sense of there being no polynomial time algorithm which can solve it) shows that no adversaries with non-negligible probability of success can exist.

The three stages are summarized in Figure 4.

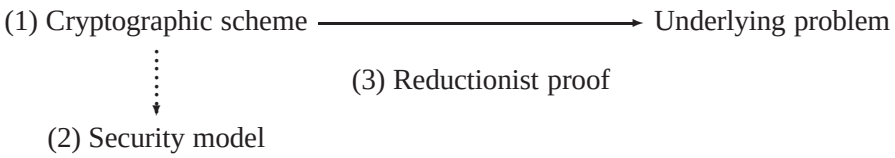


Figure 4: Framework for reductionist security proofs

0.2.2 The Random Oracle Model

The reductionist security proofs proposed in this thesis require the hash functions to be instantiated by ideal hash functions, also referred to as random oracles. Concretely, as explained in [3], if the challenger controls an oracle simulating some hash function $H : \mathcal{A} \rightarrow \mathcal{B}$ used by the considered cryptographic scheme, then the outputs of the oracle, which are returned to the adversary, are computationally indistinguishable from a random output. This is performed according to a heuristic that consists in replacing H with a member of the family of all truly random functions from $\mathcal{A}$ to $\mathcal{B}$, chosen uniformly at random. Since all the adversary's queries to H are answered by selecting an output at random, H is now effectively selected uniformly at random from the family of all functions. When queried on the same input, the oracle must be defined to produce the same output, since the hash function will behave this way in the real world. Hence, in the random oracle model, no adversary can make use of the underlying structure of the real hash function.

First formulated by Bellare and Rogaway in [31], the random oracle model is considered as the most widely accepted assumption in the provable security paradigm in the last ten years. Provided that the adversary has no insight into the hash function, using this black box idealized approach to model hash functions clearly captures the security essence of the overall cryptographic scheme. Moreover, the abstraction allows designing cryptographic schemes whose efficiency cannot be achieved if the security proofs are performed without any ideal assumption. As presented in [31], the random oracle model provides thus a bridge between cryptographic theory and cryptographic practice. Critics argue that no single deterministic polynomial time function can provide a good implementation of random oracles in the real world. In other words, they argue that the random oracle methodology is flawed. More recently, another direction has been taken to prove the security of efficient schemes in the so called "standard model" by using stronger computational assumptions. As we only address the theoretical security of our schemes in this thesis, we do not elaborate more on the details of the new approach. We refer the reader to [48] for a critical look at the relationship between the security of cryptographic schemes in the random oracle model and the security of the schemes that result from implementing the random oracle by real hash functions.

0.2.3 Security Notions for Public-Key Encryption Schemes

While we refer the reader to [29] for a thorough study of the formal security models related to public-key encryption schemes (PKE), we review those that will be referenced in this manuscript. Given a PKE scheme, the most intuitive goal of an adversary is to decrypt a message encrypted using some public key without having access to the corresponding private key. A more advanced goal considered in the literature is to distinguish between two messages, chosen by the adversary itself, which one has been encrypted, with a probability significantly greater than one half. The first adversarial goal is referred to as one-way encryption (OW), whereas the second is referred to as indistinguishability (IND).

In addition to the adversarial goal, a security model specifies the attack scenario under which the security of the considered public-key encryption scheme have to be proved. Depending on the information that is given by the challenger to the adversary, two attack scenarios are considered in the literature. In the first scenario, the adversary does not have access to any specific information apart from the fact that it is able to encrypt any plaintext of its choice. This is referred to as the chosen plaintext attacks scenario (CPA). In the second scenario, in addition to being able to encrypt any plaintext of its choice, the adversary is allowed to ask the challenger to decrypt any ciphertext of its choice, with the natural exception of the ciphertext it will be challenged on. This second scenario is referred to as the chosen ciphertext attacks scenario (CCA).

According to the considered adversarial goals and attack scenarios, the most basic security notion under which one would require a public-key encryption scheme to be secure is thus one-way encryption against chosen plaintext attacks (OW-CPA), where with just public data, the adversary should not retrieve the whole plaintext corresponding to a given ciphertext. Whereas the strongest security notion is indistinguishability against chosen ciphertext attacks (IND-CCA), where the adversary has access to a decryption oracle controlled by the challenger. A formal description of each of the two models is given below.

One-way encryption against chosen plaintext attacks is defined in terms of an interactive game, denoted by OW-CPA, played between a challenger and an adversary.

Definition 0.8 *The* **OW-CPA *game*** *consists of three stages:* Setup, Challenge *and* Guess, *which we describe below:*

- **Setup**. *On input of a security parameter k, the challenger first runs algorithm* Setup *to obtain the system public parameters $\mathcal{P}$. Then, the challenger runs algorithm* KeyGen *to obtain a pair of keys (pk_{ch}, sk_{ch}). The challenger gives to the adversary the public parameters $\mathcal{P}$ as well as the public key pk_{ch}.*
- **Challenge**. *The challenger picks at random a message $M \in \mathcal{M}$, runs algorithm* Encrypt *on input of the tuple (M, pk_{ch}), and finally returns the resulting ciphertext C_{ch} to the adversary.*
- **Guess**. *After running some computations, the adversary outputs a guess M', and wins the game if $M = M'$.*

Definition 0.9 *The advantage of an adversary $\mathcal{A}$ in the* OW-CPA *game is defined to be the quantity $Adv_{\mathcal{A}} = Pr[M = M']$. A public-key encryption scheme is* OW-CPA *secure if no probabilistic polynomial time adversary has a non-negligible advantage in the* OW-CPA *game.*

Remark 0.6 *In Definition 0.9, the probability $Pr[M = M']$ is measured over the random bits used by the challenger and the adversary. This remark holds for the different security models considered henceforth.*

Indistinguishability against chosen ciphertext attacks is defined in terms of an interactive game, denoted by IND-CCA, played between a challenger and an adversary.

Definition 0.10 *The* **IND-CCA *game*** *consists of five stages:* Setup, Phase-1, Challenge, Phase-2 *and* Guess, *which we describe below:*

- **Setup**. *This stage is similar to the* Setup *stage described in Definition 0.8.*
- **Phase-1**. *The adversary performs a polynomial number of oracle queries.*
- **Challenge**. *Once the adversary decides that* Phase-1 *is over, it gives the challenger two equal length messages M_0, M_1 on which it wishes to be challenged. The challenger picks at random $b \in \{0,1\}$, then runs algorithm* Encrypt *on input of the tuple (M_b, pk_{ch}), and finally returns the resulting ciphertext C_{ch} to the adversary.*
- **Phase-2**. *The adversary performs again a polynomial number of oracle queries.*
- **Guess**. *The adversary outputs a guess b', and wins the game if $b = b'$.*

During Phase-1 *and* Phase-2, *the adversary may perform queries to a decryption oracle, denoted by* Decrypt-O, *controlled by the challenger. While the oracle is executed by the challenger, its input is specified by the adversary. The oracle is defined as follows: on input of a ciphertext $C \in \mathcal{C}$, run algorithm* Decrypt *on input of the tuple (C, sk_{ch}) and return the resulting output.*

Definition 0.11 *The advantage of an adversary $\mathcal{A}$ in the* IND-CCA *game is defined to be the quantity $Adv_{\mathcal{A}} = |Pr[b = b'] - \frac{1}{2}|$. A public-key encryption scheme is* IND-CCA *secure if no probabilistic polynomial time adversary has a non-negligible advantage in the* IND-CCA *game.*

Definition 0.12 *In Definition 0.10, the decryption oracle queries performed by the adversary in* Phase-2 *can be forbidden as soon as the challenge ciphertext is known, the attack is thus said non-adaptive since these oracle queries cannot depend on the challenge ciphertext, while they depend on previous answers. On the opposite, access can be unlimited and attacks are thus called adaptive attacks (with respect to the challenge ciphertext). The latter scenario is definitely the strongest one, and is by the way the attack scenario considered in this thesis.*

IND-CCA $\Longrightarrow$ IND-CPA $\Longrightarrow$ OW-CPA

Figure 5: Relations between the Security Notions for PKE Schemes

Indistinguishability against chosen plaintext attacks is defined in terms of an interactive game, denoted by IND-CPA, played between a challenger and an adversary. The formal definition of the IND-CPA game is similar to the one of the IND-CCA game apart from the fact that the

adversary does not have access to the decryption oracle during Phase-1 and Phase-2. Figure 5 shows the relations between the OW-CPA, IND-CPA and IND-CCA games. Intuitively, a PKE scheme that is IND-CPA is necessarily OW-CPA secure, and a PKE scheme that is IND-CCA is necessarily IND-CPA secure.

In Section 0.2.4, we describe two generic transformations that allow to strengthen the security of PKE schemes. The transformations, proposed by Fujisaki and Okamoto, use E^{sym} schemes that are required to be secure in the so called *find-guess* security model, which is defined in terms of an interactive game, denoted by FG, played between a challenger and an adversary.

Definition 0.13 *The* **FG *game*** *consists of three stages:* Setup*,* Find *and* Guess*, which are described below:*

- **Setup**. *On input of a security parameter k, the challenger first runs algorithm* Setup *to obtain the system public parameters $\mathcal{P}$. Then, the challenger runs algorithm* KeyGen *to obtain a symmetric key k_{sch}. The challenger gives to the adversary the public parameters $\mathcal{P}$, while keeping secret the key k_{sch}.*
- **Find**. *This is a challenger stage during which the adversary gives to the challenger two equal length messages M_0, M_1 on which it wishes to be challenged. The challenger picks at random $b \in \{0,1\}$, then runs algorithm* Encrypt^{sym} *on input of the tuple (M_b, k_{sch}), and finally returns the resulting ciphertext C_{ch} to the adversary.*
- **Guess**. *The adversary outputs a guess b', and wins the game if $b = b'$.*

Definition 0.14 *The advantage of an adversary $\mathcal{A}$ in the* FG *game is defined to be the quantity $Adv_{\mathcal{A}} = |Pr[b = b'] - \frac{1}{2}|$. A symmetric encryption scheme is* FG *secure if no probabilistic polynomial time adversary has a non-negligible advantage in the* FG *game.*

Remark 0.7 *In Definition 0.13, the* FG *security model corresponds to indistinguishability for which the adversary is not allowed to have access to encryption oracles. We refer the reader to [28] for further details about the security notions related to E^{sym} schemes.*

0.2.4 The Fujisaki-Okamoto Transformations

In this section, we describe two generic transformations proposed by Fujisaki and Okamoto in [78] and [79] respectively. While the first transformation allows to convert any IND-CPA secure PKE scheme into an IND-CCA one, the second transformation can be considered more interesting as it allows to transform any OW-CPA secure PKE scheme (scheme with weaker security requirement) into an IND-CCA one. The second transformation is used in the security proof of our policy-based encryption scheme described in Chapter 1, whereas the first one is used in the security proof of our collusion-free policy-based encryption scheme proposed in Chapter 2. The two Fujisaki-Okamoto transformations are naturally performed in the context of the random oracle model.

Remark 0.8 *The Fujisaki-Okamoto transformation techniques are called* hybridization techniques *because the obtained 'hybrid'* PKE *schemes result from the combination of the 'basic'* PKE *scheme and a specific* E^{sym} *scheme.*

In order to illustrate the two Fujisaki-Okamoto transformations, we need an alternative definition for PKE schemes that is slightly different from Definition 0.2.

Definition 0.15 *A (basic) public-key encryption scheme, denoted by* PKE^{basic}*, is specified by four algorithms:* Setup, KeyGen, $\mathsf{Encrypt}^{basic}$ *and* $\mathsf{Decrypt}^{basic}$*, which are defined as follows:*

- **Setup**. *On input of a security parameter k, this algorithm generates a set* $\mathcal{P}$ *of public parameters that specifies the different parameters, groups and public functions that will be referenced by subsequent algorithms. Furthermore, it specifies a key space* $\mathcal{K}$*, a message space* $\mathcal{M}^{basic}$*, a finite coin space* $\mathcal{R}^{basic}$*, and a ciphertext space* $\mathcal{C}^{basic}$*.*
- **KeyGen**. *This algorithm is similar to the* KeyGen *algorithm described in Definition 0.2.*
- **Encrypt**$^{\mathsf{basic}}$. *On input of message* $M \in \mathcal{M}^{basic}$ *and public key pk, this algorithm picks at random a coin* $r \in \mathcal{R}^{basic}$ *and returns a ciphertext* $C = \mathsf{Encrypt}_{pk}^{basic}(M, r) \in \mathcal{C}^{basic}$*.*
- **Decrypt**$^{\mathsf{basic}}$. *On input of a ciphertext* $C \in \mathcal{C}^{basic}$ *and a private key sk, this algorithm returns a message* $M = \mathsf{Decrypt}_{sk}^{basic}(C) \in \mathcal{M}^{basic}$*.*

Remark 0.9 *As opposed to Definition 0.2, the encryption algorithm* $\mathsf{Encrypt}^{basic}$ *described in Definition 0.15 involves a random value r, while the decryption algorithm* $\mathsf{Decrypt}^{basic}$ *is deterministic as it never outputs an error message* $\perp$*.*

In [78], Fujisaki-Okamoto propose their first transformation, which we denote by FO_I. Their transformation allows to convert a $\mathsf{PKE}^{\mathsf{basic}}$ scheme that is IND-CPA secure into a 'hybrid' PKE scheme (denoted by PKE^{hy_I}) that is IND-CCA secure. We refer to [78] for the details of the corresponding reductionist security proof.

Definition 0.16 *Let* PKE^{hy_I} *be the (hybrid) public-key encryption scheme obtained from an* IND-CPA *secure* PKE^{basic} *scheme by applying the Fujisaki-Okamoto transformation* FO_I *to* PKE^{basic} *and a symmetric encryption scheme* E^{sym}*. Then,* PKE^{hy_I} *consists of four algorithms:* Setup, KeyGen, $\mathsf{Encrypt}^{hy_I}$ *and* $\mathsf{Decrypt}^{hy_I}$*, which are defined as follows:*

- **Setup**. *This algorithm is similar to the one described in Definition 0.15. Here, the message space is denoted by* $\mathcal{M}^{hy_I} = \mathcal{M}^{sym}$ *while the coin space is defined as* $\mathcal{R}^{hy_I} = \mathcal{M}^{basic}$*. Furthermore, the system parameters include two hash functions:* $H_1 : \{0,1\}^* \rightarrow \mathcal{R}^{basic}$ *and* $H_3 : \mathcal{R}^{hy_I} \rightarrow \mathcal{K}^{sym}$*.*
- **KeyGen**. *This algorithm is similar to the* KeyGen *algorithm described in Definition 0.2.*

- **Encrypt**^{hy_I}. *On input of message* $M \in \mathcal{M}^{hy_I}$ *and public key pk, this algorithm picks at random a coin* $t \in \mathcal{R}^{hy_I}$ *and returns a ciphertext* $C = \mathsf{Encrypt}_{pk}^{hy_I}(M,t) \in \mathcal{C}^{hy_I}$ *that is obtained as follows:*

$$C = (\mathsf{Encrypt}_{pk}^{basic}(t,r), \mathsf{Encrypt}_{H_3(t)}^{sym}(M)), \text{ where } r = H_1(M||t)$$

- **Decrypt**^{hy_I}. *On input of a ciphertext* $C = (C_1, C_2) \in \mathcal{C}^{hy_I}$ *and a private key sk, this algorithm returns either a message* $M \in \mathcal{M}^{hy_I}$ *or* $\perp$ *(for '*error*'). This is performed as follows:*

  1. *Compute* $t = \mathsf{Decrypt}_{sk}^{basic}(C_1)$, *then compute* $M = \mathsf{Decrypt}_{H_3(t)}^{sym}(C_2)$ *and* $r = H_1(M||t)$
  2. *If* $C_1 = \mathsf{Encrypt}_{pk}^{basic}(t,r)$, *then output M. Otherwise, return* $\perp$.

The first Fujisaki-Okamoto transformation is summarized in Figure 6.

Figure 6: FO_I Transformation

In [79], Fujisaki-Okamoto propose their second transformation, which we denote by FO_{II}. This transformation allows to convert a $\mathsf{PKE}^{\text{basic}}$ scheme that is OW-CPA secure into a 'hybrid' PKE scheme (denoted by $\mathsf{PKE}^{hy_{II}}$) that is IND-CCA secure. We refer to [79] for the details of the corresponding reductionist security proof.

Definition 0.17 *Let* $\mathsf{PKE}^{hy_{II}}$ *be the (hybrid) public-key encryption scheme obtained from a* OW-CPA *secure* PKE^{basic} *scheme by applying the Fujisaki-Okamoto transformation* FO_{II} *to* PKE^{basic} *and a symmetric encryption scheme* E^{sym}. *Then,* $\mathsf{PKE}^{hy_{II}}$ *consists of four algorithms:* Setup, KeyGen, Encrypt$^{hy_{II}}$ *and* Decrypt$^{hy_{II}}$ *which are defined as follows:*

- **Setup**. *This algorithm is almost similar to the one described in Definition 0.15. Let* $\mathcal{M}^{basic} = \{0,1\}^n$ *(for some integer* $n \in \mathbb{N}^*$*),* $\mathcal{M}^{hy_{II}} = \{0,1\}^{n-n_0}$ *(for some positive integer* $n_0 \ll n$*) and* $\mathcal{R}^{hy_{II}} = \{0,1\}^{n_0}$. *In addition, specify a hash function:* $H_1 : \{0,1\}^* \rightarrow \mathcal{R}^{basic}$.
- **KeyGen**. *This algorithm is similar to the* KeyGen *algorithm described in Definition 0.2.*
- **Encrypt**$^{hy_{II}}$. *On input of message* $M \in \mathcal{M}^{hy_{II}}$ *and public key pk, this algorithm picks at random a coin* $t \in \mathcal{R}^{hy_{II}}$ *and returns a ciphertext* $C = \mathsf{Encrypt}_{pk}^{hy_{II}}(M,t) \in \mathcal{C}^{hy_{II}}$ *that is obtained as follows:*

$$C = \mathsf{Encrypt}_{pk}^{basic}(M||t,r), \text{ where } r = H_1(M||t)$$

- **Decrypt**$^{hy_{II}}$. *On input of a ciphertext* $C \in \mathcal{C}^{hy_{II}}$ *and a private key sk, this algorithm returns either a message* $M \in \mathcal{M}^{hy_{II}}$ *or* $\perp$ *(for '*error*'). This is performed as follows:*

  1. *Compute* $M||t = \mathsf{Decrypt}^{basic}_{sk}(C)$ *and* $r = H_1(M||t)$
  2. *If* $C = \mathsf{Encrypt}^{basic}_{pk}(M||t, r)$*, then output* M*. Otherwise, return* $\perp$*.*

The second Fujisaki-Okamoto transformation is summarized in Figure 7.

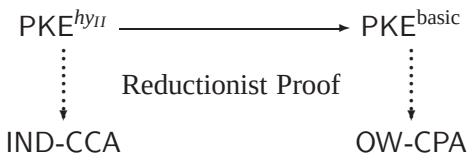


Figure 7: FO_{II} Transformation

Concrete examples of the application of the two Fujisaki-Okamoto transformations are given in Section 0.3.5 in the context of pairing-based encryption schemes.

0.2.5 Security Notions for Digital Signature Schemes

The standard acceptable security model for digital signature schemes is existential unforgeability against chosen message attacks. In this security model, the goal of the adversary is to generate a signature on a message in a way such that the signature is valid with respect to a public key defined by the challenger. Here, the adversary is assumed to be unable to have access to the corresponding private key. The adversary is allowed to choose the message according to which it wants to be challenged on and it naturally does not have access to the challenger's private key. Besides, the attack scenario is such that the adversary can ask the challenger to generate a signature on any message of its choice, in an adaptive way i.e. it can adapt its queries according to previous challenger's answers. We refer the reader to [86] for a detailed analysis of the security models associated to digital signature schemes.

Existential unforgeability against chosen message attacks is defined in terms of an interactive game, denoted by EUF-CMA, played between a challenger and an adversary.

Definition 0.18 *The* **EUF-CMA *game*** *consists of three stages:* Setup, Probing *and* Forge, *which are defined as follows:*

- **Setup**. *On input of a security parameter k, the challenger first runs algorithm* Setup *to obtain the system public parameters* $\mathcal{P}$*. Then, the challenger runs algorithm* KeyGen *to obtain a pair of keys* pk_f, sk_f*. The challenger gives to the adversary the public parameters* $\mathcal{P}$ *as well as the public key* pk_f*.*

- **Queries**. *The adversary performs a polynomial number of adaptive oracle queries.*
- ***Forge***. *Once the adversary decides that the Queries stage is over, it outputs a message M_f and a signature σ_f, and wins the game if $\mathsf{Verify}_{pk_f}(M_f, \sigma_f) = \top$.*

During the Queries *stage, the adversary may perform queries to a signature oracle, denoted by* Sign-O*, controlled by the challenger. While the oracle is executed by the challenger, its input is specified by the adversary. The oracle is defined as follows: given a message $M \in \mathcal{M}$, run algorithm* Sign *on input of the tuple (M, sk_{ch}) and return the resulting output. The adversary is obviously not allowed to query a signature on the message M_f.*

Definition 0.19 *The advantage of an adversary $\mathcal{A}$ in the* EUF-CMA *game is defined to be the quantity $Adv_{\mathcal{A}} = Pr[\mathcal{A}\ wins]$. A digital signature scheme is* EUF-CMA *secure if no probabilistic polynomial time adversary has a non-negligible advantage in the* EUF-CMA *game.*

In the following section, we describe a reductionist proof technique, called the oracle replay technique, which was defined by Pointcheval and Stern in [127]. This technique is used later in Chapter 3 to prove the security of our policy-based signature scheme.

0.2.6 The Oracle Replay Technique

The intuition behind the oracle replay technique is as follows: by a polynomial replay of the attack with the same random tape and a different oracle, we obtain two signatures of a specific form which open a way to solve the underlying hard problem. To illustrate this technique, consider the Schnorr signature scheme [135] described below.

Definition 0.20 *The **Schnorr digital signature scheme** is specified by four algorithms:* Setup*,* KeyGen*,* Sign *and* Verify*, which are defined as follows:*

- **Setup**. *On input of a security parameter k, this algorithm defines a cyclic group $(\mathbb{G}, +)$ of prime order q such that $2^{k-1} \leq q < 2^k$, specifies at random a generator $P \in \mathbb{G}$, and defines a hash function $H : \{0,1\}^* \rightarrow \mathbb{Z}_q$. The message space is $\mathcal{M} = \{0,1\}^*$.*
- **KeyGen**. *This algorithm picks at random a private key $sk \in \mathbb{Z}_q^*$ and computes the corresponding public key $pk = -sk \cdot P$.*
- **Sign**. *On input of a message M, this algorithm picks at random $t \in \mathbb{Z}_q^*$, computes $r = t \cdot P$, $h = H(M||r)$ and $s = t + sk \cdot h \bmod q$, and returns the tuple (r, h, s) as the signature on M.*
- **Verify**. *On input of a message M and a tuple (r, h, s), if $h = H(M||r)$ and $r = s \cdot P + h \cdot pk$, then output $\top$. Otherwise, output $\bot$.*

Thanks to the oracle replay technique, the Schnorr signature scheme described above was proved to be EUF-CMA secure, under the assumption that the Discrete Logarithm Problem (DLP) is hard [126]. In the EUF-CMA game that occurs between the challenger and the adversary, the former simulates the hash function used by the signature algorithm by controlling a random oracle H. The signature on a message M is a tuple $(\sigma_1 = r, h = H(M||\sigma_1), \sigma_2)$. Note that the quantity h depends on the message M and the first element of the signature σ_1. The idea of the replay technique is that if the probability of forgery is high enough, then with good probability, the adversary is able to answer to many distinct outputs from the H random oracle, on the same input (M, σ_1). Therefore, if the challenger and the adversary play twice the EUF-CMA game and if it is assumed that the adversary is able to break the signature scheme, then this can lead to two distinct signatures (σ_1, h, σ_2) and $(\sigma'_1, h', \sigma'_2)$ forged by the adversary on the same message m and such that $\sigma_1 = \sigma'_1$ and $h \neq h'$. Thereafter, the discrete algorithm of the public key can be retrieved as follows:

$$\left.\begin{array}{l} r = s \cdot g + h \cdot y \\ r = s' \cdot g + h' \cdot y \end{array}\right\} \Rightarrow (s - s') \cdot g = (h' - h) \cdot y \Rightarrow log_g(y) = (s - s').(h' - h)^{-1} mod q$$

Before moving to the section on bilinear pairings over elliptic curves, we recall a probabilistic tool, called the splitting lemma, which was defined in [127] and which is used later in the security proof of our policy-based signature scheme. Informally, this lemma shows that when a subset A is "large" in a product space $X \times Y$, it has many "large" sections.

Lemma 0.1 (Splitting Lemma) *Let $A \subset X \times Y$ such that $Pr[(x,y) \in A] \geq \varepsilon$, for some $\varepsilon > 0$. Given $\alpha < \varepsilon$, define the set $B = \{(x,y) \in X \times Y | Pr_{y' \in Y}[(x,y') \in A] \geq \varepsilon - \alpha\}$. The following statements hold:*

1. $Pr[(x,y) \in B | (x,y) \in X \times Y] \geq \alpha$
2. $\forall\ (x,y) \in B,\ Pr_{y' \in Y}[(x,y') \in A] \geq \varepsilon - \alpha$
3. $Pr[(x,y) \in B | (x,y) \in A] \geq \frac{\alpha}{\varepsilon}$

0.3 Bilinear Pairings

Elliptic curves have been studied by mathematicians for over a hundred years [87]. In 1985, they were introduced to cryptography independently by Koblitz [101] and Miller [120], who proposed to use the associated groups in the design of public-key cryptographic systems. Since then a large number of publications addressed the performance and security of elliptic curve-based cryptographic schemes. In the late nineties, elliptic curve systems started receiving commercial acceptance when accredited standards organizations specified curve-based cryptographic schemes that have been included by private companies in their security products.

The intuition behind pairings is the construction of a mapping between two well-defined groups. The mapping allows the design of new cryptographic schemes whose security is based on the reduction of one problem in the first group to a different and usually easier problem in the second group. Today, the known implementations of such mappings, namely the Weil and the Tate pairings, use groups over elliptic curves, while the most popular pairing-based cryptographic scheme is the identity-based encryption scheme proposed by Boneh and Franklin in [38].

In the following section, we provide a brief introduction to the notions of groups and fields, which are intensively used in the design of cryptographic schemes.

0.3.1 Abstract Algebra

A group is defined as follows:

Definition 0.21 *A **group** is a set of elements $\mathbb{G}$ coupled with a binary operation $\circ$ that satisfy the following properties:*

1. *Closure: for all $x,y \in \mathbb{G}$, $x \circ y \in \mathbb{G}$*
2. *Associativity: the operation $\circ$ is associative i.e. $(x \circ y) \circ z = x \circ (y \circ z)$*
3. *Identity: there exists $I \in \mathbb{G}$, called identity element, such that $I \circ x = x \circ I = x$ for all $x \in \mathbb{G}$.*
4. *Inverse: every element $x \in \mathbb{G}$ has an inverse $x' \in \mathbb{G}$ such that $x \circ x' = I$*

Remark 0.10 *In this thesis, we use the following notations: the operation $\circ$ is denoted by '$+$' for additive groups and by '$*$' for multiplicative groups. The identity element of an additive group $(\mathbb{G},+)$ is denoted by $0_{\mathbb{G}}$ and the identity element of a multiplicative group $(\mathbb{G},*)$ is denoted by $1_{\mathbb{G}}$. Finally, the scalar multiplication is denoted by '$\cdot$' and $\mathbb{G} \setminus \{I\}$ is denoted by $\mathbb{G}^*$.*

Definition 0.22 *The number of elements in a group $\mathbb{G}$, denoted by $|\mathbb{G}|$, is called the **order** of $\mathbb{G}$. A group $\mathbb{G}$ is a finite group if $|\mathbb{G}|$ is finite.*

Definition 0.23 *A group whose elements commute (for all $x,y \in \mathbb{G}$, $x \circ y = y \circ x$) is called an **Abelian group.***

Definition 0.24 *A group $\mathbb{G}$ is a **cyclic group** if there is an element $g \in \mathbb{G}$ such that for each $x \in \mathbb{G}$ there is an integer $e \in \mathbb{Z}$ such that $x = g^e$. Such an element g is called a generator of $\mathbb{G}$.*

Remark 0.11 *Any cyclic group is Abelian and any prime-order group is cyclic.*

Definition 0.25 *A **subgroup** is a subset of a group G that satisfies the four group properties given in Definition 0.21.*

The main groups used in this thesis are $\mathbb{Z}_q$, $\mathbb{G}_1$ and $\mathbb{G}_2$. The group $\mathbb{Z}_q$ denotes the set of integers under addition modulo the prime number q. The two other groups are an additive group $\mathbb{G}_1$ and a related multiplicative group $\mathbb{G}_2$. Both are cyclic groups of large prime order related to elliptic curves over finite fields.

A field is defined as follows:

Definition 0.26 *A **field** $\mathbb{F}$ is a set of elements with two binary operations $+$ and $*$ that together satisfy the following properties:*

1. *$(\mathbb{F},+)$ is an Abelian group with the (additive) identity element denoted by $0_{\mathbb{F}}$*
2. *$(\mathbb{F}\setminus\{0\},*)$ is an Abelian group with the (multiplicative) identity element denoted by $1_{\mathbb{F}}$*
3. *Distributivity: for all $x,y,z\in\mathbb{F}$, $(x+y)*z = x*z+y*z$*

Definition 0.27 *The number of elements in a field $\mathbb{F}$, denoted by $|\mathbb{F}|$, is called the **order** of $\mathbb{F}$. A field $\mathbb{F}$ is finite if $|\mathbb{F}|$ is finite. A field of order q is denoted by $\mathbb{F}_q$.*

Remark 0.12 *The efficient implementation of finite field arithmetic is an important prerequisite in elliptic curve systems because curve operations are performed using arithmetic operations in the underlying field.*

0.3.2 Elliptic Curves

A general definition of elliptic curves is given below:

Definition 0.28 *An **elliptic curve** over a finite field $\mathbb{F}_q$, denoted by $\mathcal{E}(\mathbb{F}_q)$, is a set of points $(x,y)\in\mathbb{F}_q\times\mathbb{F}_q$ together with a special point O, called the point at infinity. The set of points satisfies the equation, called **affine Weierstrass equation**, defined below:*

$$y^2 + a_1*x*y + a_3*y = x^3 + a_2*x^2 + a_4*x + a_6 \tag{1}$$

where $a_i\in\mathbb{F}_q$ for $i=1,2,3,4,6$ and $\Delta\neq 0_{\mathbb{F}_q}$, where Δ, called the discriminant of $\mathcal{E}(\mathbb{F}_q)$, is defined as follows:

$$\begin{aligned}
\Delta &= -d_2^2*d_8 - 8\cdot d_4^3 - 27\cdot d_6^2 + 9\cdot d_2*d_4*d_6\\
d_2 &= a_1^2 + 4\cdot a_2\ ,\ d_4 = 2\cdot a_4 + a_1*a_3\\
d_6 &= a_3^2 + 4\cdot a_6\ ,\ d_8 = a_1^2*a_6 + 4\cdot a_2*a_6 - a_1*a_3*a_4 + a_2*a_3^2 - a_4^2
\end{aligned}$$

In Definition 0.28, the condition $\Delta \neq 0_{\mathbb{F}_q}$ ensures that the elliptic curve is "smooth", that is, there are no points at which the curve has two or more distinct tangent lines.

Remark 0.13 *The Weierstrass equation defined over the field $\mathbb{F}_q$ (Equation 1) can be considerably simplified by applying admissible changes of variables. For instance, in the case where the characteristic of the field $\mathbb{F}_q$ is neither 2 nor 3, the admissible change of variables:*

$$(x,y) \rightarrow (\frac{x-3\cdot a_1^2-12\cdot a_2}{36}, \frac{y-3\cdot a_1 * x}{216} - \frac{a_1^3+4\cdot a_1 * a_2-12\cdot a_3}{24})$$

leads to the elliptic curve equation: $y^2 = x^3 + a*x+b$ ($a,b \in \mathbb{F}_q$), for which the discriminant is $\Delta = -16\cdot(4\cdot a^3+27\cdot b^2)$. We refer to Chapter 3 of [87] for more details about simplified Weierstrass equations.

The set of points $(x,y) \in \mathbb{F}_q \times \mathbb{F}_q$ satisfying the equation that defines the elliptic curve $\text{E}(\mathbb{F}_q)$, together with the point O and some (additive) group operation $+$, can be used to form a group, denoted by $(\text{E}(\mathbb{F}_q),+)$, known as an elliptic curve group.

Remark 0.14 *Since the Weierstrass equation 1 has at most two solutions for each $x \in \mathbb{F}_q$, the order of the group $(\text{E}(\mathbb{F}_q),+)$, denoted by $\#\text{E}(\mathbb{F}_q)$, is such that $\#\text{E}(\mathbb{F}_q) \in [1, 2.q+1]$. Hasse's theorem provides tighter bounds for $\#\text{E}(\mathbb{F}_q)$ [87]:*

$$q+1-2\sqrt{q} \leq \#\text{E}(\mathbb{F}_q) \leq q+1+2\sqrt{q}$$

In order to illustrate how the group operation is defined for elliptic curve groups, we consider the particular case where the elliptic curve is defined over the real number field $\mathbb{R}$. Figure 8 shows the geometric addition and doubling of elliptic curve points, which are defined through the following rules:

- Let $P \in \text{E}(\mathbb{F}_q)$. Then, $P+O=P$ and $O+P=P$. Thus, O serves as the additive identity for the group $(\text{E}(\mathbb{F}_q),+)$.
- Let $P=(x,y) \in \text{E}(\mathbb{F}_q)^*$. Then, the inverse of P is $-P=(x,-y)$. In Figure 8, the point $-P$ represents a reflexion of P in the x-axis.
- Let $(P=(x,y), Q=(x',y')) \in \text{E}(\mathbb{F}_q)^* \times \text{E}(\mathbb{F}_q)^*$. If $x \neq x'$, then $P+Q=-R$. In Figure 8, $-R$ is a reflection of R in the x-axis and R is the point of intersection of the line joining P and Q with the elliptic curve.
- Let $P=(x,y) \in \text{E}(\mathbb{F}_q)^*$. Then $2\cdot P = P+P=-R$. In Figure 8, $-R$ is a reflection of R in the x-axis and R is the point of intersection of the tangent at P with the elliptic curve.

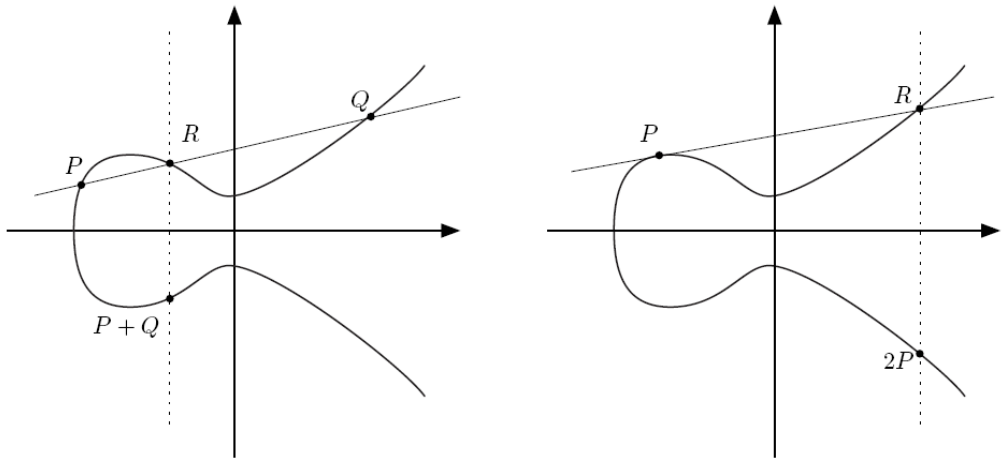


Figure 8: Elliptic curve operations defined over the real number field $\mathbb{R}$

Remark 0.15 *The group $(\mathcal{E}(\mathbb{F}_q),+)$ is commutative and associative i.e. $P+Q=Q+P$ and $(P+Q)+R=P+(Q+R)$, for all $P,Q,R\in\mathcal{E}(\mathbb{F}_q)$. Thus, the composition rules yield an Abelian group $(\mathcal{E}(\mathbb{F}_q),+)$ with identity element O. The geometric definitions presented above lead to algebraic formulae for the group law which are valid for any characteristic of the underlying field, which can be found in [87].*

The notation $r\cdot P$ denotes the scalar multiplication of $P\in\mathcal{E}(\mathbb{F}_q)$ by the integer $r\in\mathbb{Z}$. The value of $r\cdot P$ is the following: for $r=0$ it is equal to O; for $r\leq -1$ it is equal to $(-r)\cdot(-P)$; and for $r\geq 1$ it is equal to $\underbrace{P+\cdots+P}_{r \text{ times}}$.

Scalar multiplication on elliptic curves is believed to be hard to invert. In other words, given $P\in\mathcal{E}(\mathbb{F}_q)$ and $r\cdot P$ for some $r\in\mathbb{Z}_q^*$, it is hard to find r. This referred to as the Elliptic Curve Discrete Logarithm Problem, denoted by ECDLP.

0.3.3 Bilinear Pairings over Elliptic Curves

Let P denote a generator of group $\mathbb{G}_1$, where $\mathbb{G}_1$ is an additive group of some large prime order q. Let $\mathbb{G}_T$ be a related multiplicative group with $|\mathbb{G}_1|=|\mathbb{G}_T|$.

Remark 0.16 *Typically, the group $(\mathbb{G}_1,+)$ is a subgroup of the group of points on an elliptic curve over a finite field $\mathbb{F}_q$, while $\mathbb{G}_T$ is a subgroup of the multiplicative group of a finite field $\mathbb{F}_{q^r}$, where r is known as the embedding degree.*

Definition 0.29 *A **bilinear pairing** is a map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$, with the following properties:*

- ***bilinear**: for $Q, Q' \in \mathbb{G}_1$ and for $a, b \in \mathbb{Z}_q^*$, $e(a \cdot Q, b \cdot Q') = e(Q, Q')^{ab}$*
- ***non-degenerate**: $e(P, P) \neq 1$ and therefore it is a generator of $\mathbb{G}_T$*
- ***computable**: there exists an efficient algorithm to compute $e(Q, Q')$ for all $Q, Q' \in \mathbb{G}_1$*

Remark 0.17 *A bilinear pairing, as presented in Definition 0.29 is symmetric. Indeed, the prime-order group $\mathbb{G}_1$ is cyclic i.e. there is an element $P \in \mathbb{G}_1$ such that for any element $Q \in \mathbb{G}_1$, there exists $s \in \mathbb{Z}_q^*$ such that $Q = s \cdot P$. Hence, given $Q, Q' \in \mathbb{G}_1$, there exists $s, s' \in \mathbb{Z}_q^*$ such that $Q = s \cdot P$ and $Q' = s' \cdot P$. Thus, we have*

$$e(Q, Q') = e(s \cdot P, s' \cdot P) = e(P, P)^{ss'} = e(s' \cdot P, s \cdot P) = e(Q', Q)$$

Remark 0.18 *Typically, a bilinear pairing is derived by modifying either the Tate pairing [77] or the Weil pairing [117, 83] on an elliptic curve over a field $\mathbb{F}_q$. The computational complexity of the Tate pairing is less than that of the Weil pairing. The Weil and Tate pairings need to be modified because the pairings may always return $1_{\mathbb{G}_T}$ (as the self pairing of any point with itself in an unmodified Weil pairing always returns $1_{\mathbb{G}_T}$). In [141], Verheul introduced a tool, called distortion maps, for modifying these pairings. Distortion maps are applicable to a special class of curves called supersingular curves. A distortion map, Φ, is an efficiently computable group endomorphism from $\mathbb{E}(\mathbb{F}_q)$ to $\mathbb{E}(\mathbb{F}_{q^r})$. Applying Φ to one of the inputs to a pairing ensures that the two inputs are linearly independent, therefore, one obtains a non-trivial pairing result. An alternative modification to eliminate trivial pairing results uses trace maps [39] (these are group isomorphisms from $\mathbb{E}(\mathbb{F}_{q^r})$ to $\mathbb{E}(\mathbb{F}_q)$); this technique works on all curves.*

Remark 0.19 *In the general case, a bilinear pairing is of the form $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$, where $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are groups of prime order q. In this thesis, we consider the particular case where $\mathbb{G}_1 = \mathbb{G}_2$. However, our schemes can be adapted to situations in which $\mathbb{G}_1 \neq \mathbb{G}_2$.*

Remark 0.20 *Bilinear pairings over elliptic curves involve fairly complex mathematics, which are detailed in [35]. Fortunately, they can be dealt with abstractly, using only the group structure and mapping properties. As for many interesting schemes in the literature, our policy-based cryptographic schemes are built purely based on abstract bilinear pairings.*

0.3.4 Bilinear Diffie-Hellman Problems

Bilinear Diffie-Hellman generators, which were first introduced in [38], are defined as follows:

Definition 0.30 *A **bilinear Diffie-Hellman generator** is a polynomial-time randomized algorithm, denoted by $\mathcal{IG}(\mathbf{k})$, that takes a security parameter $k \geq 1$ as input, and returns a tuple $(q, \mathbb{G}_1, \mathbb{G}_T, e)$ where the map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$ is a bilinear pairing, $(\mathbb{G}_1, +)$ and $(\mathbb{G}_T, *)$ are two groups of the same order q.*

The security of our policy-based encryption schemes is based on the Bilinear Diffie-Hellman Problem, denoted by BDHP, which is defined as follows:

Definition 0.31 *Given $(q, \mathbb{G}_1, \mathbb{G}_T, e)$, a tuple returned by algorithm $\mathcal{IG}(k)$, and P, a generator of group $\mathbb{G}_1$, the* **BDHP *problem*** *is defined as follows: given a tuple $(P, a\cdot P, b\cdot P, c\cdot P)$ for randomly chosen $a, b, c \in \mathbb{Z}_q^*$, compute the value $e(P,P)^{abc} \in \mathbb{G}_T$. An algorithm $\mathcal{A}$ has an advantage ε in solving* BDHP *if $Pr[\mathcal{A}(P, a\cdot P, b\cdot P, c\cdot P) = e(P,P)^{abc}] \geq \varepsilon$.*

Remark 0.21 *In Definition 0.31, the probability is measured over the random choices of the integers $a, b, c \in \mathbb{Z}_q^*$, $P \in \mathbb{G}_1^*$ and the random bits of the algorithm $\mathcal{A}$.*

Given the definition of BDHP, the BDH-assumption is defined as follows:

Definition 0.32 *The* **BDH-*assumption*** *states that there exists no probabilistic polynomial time algorithm that has non-negligible advantage in solving the* BDHP *for any tuple $(q, \mathbb{G}_1, \mathbb{G}_T, e)$ generated by $\mathcal{IG}(k)$.*

Remark 0.22 *The validity of the* BDH*-assumption can be ensured by choosing groups on supersingular elliptic curves or hyperelliptic curves over finite fields and deriving the bilinear pairings from Weil or Tate pairings [96]. As we merely apply these mathematical primitives in this thesis, we refer to [148] for further details.*

Definition 0.33 *We define algorithm* BDH-Setup *as follows: on input of a security parameter k, generate a tuple $(q, \mathbb{G}_1, \mathbb{G}_2, e, P)$ where the tuple $(q, \mathbb{G}_1, \mathbb{G}_2, e)$ is generated by a algorithm $\mathcal{IG}(k)$ and the element P is a random generator of $\mathbb{G}_1$. The generated parameters are such that the* BDH*-assumption is valid.*

The security of our policy-based signature scheme relies on the hardness of two mathematical problems: on one hand, the Computational Diffie-Hellman Problem, denoted by CDHP, and, one the other hand, the $(k+1)$-Exponent Problem, denoted by $(k+1)$-EP. The two problems are defined below.

Definition 0.34 *Let $\mathbb{G}$ be a finite cyclic group and let P be a generator of the group $\mathbb{G}$. The* **CDHP *problem*** *is defined as follows: given a tuple $(P, a\cdot P, b\cdot P)$ for randomly chosen $a, b \in \mathbb{Z}_{|\mathbb{G}|}^*$, compute $ab\cdot P$.*

Remark 0.23 *The hardness of* BDHP *implies the hardness of* CDHP*. Indeed, assume one moment that* CDHP *is an easy problem. Given a tuple $(P, a\cdot P, b\cdot P, c\cdot P)$ for randomly chosen $a, b, c \in \mathbb{Z}_q^*$, it is easy to compute the value $ab\cdot P$ and consequently the value $e(P,P)^{abc} = e(ab\cdot P, c\cdot P)$.*

Definition 0.35 *Let $\mathbb{G}$ be a finite cyclic group and let P be a generator of the group $\mathbb{G}$. The* **$(k+1)$-EP *problem*** *is defined as follows: given the k-tuple $(P, a\cdot P, a^2\cdot P, \ldots, a^k\cdot P)$ for some $a \in \mathbb{Z}_q^*$, compute $a^{k+1}\cdot P$.*

Remark 0.24 *The $(k+1)$-EP problem is known to be no harder than the* CDHP *problem. See [151] for further details.*

0.3.5 Pairing-Based Cryptographic Schemes

In 2001, Boneh and Franklin used bilinear pairings over elliptic curves to design an identity-based encryption scheme [38]. Apart from the fact that they managed to construct the first practical and provably secure identity-based encryption scheme, they introduced to the wider research community a new tool, namely bilinear pairings, that allows to construct a wide range of cryptographic schemes with performance and properties that cannot be achieved using standard mathematical primitives. Before the publication of Boneh and Franklin, some other publications on the usage of pairings in cryptography can be found in the literature. In fact, in 2000, Sakai et al [128] and Joux [95] independently proposed two cryptographic protocols using pairings over elliptic curves. On one hand, the protocol proposed by Joux is similar to the Diffie Hellman protocol [63], except the fact that, instead of two interacting entities, it allows three entities to create and exchange a secret key. On the other hand, Sakai et al. presented a non-interactive identifier-based key agreement protocol. In their scheme, a hash function is used to map identifiers to elements of a group $\mathbb{G}_1^*$. This mapping function and the way in which Sakai et al. use pairings can be found in many subsequent publications.

The three policy-based cryptographic schemes presented in this thesis are based on bilinear pairings. As an appetizer, we describe in the following a couple of simple pairing-based encryption schemes. Furthermore, we apply the Fujisaki-Okamoto transformations defined in Section 0.2.4 to these schemes. The described encryption schemes will be mainly referenced in the reductionist proofs of security described in Chapter 1 and Chapter 2.

We start by describing a pairing-based PKE scheme, denoted by BasicPub, which was first defined by Boneh and Franklin in [38].

Definition 0.36 *The* **BasicPub** *scheme consists of four algorithms:* Setup, KeyGen, Encrypt *and* Decrypt, *which are defined as follows:*

- **Setup**. *On input of the security parameter k, do the following:*
  1. *Run algorithm* BDH-Setup *(Definition 0.33) to obtain a tuple* $(q, \mathbb{G}_1, \mathbb{G}_T, e, P)$
  2. *Let* $\mathcal{M} = \{0,1\}^n$ *and* $\mathcal{C} = \mathbb{G}_1 \times (\{0,1\}^n)^*$ *(for some* $n \in \mathbb{N}^*$*)*
  3. *Define a hash function:* $H_2 : \{0,1\}^* \rightarrow \{0,1\}^n$
  4. *Let* $\mathcal{P} = (q, \mathbb{G}_1, \mathbb{G}_T, e, P, n, H_2)$
- **KeyGen**. *Pick at random a point* $Q \in \mathbb{G}_1$. *Then, pick at random a private key* $sk \in \mathbb{Z}_q^*$ *and let the tuple* $(pk = sk \cdot P, Q)$ *be the corresponding public key.*
- **Encrypt**. *On input of a message* $M \in \mathcal{M}$ *and public key* (pk, Q), *do the following: pick at random* $r \in \mathbb{Z}_q^*$ *and* $U = r \cdot P$. *Then, compute* $\pi = e(pk, Q)$ *and* $W = M \oplus H_2(\pi^r)$. *Finally, return* $C = (U, W)$.

- **Decrypt**. *On input of ciphertext $C = (U, W)$ and private key sk, do the following: compute $\tilde{\pi} = e(U, sk)$, then return $M = W \oplus H_2(\tilde{\pi})$.*

In [38], the BasicPub scheme is proved to be IND-CPA secure in the random oracle model under the BDH-assumption (Definition 0.32). The FO_I transformation (presented in Section 0.2.4) applied to this scheme leads to the $\mathsf{BasicPub}^{hy_I}$ scheme defined below.

Definition 0.37 *The* **BasicPub$^{\mathbf{hy_I}}$** *scheme consists of four algorithms:* Setup, KeyGen, Encrypt *and* Decrypt, *which are defined as follows:*

- **Setup**. *On input of the security parameter k, do the following:*
  1. *Run algorithm* BDH-Setup *(Definition 0.33) to obtain a tuple $(q, \mathbb{G}_1, \mathbb{G}_T, e, P)$*
  2. *Let $\mathcal{M} = \{0,1\}^n$ and $\mathcal{C} = \mathbb{G}_1 \times (\{0,1\}^n)^* \times \{0,1\}^n$ (for some $n \in \mathbb{N}^*$)*
  3. *Define three hash functions: $H_1 : \{0,1\}^* \to Z_q^*$, $H_2 : \{0,1\}^* \to \{0,1\}^n$, and $H_3 : \{0,1\}^n \to \{0,1\}^n$*
  4. *Let $\mathcal{P} = (q, \mathbb{G}_1, \mathbb{G}_T, e, P, n, H_1, H_2, H_3)$*
- **KeyGen**. *Pick at random a point $Q \in \mathbb{G}_1$. Then, pick at random a private key $sk \in \mathbb{Z}_q^*$ and let the tuple $(pk = sk \cdot P, Q)$ be the corresponding public key.*
- **Encrypt**. *On input of a message $M \in \mathcal{M}$ and public key (pk, Q), do the following: pick at random $t \in \{0,1\}^n$, then compute $r = H_1(M||t) \in \mathbb{Z}_q^*$ and $U = r \cdot P$. Compute $\pi = e(pk, Q)$, then compute $v = t \oplus H_2(\pi^r)$ and $W = M \oplus H_3(t)$. Return $C = (U, v, W)$.*
- **Decrypt**. *On input of ciphertext $C = (U, v, W)$ and private key sk, do the following: compute $\tilde{\pi} = e(U, sk)$, then compute $t = v \oplus H_2(\tilde{\pi})$ and $M = W \oplus H_3(t)$. If $U = H_1(M||t) \cdot P$, then return M. Otherwise, return $\perp$.*

The BasicPub scheme is IND-CPA secure and therefore OW-CPA secure in the random oracle model under the BDH-assumption. The FO_{II} transformation (presented in Section 0.2.4) applied to this scheme leads to the $\mathsf{BasicPub}^{hy_{II}}$ scheme defined below.

Definition 0.38 *The* **BasicPub$^{\mathbf{hy_{II}}}$** *scheme consists of four algorithms:* Setup, KeyGen, Encrypt *and* Decrypt, *which are defined as follows:*

- **Setup**. *On input of the security parameter k, do the following:*
  1. *Run algorithm* BDH-Setup *(Definition 0.33) to obtain a tuple $(q, \mathbb{G}_1, \mathbb{G}_T, e, P)$*
  2. *Let $\mathcal{M} = \{0,1\}^{n-n_0}$ and $\mathcal{C} = \mathbb{G}_1 \times (\{0,1\}^n)^*$ (for some $n, n_0 \in \mathbb{N}^*$ such that $n_0 \ll n$)*
  3. *Define two hash functions: $H_1 : \{0,1\}^* \to \mathbb{Z}_q^*$ and $H_2 : \{0,1\}^* \to \{0,1\}^n$*

4. *Let* $\mathcal{P} = (q, \mathbb{G}_1, \mathbb{G}_T, e, P, n, H_1, H_2)$

- **KeyGen**. *Pick at random a point* $Q \in \mathbb{G}_1$. *Then, pick at random a private key* $sk \in \mathbb{Z}_q^*$ *and let the tuple* $(pk = sk \cdot P, Q)$ *be the corresponding public key.*
- **Encrypt**. *On input of a message* $M \in \mathcal{M}$ *and public key* (pk, Q), *do the following: pick at random* $t \in \{0,1\}^{n_0}$, *then compute* $r = H_1(M||t) \in \mathbb{Z}_q^*$ *and* $U = r \cdot P$. *Next, compute* $\pi = e(pk, Q)$, *then compute* $W = (M||t) \oplus H_2(\pi^r)$. *Return* $C = (U, W)$.
- **Decrypt**. *On input of ciphertext* $C = (U, W)$ *and private key sk, do the following: compute* $\tilde{\pi} = e(U, sk)$, *then compute* $M||t = W \oplus H_2(\tilde{\pi})$. *If* $U = H_1(M||t) \cdot P$, *then return M. Otherwise, return* $\perp$.

Now, we describe a pairing-based PKE scheme, denoted by EIG-BasicPub (first defined in [3]) as an application of the original ElGamal PKE scheme [82] to groups on elliptic curves.

Definition 0.39 *The* **EIG-BasicPub** *scheme consists of four algorithms:* Setup, KeyGen, Encrypt *and* Decrypt, *which are defined as follows:*

- **Setup**. *On input of the security parameter k, do the following:*
  1. *Run algorithm* BDH-Setup *(Definition 0.33) to obtain a tuple* $(q, \mathbb{G}_1, \mathbb{G}_T, e, P)$
  2. *Let* $\mathcal{M} = \{0,1\}^n$ *and* $\mathcal{C} = \mathbb{G}_1 \times \{0,1\}^n \times \{0,1\}^n$ *(for some* $n \in \mathbb{N}^*$*)*
  3. *Define a hash function:* $H_2 : \mathbb{G}_1^* \to \{0,1\}^n$
  4. *Let* $\mathcal{P} = (q, \mathbb{G}_1, \mathbb{G}_T, e, P, n, H_2)$
- **KeyGen**. *Pick at random a private key* $sk \in \mathbb{Z}_q^*$ *and let* $pk = sk \cdot P$ *be the corresponding public key.*
- **Encrypt**. *On input of a message* $M \in \mathcal{M}$ *and public key pk, do the following: pick at random* $r \in \mathbb{Z}_q^*$, *then compute* $U = r \cdot P$ *and* $W = M \oplus H_2(r \cdot pk)$. *Return* $C = (U, W)$.
- **Decrypt**. *On input of ciphertext* $C = (U, W)$ *and private key sk, then return the message* $M = W \oplus H_2(sk \cdot U)$.

The EIG-BasicPub scheme is proved to be IND-CPA secure in the random oracle model under the assumption that CDHP is hard. The FO_I transformation applied to this scheme leads to the EIG-BasicPub^{hy_I} scheme defined below.

Definition 0.40 *The* **EIG-BasicPub^{hy_I}** *scheme consists of four algorithms:* Setup, KeyGen, Encrypt *and* Decrypt, *which are defined as follows:*

- **Setup**. *On input of the security parameter k, do the following:*

1. *Run algorithm* BDH-Setup *(Definition 0.33) to obtain a tuple* $(q, \mathbb{G}_1, \mathbb{G}_T, e, P)$
2. *Let* $\mathcal{M} = \{0,1\}^n$ *and* $\mathcal{C} = \mathbb{G}_1 \times \{0,1\}^n$ *(for some* $n \in \mathbb{N}^*$*)*
3. *Define three hash functions:* $H_1 : \{0,1\}^* \to Z_q^*$, $H_2 : \mathbb{G}_1^* \to \{0,1\}^n$, *and* $H_3 : \{0,1\}^n \to \{0,1\}^n$
4. *Let* $\mathcal{P} = (q, \mathbb{G}_1, \mathbb{G}_T, e, P, n, H_1, H_2, H_3)$

- **KeyGen**. *Pick at random a private key* $sk \in \mathbb{Z}_q^*$ *and let* $pk = sk \cdot P$ *be the corresponding public key.*
- **Encrypt**. *On input of a message* $M \in \mathcal{M}$ *and public key pk, do the following: pick at random* $t \in \mathbb{Z}_q^*$, *then compute* $r = H_1(M||t) \in \mathbb{Z}_q^*$, $U = r \cdot P$, $v = t \oplus H_2(r \cdot pk)$ *and* $W = M \oplus H_3(t)$. *Return* $C = (U, v, W)$.
- **Decrypt**. *On input of ciphertext* $C = (U, v, W)$ *and private key sk, do the following: compute* $t = v \oplus H_2(sk \cdot U)$, *then compute* $M = W \oplus H_3(t)$. *If* $U = H_1(M||t) \cdot P$, *then return M. Otherwise, return* $\perp$.

Remark 0.25 *Note that the Fujisaki-Okamoto transformations presented in this section correspond to the particular case where the* E^{sym} *scheme is the one-time pad denoted by* $\oplus$.

0.4 Conclusion

In this chapter, we presented the background notions and the notational conventions that will be used in the sequel of this manuscript. In particular, we briefly presented the foundations of the concept of provable security, including the reductionist proof strategy, the random oracle model as well as the formal security models from which will be derived the security models related to our policy-based encryption and signature primitives. Moreover, we gave an overview of bilinear pairings over elliptic curves, which are used in the proposed policy-based cryptographic schemes. Now, the reader can move on either to Chapter 1 then to Chapter 2, which both tackle policy-based encryption, or to Chapter 3, which independently tackles policy-based signature.

CHAPTER 1

Policy-Based Encryption

1.1 Introduction

Cryptography is considered as the art and science of encryption, at least in its beginnings. It is therefore natural to start our formalization of the concept of policy-based cryptography by addressing the policy-based encryption primitive. As for asymmetric encryption schemes, a policy-based encryption scheme allows an entity to encrypt a message using a 'public' information in such a way that only a legitimate entity can decrypt it using the associated 'private' information. While in identity-based encryption schemes the legitimacy of an entity that is allowed to decrypt an encrypted message is based on its identity, it is based on its compliance with a credential-based policy in policy-based encryption schemes. The shift from the identity-based approach to our policy-based approach is discussed below.

The concept of identity-based cryptography was first formulated by Shamir in 1984 [137]. Its primary goal was to eliminate the need for directories and certificates used in public-key infrastructures. This is achieved by using the identity of the recipient a message is intended for as its public key. Informally, an identity-based encryption scheme allows to encrypt a message with respect to an identity in a way such that only an entity whose identity is the one according to which the message was encrypted can decrypt the message. No identity-based encryption scheme proposed between 1984 and 2000 was fully satisfactory in terms of both security and efficiency. In 2001, Boneh and Franklin managed to develop a secure and practical identity-based encryption scheme [38] using bilinear pairings over elliptic curves [96].

The identity-based encryption primitive is depicted in Figure 1.1.

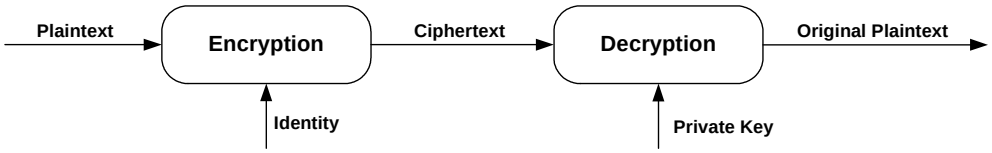


Figure 1.1: Identity-Based Encryption

The identity of an entity is rarely relevant for deciding whether it should be authorized to have access to a sensitive message. This is especially true in the context of large-scale open environments like the Internet, as in such environments, interactions often occur between entities from different security domains without pre-existing knowledge of each other. An increasingly popular approach consists in expressing authorization requirements through policies fulfilled by digital credentials. A policy consists of conjunctions (logical AND operation) and disjunctions (logical OR operation) of conditions, where each condition is fulfilled by a specific credential. A credential is the signature of a specific trusted entity, called credential issuer, on a certain assertion about an entity, called credential owner. The assertion consists of a set of statements, where each statement can be an attribute, a property, an authorization, *etc*. Together with the set of statements, the assertion may optionally contain additional information such as the identity of the entity or the validity period of the assertion.

Because the identity of the recipient of an encrypted message is not relevant to decide whether it should be authorized to have access to the message, identity-based encryption need to be used subsequently to a mechanism that proves the compliance of the recipient with the authorization policy defined by the owner of the message. The standard approach is that the owner of the message first receives a set of credentials from the entity the message is intended for. Then, it has to verify the validity of each of the received credentials, and finally has to check that the received set of credentials is effectively a qualified set of credentials for the policy. According to this approach, two separate mechanisms are required to implement the authorization and confidentiality security services. The notion of ‘identity’, which is central to the identity-based encryption primitive, just represents the logical link between the two mechanisms.

The policy-based encryption primitive achieves confidentiality and authorization in a logically single step. Indeed, a policy-based encryption scheme allows an entity to encrypt a message with respect to a policy in such a way that only an entity that is compliant with the policy is able to decrypt the message. In contrast with the conventional approach where an entity needs to disclose its credentials in order to prove its compliance with an authorization policy, the credentials in policy-based encryption are private as they are used as inputs to the decryption algorithm. Intuitively the notion of ‘policy’ is for policy-based encryption what the notion of ‘identity’ is for identity-based encryption.

An illustration of the policy-based encryption primitive is shown in Figure 3.1.

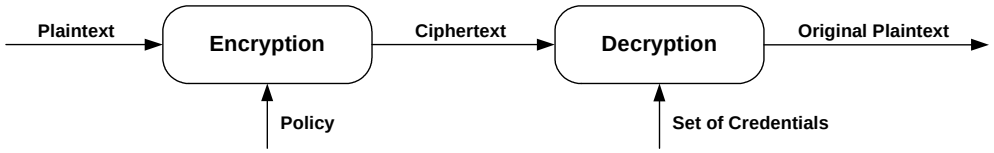


Figure 1.2: Policy-Based Encryption

The policy-based encryption primitive has to fulfill the following requirements:

- **policy-based confidentiality**: in contrast with identity-based encryption whereby the confidentiality is based on the identity of the entity the message is intended for, the confidentiality of a message encrypted using a policy-based encryption algorithm should be based on its compliance with the policy according to which the message is encrypted. This property translates the fact that policy-based encryption achieves both confidentiality and authorization in a logical single step as mentioned before. For the sake of normalization, the considered policies should be formalized as monotone Boolean expressions written in standard normal forms.

- **multiple credential issuers**: a policy according to which a message is encrypted has to support credentials issued by multiple independent credential issuers, where each issuer corresponds to a specific, autonomous and limited administrative domain and responsibility area. This is because a centralized credential issuer cannot be responsible and cannot even be trusted to check and certify the validity of all the types of assertions for all the possible communicating entities. Besides, an entity can require, for example, that in order for an assertion to be considered as valid it has to be checked and certified by two independent credential issuers. Furthermore, two entities can trust two different credential issuers for checking and certifying the validity of the same assertion.

- **independence from credential generation**: the encryption algorithm and the credential system should be defined in such a way that the encryption of a message is independent from the issuance of credentials. In other words, it should be possible to encrypt a message with respect to a policy before the issuance of a qualified set of credentials for the policy. Indeed, some credentials can be issued only when some action has been performed or some event has already occurred and this should not affect the time at which a message can be encrypted.

- **public representation of policies**: the public information used by the encryption algorithm should be easily derived from the cleartext representation of the policy according to which a message is encrypted. This property is equivalent to the one fulfilled by identity-based encryption schemes where the public information used to encrypt a message is easily derived from the plaintext identity of the recipient the message is intended for.

- **performance**: a concrete implementation of the policy-based encryption primitive can be achieved using a basic identity-based encryption scheme. This is performed according to a naive approach that consists in using multiple encryptions for disjunctions and

onion-like encryptions for conjunctions, while replacing, in the identity-based encryption algorithm, the identity by an assertion and the public key of the private key generation authority by the public key of the credential issuer that is specified by the policy to check and certify the validity of the assertion. The performance of this approach is obviously not satisfactory from the perspective of both computational efficiency and bandwidth consumption (ciphertext size), especially when the policy according to which the encryption is performed becomes complex. A concrete implementation of policy-based encryption must therefore improve the performance of this naive approach.

- **provable security**: a policy-based encryption scheme has to be provably secure under a well-defined and strong security model that takes into consideration the specific features of policy-based encryption. Here, we consider indistinguishability against chosen ciphertext attacks in the random oracle model.

In this chapter, we formalize the concept of policy-based encryption, we propose a provably security policy-based encryption scheme and prove its usefulness through the description of three application scenarios. The rest of the chapter is organized as follows: in Section 1.2 we discuss a number of cryptographic schemes related to our policy-based encryption primitive. In Section 1.3, we present our policy model and the related terminology. Then, we formally define policy-based encryption and present our formal model of semantic security against chosen ciphertext attacks for policy-based encryption schemes. In Section 1.4, we first describe our policy-based encryption scheme from bilinear pairings. Then, we discuss its consistency and efficiency before analyzing its security in the random oracle model under the BDH assumption. In Section 1.5, we present a framework for controlling access to released XML documents using the policy-based encryption primitive. This application illustrates how the policy-based encryption primitive can be elegantly used as a policy enforcement mechanism in the context of tree-structured documents. In Section 1.6.3, we show how our primitive can be used in combination with exisiting tools to implement the sticky privacy policy paradigm. Finally, in Section 1.7, we show how the policy-based encryption primitive can be used for privacy-enhanced establishment of ad-hoc communities. More precisely, we focus on the privacy principle of data minimization which cannot be fulfilled using a standard enforcement mechanism of credential-based policies.

1.2 Related Work

The concept of policy-based encryption allows to encrypt a message with respect to a policy formalized as monotone Boolean expression written in a standard normal form. The policy represents an access structure defining the conditions under which access to the message is permitted. Many advanced cryptographic primitives dealing with different forms of access structures can be found in the literature. In this section, we discuss the relevant ones and compare them with our policy-based encryption primitive. Besides our pairing-based implementation

of policy-based encryption owes much to recent publications on pairing-based cryptography in general, and in particular on identity-based encryption. More details are given below.

Pairing-Based Short Signatures

In [39], Boneh et al. introduce a short signature scheme from bilinear pairings over elliptic curves. Their scheme achieves signatures that are much shorter than all the current variants of RSA and DSA signatures [94, 147], while achieving an equivalent security level. For instance, for a 1024-bit security level, their signature scheme produces 171-bit signatures, while standard DSA signatures are 320 bits long [147] and standard RSA signatures are 1024 bits long. Their short signature scheme is extensively used by the different pairing-based schemes presented in this thesis for the purpose of generating digital credentials.

Identity-Based Encryption using Bilinear Pairings

In [38], Boneh and Franklin propose an identity-based encryption scheme, denoted by IBE, using bilinear pairings over elliptic curves. They define the notion of semantic security against chosen ciphertext attacks for IBE schemes, and prove the security of their scheme in the random oracle model under the BDH assumption (Definition 0.32). As mentioned in their paper, the key structure of their scheme allows to associate public keys to strings which may represent not only the recipient's identity but also any set of statements about him in which case the private key of the recipient plays the role of a credential, whereas the private key generator (PKG) plays the role of a credential issuer. Their IBE scheme can be therefore seen as a policy-based encryption scheme for which the policies are restricted to one condition fulfilled by a credential delivered by a single credential issuer.

As for the majority of existing pairing-based cryptographic schemes, our work on policy-based encryption owes much to the work of Boneh and Franklin. Indeed, our formal definitions for policy-based encryption schemes and the related security models are inspired by the ones given in [38] for IBE schemes. Furthermore, the method we use to prove the security of our scheme is similar but not exactly the same as the one used in [38].

Multi-Recipient and Broadcast Encryption

The concept of multi-recipient (receiver) encryption, denoted by MRE, was independently formalized by Bellare et al. in [26], and Baudron et al. in [24]. Informally, an MRE scheme allows a user to encrypt a sequence of messages using a sequence of public keys in such a way that each message can be decrypted only by a receiver having access to the corresponding private

key. In [26, 24], the authors show that the security of public-key encryption in the 'single-receiver' setting implies the security in the 'multi-recipient' setting. In other words, one can construct a semantically secure MRE scheme by simply encrypting a message under multiple public keys of a semantically secure PKE scheme.

For the sake of performance, Kurosawa proposed in [104] a technique, called *randomness re-use*, that is used to construct an MRE scheme derived from the El-Gamal PKE scheme [82]. Later, the randomness re-use technique was refined by Bellare et al. in [27] where they provide a general test that can be applied to a standard PKE scheme to determine whether the associated MRE scheme built using the randomness re-use technique is still secure. The randomness re-use technique will be later used in our concrete implementation of policy-based encryption to deal with disjunctions of conditions.

The concept of broadcast encryption, denoted by BE, was first formalized by Fiat and Naor in [68]. It is similar but not exactly the same as the concept of multi-recipient encryption. Both MRE and BE schemes share the fact that they deal with access structures that consist of disjunctions of conditions where each condition is fulfilled by a specific key exclusively held by an end user. In [27], the authors explain the differences between the two concepts as follows: 1) the key generation process in a BE scheme may be executed by the sender and yields a sequence of possibly related encryption keys (one per recipient), while key generation in a MRE scheme is similar to standard PKE scheme, meaning that each recipient produces (and registers) its own pair of keys for its own use; 2) the encryption process in a BE scheme takes as input a sequence of encryption keys and a single message and produces a single ciphertext, called a broadcast ciphertext, while the encryption process in a MRE scheme takes as input a sequence of encryption keys and a sequence of messages, and produces a corresponding sequence of ciphertexts, one for each recipient.

In [10], Baek et al. propose an efficient and provably secure multi-recipient identity-based encryption scheme, denoted by MR-IBE. They discuss how their scheme can lead to an efficient public-key broadcast encryption scheme based on the 'subset-cover' framework. The key structure of their scheme is similar to the one used in the IBE scheme of Boneh and Franklin [38]. The private keys can play the role of credentials and their scheme can be seen as a policy-based encryption scheme whereby the policies are restricted to disjunctions of conditions fulfilled by credentials issued by multiple credential issuers.

Access Control using Pairing-Based Cryptography

In [52], Chen et al. present various applications of the use of multiple trusted authorities and multiple identities in the type of identity-based cryptography. They show how to perform encryption according to disjunctions and conjunctions of credentials. However, their solution remains restricted to a limited number of disjunctions. In [138], Smart further pursues the ideas discussed in [52] and presents an elegant and efficient mechanism to perform access control

based on encryption with respect to monotone Boolean expressions written in standard normal forms. However, the solution is limited to credentials generated by a centralized credential issuer. Furthermore, the proposed scheme lacks formal security arguments.

Hidden Credentials and Oblivious Signature-based Envelopes

In [92], Holt et al. introduce the concept of *hidden credentials* as a solution to perform privacy-enabled trust negotiation. Their solution uses the Boneh-Franklin IBE scheme [38] and relies on onion-like encryption and multiple encryption operations to deal, respectively, with conjunctions and disjunctions of credentials. Such approach remains inefficient in terms of both computational costs and bandwidth consumption (ciphertext size), especially when authorization structures become very complex. In [42], Bradshaw et al. propose a solution to improve decryption efficiency as well as policy concealment when implementing hidden credentials with sensitive policies. They prove the security of their solution while relying on the security models defined for IBE schemes, as opposed to our approach that considers dedicated policy-oriented security models. Further details about the concept of trust negotiation are given in Section 2.5.

In [110], Li et al. propose the concept of oblivious signature-based envelopes, denoted by OSBE, for efficiently solving the cyclic policy interdependency problem. They describe a one-round OSBE protocol based on the Boneh-Franklin IBE scheme [38]. Later work specified generalized OSBE, denoted by GOSBE, which allows dealing with general monotone access structures [108, 111]. We refer to [91] for a detailed analysis of the subtle differences between OSBE, hidden credentials and related encryption schemes. The different schemes presented in [92, 42, 111], consider policies formalized as monotone Boolean expressions represented as general conjunctions and disjunctions of atomic terms. The size of the resulting ciphertexts linearly depends on the number of these terms, whereas the normal forms considered by policy-based encryption schemes, as will be shown later in this chapter, substantially reduce the size of the produced ciphertexts in addition to improving the computational performance of the proposed schemes.

Threshold and Generalized Threshold Decryption

The concept of threshold cryptography was first addressed in [36, 136]. The primary motivation behind this concept is *"to share the power of a cryptosystem"*, typically in applications where a group of mutually 'suspicious' entities with potentially 'conflicting' interests must cooperate to achieve a common goal [62]. The latter can be, for instance, the decryption of an encrypted message or the generation of a valid signature on a certain message. The fundamental problem of threshold cryptography turns often to secret sharing. A secret sharing scheme [136, 139] allows an entity to distribute a piece of secret information among several entities in a way such that the following two conditions are fulfilled: 1) no group of corrupt entities (smaller than a certain threshold) can figure out what the secret is, even if they cooperate, 2) when it becomes

necessary that the secret information be reconstructed, a large enough number of entities (a number larger than the fixed threshold) can always do it.

Intuitively, a (k,n)-threshold decryption scheme allows to encrypt a message in a way such that the decryption requires the collaboration of at least *k*-out-of-*n* users included in a predefined set of users $\mathcal{U} = \{u_1, \ldots, u_n\}$. The basic setting is as follows: a key generation authority generates a pair of standard public/private keys. The public key will be used to encrypt the messages, while the private key is distributed among the different users according to a (k,n)-secret sharing scheme i.e. each user $u_\kappa \in \mathcal{U}$ is given a piece of the original private key (that is required to decrypt the messages) in a way such that the original private key can be retrieved if and only if at least *k* users included in $\mathcal{U}$ pool their private key shares. The most popular technique used to construct the secret information in a (k,n)-secret sharing is 'Lagrange polynomial interpolation' [136].

While a policy-based encryption primitive allows to encrypt a message according to an access structure defined through a credential-based policy formalized as a monotone Boolean expression written in a standard normal form, a threshold decryption scheme allows to encrypt a message according to an access structure defined through a (k,n)-secret sharing scheme. A (k,n)-threshold access structure can be easily expressed as a monotone Boolean expression written in a standard normal form as well. For example, consider a set of users $\mathcal{U} = \{u_1, u_2, u_3\}$ and a message encrypted using a $(2,3)$-threshold decryption scheme. The message can be decrypted if and only if 2-out-of-3 users collaborate. This is equivalent to saying that the message can be decrypted if and only if either users (u_1, u_2), or users (u_1, u_3) or users (u_2, u_3) cooperate, which corresponds to a monotone Boolean expression written in the disjunctive normal form.

A number of identity-based threshold decryption schemes, denoted by ID-ThD, have recently been proposed in the literature [65, 112, 11]. By analogy with the multi-recipient identity-based encryption primitive discussed above, one might think that an ID-ThD scheme can be seen as a policy-based encryption scheme whereby policies are restricted to the monotone Boolean expressions that are equivalent to threshold access structures. This is not true because the intuition behind ID-ThD schemes is to define the group of users $\mathcal{U}$ through an identity from which will be derived the public key used to encrypt the messages, while the corresponding private key (which might be compared to a credential) is splitted according to an adequate secret sharing scheme and distributed among the users included in $\mathcal{U}$. Actually, what is needed is a threshold decryption scheme for which the encryption is performed with respect to the identities of the members of $\mathcal{U}$ so that these identities can be replaced by the assertions specified by a policy.

In [50], Chai et al. propose an identity-based broadcast threshold decryption scheme that can be seen as a policy-based encryption scheme for which policies are restricted to monotone Boolean expressions that are equivalent to threshold access structures. In fact, in their scheme, in contrast with the basic ID-ThD schemes, each user included in $\mathcal{U}$ is assigned an identifier and is given the corresponding private key (issued by the key generation authority). A message is encrypted with respect to the identifiers of the users included in $\mathcal{U}$, and its decryption requires at least *k* of the users' identifier-based private keys. A policy-based encryption primi-

tive offers an advantage over the identity-based broadcast threshold decryption scheme in that it addresses general access structures including those for which there exists no corresponding (k, n)-threshold representation. Such structures exist according to Theorem 1 of [32]. Although any threshold structure may be written as a normal-form Boolean expression, we believe that dedicated identity-based broadcast threshold schemes might handle such structures more efficiently and elegantly than any policy-based encryption scheme. In fact, as a general rule, a dedicated solution is almost always more efficient than a generic one.

The concept of generalized threshold cryptosystems was introduced in [105]. It is basically an extension of the original concept of threshold cryptography to the case of general access structures. Our concrete implementation of policy-based encryption scheme uses a technique similar to but not exactly the same than the secret sharing method presented in [32]. In fact, the latter is used for the implementation of the hidden credentials system proposed in [42], which is less efficient than our scheme both in terms of computational cost and ciphertext size. In [76], it has been observed that using the previous general secret sharing method it is possible to construct an RSA-based generalized threshold decryption scheme. In contrast with the RSA-oriented approach, our policy-based encryption primitive supports the notion of cryptographic workflow which allows to encrypt a message with respect to a specific access structure before the decryption keys (the credentials) are generated and given to the authorized users. In other words, the encryption algorithm does not depend on the generation of a set of credentials required to successfully perform the decryption.

Summary of Related Work

Encrypting a message with respect to an access structure formalized as monotone Boolean expression clearly is not a new concept in cryptography. In contrast with the majority of the discussed schemes, the concept of policy-based encryption is generic and its applications are not limited to specific contexts. The functionality of policy-based encryption can be partially achieved using some of the discussed schemes: an identity-based encryption scheme [38] can be used to address the case where policies are limited to single conditions, an identity-based multi-recipient scheme [10] can be used to address the case where policies are limited to conjunctions of conditions, and an identity-based broadcast threshold decryption scheme [50] can be used to address the case where policies are limited to the monotone Boolean expressions that are equivalent to threshold structures. A fully functional policy-based encryption scheme can be realized using the encryption scheme presented in [42]. However, the proposed scheme is less efficient than the one proposed in this chapter and is not supported by adequate security arguments.

1.3 Formal Definitions

1.3.1 Policy Model

We consider a set of credential issuers $\mathcal{I} = \{I_1, \ldots, I_N\}$, where the public key of a credential issuer I_κ, for $\kappa \in \{1, \ldots, N\}$, is denoted by R_κ while the corresponding master key is denoted by s_κ. We assume that a trustworthy value of the public key of each of the credential issuers is known by the end users. Any credential issuer $I_\kappa \in \mathcal{I}$ may be asked by an end user to issue a credential corresponding to a set of statements. The requested credential is basically the digital signature of the credential issuer on an assertion denoted by A. The assertion contains, in addition to the set of statements, an optional identifier of the end user as well as a set of additional information such as the validity period of the credential.

Upon receiving a request for generating a credential on assertion A, a credential issuer I_κ first checks the validity of the assertion A. If the assertion is valid, then I_κ executes a credential generation algorithm and returns a credential denoted by $\varsigma(R_\kappa, A)$. Otherwise, I_κ returns an error message. Upon receiving the credential $\varsigma(R_\kappa, A)$, the end user may check its integrity using the public key of issuer I_κ. As the credentials will play the role of decryption keys in policy-based encryption, we assume that they are transmitted to the requesters through secure channels.

Remark 1.1 *In the following, we give some remarks about the content and the representation of the considered assertions:*

- *As said before, an assertion contains basically a set of statements, an optional identifier of the requester, and a set of additional information such as the validity period of the credential. There are different types of statements such as authentication statements, attribute statements and authorization statements as defined in [75]. In some applications, an identifier (such as a name, an e-mail address or a public-key) need to be included in the assertion in order to bind the issued credential to a unique user. In other applications, the same credential need to be shared by multiple users, in which case the legitimate users have to be trusted or forced for not releasing the credential to illegitimate users. ...*

- *There are many alternatives for encoding the considered assertions such as text-based encoding, ASN.1, the XML-based SAML language,* etc. *In this thesis, we are not concerned with the encoding of assertions. We only assume that there is some kind of ontology that makes the different players understand the content and the representation of the considered assertions and the corresponding issued credentials. A policy-based encryption scheme must not depend on any encoding of assertions. Thus, assertions will simply be encoded as binary strings.*

- *The process of checking the validity of an assertion and issuing the corresponding credential for new users is out of the scope of this thesis. Hence, we will only assume that*

the different credential issuers are 'trusted' for not issuing credentials corresponding to invalid assertions.

We consider credential-based policies formalized as monotone Boolean expressions involving conjunctions (AND / $\wedge$) and disjunctions (OR / $\vee$) of credential-based conditions. A credential-based condition is defined through a pair $\langle I_\kappa, A\rangle$ specifying an assertion $A \in \{0,1\}^*$ and a credential issuer $I_\kappa \in \mathcal{I}$ that is trusted to check and certify the validity of A. A user fulfills the condition $\langle I_\kappa, A\rangle$ if and only if it has been issued the credential $\varsigma(R_\kappa, A)$. Our policies are written in standard normal forms, i.e. written either in conjunctive normal form (CNF) or in disjunctive normal form (DNF). In order to address the two standard normal forms, we use the conjunctive-disjunctive normal form (CDNF) introduced in [138]. Thus, a policy denoted by *Pol* is written as follows:

$$Pol = \wedge_{i=1}^{m}[\vee_{j=1}^{m_i}[\wedge_{k=1}^{m_{i,j}}\langle I_{\kappa_{i,j,k}}, A_{i,j,k}\rangle]], \text{ where } I_{\kappa_{i,j,k}} \in \mathcal{I} \text{ and } A_{i,j,k} \in \{0,1\}^*$$

Under the CDNF notation, the following holds:

- Policies written in CNF correspond to the case where $\{m_{i,j} = 1\}_{i,j}$
- Policies written in DNF correspond to the case where $m = 1$

Remark 1.2 *In this thesis, we address standard normal forms instead of general monotone Boolean expression not only for the sake of normalization but also for performance considerations (see Section 1.4.2). Besides, we address both the* CNF *and* DNF *forms instead of addressing only one of the two standard forms for the sake of completeness and flexibility.*

For the sake of readability, we introduce the following notations:

- $\varsigma(Pol)$ denotes the set of all the keys corresponding to the different conditions specified by policy *Pol* i.e. $\varsigma(Pol) = \{\{\varsigma(R_{\kappa_{i,j,k}}, A_{i,j,k})\}_{k=1}^{m_{i,j}}\}_{j=1}^{m_i}\}_{i=1}^{m}$
- $\check{\varsigma}(Pol)$ denotes the power set of $\varsigma(Pol)$ i.e. the set of all the subsets of $\varsigma(Pol)$
- For some $\{j_i \in \{1,\ldots,m_i\}\}_{i=1}^{m}$, $\varsigma_{j_1,\ldots,j_m}(Pol) = \{\{\varsigma(R_{\kappa_{i,j_i,k}}, A_{i,j_i,k})\}_{k=1}^{m_{i,j_i}}\}_{i=1}^{m}$
- For a subset of credentials $\rho \subset \check{\varsigma}(Pol)$, '$\rho \vDash Pol$' denotes the fact that ρ is a qualified set of credentials for policy *Pol*

According to the notation defined above, the following equivalence holds:

$$\forall\ \rho \in \check{\varsigma}(Pol):\ \ \rho \vDash Pol \Leftrightarrow \exists\{j_i \in \{1,\ldots,m_i\}\}_{i=1}^{m}\ \text{s.th.}\ \rho = \varsigma_{j_1,\ldots,j_m}(Pol) \tag{1.1}$$

An illustration of our policy model is given below.

Example 1.1 *Assume that there exists a set of credential issuers* $\mathcal{I} = \{I_1, I_2, I_3, I_4\}$*, where each issuer* I_κ*, , for* $\kappa = 1,2,3,4$*, is trusted by the data owner to issue credentials* A_κ *and* A'_κ*.*

- *An example of a policy written in* DNF *is given below:*

$$Pol_1 = (\langle R_1, A_1\rangle \wedge \langle R_2, A_2\rangle) \vee (\langle R_3, A_3\rangle \wedge \langle R_4, A_4\rangle)$$

Given the CDNF*-based notation defined above, policy* Pol_1 *is such that*

$$\begin{cases} m = 1, m_1 = 2 \\ m_{1,1} = 2 : \langle L_{1,1,1}, A_{1,1,1}\rangle = \langle R_1, A_1\rangle, \langle L_{1,1,2}, A_{1,1,2}\rangle = \langle R_2, A_2\rangle \\ m_{1,2} = 2 : \langle L_{1,2,1}, A_{1,2,1}\rangle = \langle R_3, A_3\rangle, \langle L_{1,2,2}, A_{1,2,2}\rangle = \langle R_4, A_4\rangle \end{cases}$$

- *An example of a policy written in* CNF *is given below:*

$$Pol_2 = (\langle R_1, A'_1\rangle \vee \langle R_2, A'_2\rangle) \wedge (\langle R_3, A'_3\rangle \vee \langle R_4, A'_4\rangle)$$

Given the CDNF*-based notation defined above, policy* Pol_2 *is such that*

$$\begin{cases} m = 2, m_1 = 2, m_2 = 2 \\ m_{1,1} = 1 : \langle L_{1,1,1}, A_{1,1,1}\rangle = \langle R_1, A'_1\rangle \\ m_{1,2} = 1 : \langle L_{1,1,2}, A_{1,1,2}\rangle = \langle R_2, A'_2\rangle \\ m_{2,1} = 1 : \langle L_{1,2,1}, A_{1,2,1}\rangle = \langle R_3, A'_3\rangle \\ m_{2,2} = 1 : \langle L_{1,2,2}, A_{1,2,2}\rangle = \langle R_4, A'_4\rangle \end{cases}$$

The qualified sets of credentials for policies Pol_1 *and* Pol_2 *are given below:*

$$\begin{cases} \varsigma_1(Pol_1) = \{\varsigma(R_1, A_1), \varsigma(R_2, A_2)\} \\ \varsigma_2(Pol_1) = \{\varsigma(R_3, A_3), \varsigma(R_4, A_4)\} \end{cases} \qquad \begin{cases} \varsigma_{1,1}(Pol_2) = \{\varsigma(R_1, A'_1), \varsigma(R_3, A'_3)\} \\ \varsigma_{1,2}(Pol_2) = \{\varsigma(R_1, A'_1), \varsigma(R_4, A'_4)\} \\ \varsigma_{2,1}(Pol_2) = \{\varsigma(R_2, A'_2), \varsigma(R_3, A'_3)\} \\ \varsigma_{2,2}(Pol_2) = \{\varsigma(R_2, A'_2), \varsigma(R_4, A'_4)\} \end{cases}$$

Now that we have defined our policy-model, we define in the following the policy-based encryption primitive and the related security model.

1.3.2 Policy-Based Encryption

A formal definition of the policy-based encryption primitive is given below:

Definition 1.1 *A **policy-based encryption scheme**, denoted by* POLBE*, is specified by five algorithms:* Setup, Issuer-Setup, CredGen, Encrypt *and* Decrypt*, which we describe below.*

- **Setup**. *On input of a security parameter k, this algorithm generates a set of public parameters $\mathcal{P}$ which specifies the different parameters, groups and public functions that will be referenced by subsequent algorithms. Furthermore, it specifies a message space $\mathcal{M}$ and a ciphertext space $\mathcal{C}$.*

- **Issuer-Setup**. *This algorithm generates a random master key s_κ and the corresponding public key R_κ for credential issuer $I_\kappa \in \mathcal{I}$.*

- **CredGen**. *On input of the public key R_κ of a credential issuer $I_\kappa \in \mathcal{I}$ and an assertion $A \in \{0,1\}^*$, this algorithm returns the credential $\varsigma(R_\kappa, A)$.*

- **Encrypt**. *On input of a message $M \in \mathcal{M}$ and a policy Pol, this algorithm returns a ciphertext $C \in \mathcal{C}$ representing the encryption of M with respect to policy Pol.*

- **Decrypt**. *On input of a ciphertext $C \in \mathcal{C}$ and a set of credentials ρ, this algorithm returns either a message $M \in \mathcal{M}$ or $\perp$ (for 'error').*

Remark 1.3 *In the* Decrypt *algorithm defined above, whenever the set of credentials ρ is such that $\rho \nvDash Pol$, the output of the algorithm is $\perp$. To avoid this trivial case, we consider, from now on, that the set of credentials ρ is such that $\rho \vDash Pol$. In other words, algorithm* Decrypt *takes as input, in addition to the ciphertext C, a qualified set of credentials $\varsigma_{j_1,\ldots,j_m}(Pol)$, for some set of indices $\{j_i \in \{1,\ldots,m_i\}\}_{i=1}^m$.*

A POLBE scheme has to satisfy the standard consistency constraint i.e.

$$C = \mathsf{Encrypt}_{Pol}(M) \;\Rightarrow\; \mathsf{Decrypt}_{\varsigma_{j_1,\ldots,j_m}(Pol)}(C) = M, \text{ for some } \{j_i \in \{1,\ldots,m_i\}\}_{i=1}^m$$

Remark 1.4 *We let $\varphi_{j_1,\ldots,j_m}(C, Pol)$ be the information from the ciphertext C and the policy Pol that is required to correctly perform the decryption of C with respect to Pol using the qualified set of credentials $\varsigma_{j_1,\ldots,j_m}(Pol)$. A concrete example of $\varphi_{j_1,\ldots,j_m}(C, Pol)$ will be given when describing our pairing-based* POLBE *scheme. Basically, the information $\varphi_{j_1,\ldots,j_m}(C, Pol)$ will be referenced in our definition of the security model associated to* POLBE *schemes.*

1.3.3 Security Model

As discussed in Section 0.2.3, the standard acceptable notion of security for PKE schemes is indistinguishability against chosen ciphertext attacks, which is defined through the IND-CCA security model. Hence, it is natural to require that a POLBE scheme also satisfies this strong notion of security. However, the corresponding security model needs to be strengthened according to the particular setting of policy-based encryption. The reason is that when an adversary attacks a message encrypted using a POLBE scheme, the adversary might already know the policy according to which the message was encrypted. Furthermore, it might already possess any set of credentials provided the fact that it does not fulfill the policy according to which the message was encrypted. In other words, the adapted security model should allow an adversary to obtain any qualified set of credentials for any policy of his choice, other than the policy which it is challenged on. Furthermore, the adversary should be allowed to specify the policy it wishes to be challenged on.

Therefore, indistinguishability against chosen ciphertext attacks for POLBE schemes is defined in terms of an interactive game, denoted by IND-Pol-CCA, played between a challenger and an adversary. A formal definition of the IND-Pol-CCA game is given below.

Definition 1.2 *The* **IND-Pol-CCA *game*** *consists of five stages:* Setup, Phase-1, Challenge, Phase-2 *and* Guess *which we describe below.*

- **Setup**. *On input of a security parameter k, the challenger first runs algorithm* Setup *to obtain the system public parameters* $\mathcal{P}$. *Then, the challenger runs algorithm* Issuer-Setup *N times to obtain a set of credential issuers* $\mathcal{I} = \{I_1, \ldots, I_N\}$. *Finally, the challenger gives to the adversary the public parameters* $\mathcal{P}$ *as well as the public keys of the different credential issuers included in* $\mathcal{I}$.
- **Phase-1**. *The adversary performs a polynomial number of oracle queries adaptively i.e. each query may depend on the replies to the previously performed queries.*
- **Challenge**. *Once the adversary decides that* Phase-1 *is over, it gives to the challenger two equal length messages* M_0, M_1 *and a policy* Pol_{ch} *on which it wishes to be challenged. The challenger picks at random* $b \in \{0,1\}$, *then runs algorithm* Encrypt *on input of the tuple* (M_b, Pol_{ch}), *and finally returns the resulting ciphertext* C_{ch} *to the adversary.*
- **Phase-2**. *The adversary performs again a polynomial number of adaptive oracle queries.*
- **Guess**. *The adversary outputs a guess* b', *and wins the game if* $b = b'$.

During Phase-1 *and* Phase-2, *the adversary may perform queries to two oracles controlled by the challenger. On one hand, a credential generation oracle denoted by* CredGen-O. *On the other hand, a policy-base decryption oracle denoted by* Decrypt-O. *While the oracles are executed by the challenger, their input is specified by the adversary. The two oracles are defined as follows:*

- **CredGen-O**. *On input of a credential $I_\kappa \in \mathcal{I}$ and an assertion $A \in \{0,1\}^*$, run algorithm* CredGen *on input of the tuple (I_κ, A) and return the resulting credential $\varsigma(R_\kappa, A)$.*
- **Decrypt-O**. *On input of $C \in \mathcal{C}$, a policy Pol and a set of indices $\{j_1, \ldots, j_m\}$, run algorithm* CredGen *multiple times to obtain the qualified set of credentials $\varsigma_{j_1,\ldots,j_m}(Pol)$, then run algorithm* Decrypt *on input of the tuple $(C, Pol, \varsigma_{j_1,\ldots,j_m}(Pol))$ and return the resulting output.*

The oracle queries made by the adversary during Phase-1 *and* Phase-2 *are subject to two restrictions:*

1. *The adversary is not allowed to obtain a qualified set of credentials for policy Pol_{ch}.*
2. *The adversary is not allowed to perform a query on a tuple $(C, Pol, \{j_1, \ldots, j_m\})$ to oracle* Decrypt-O *such that $\varphi_{j_1,\ldots,j_m}(C, Pol) = \varphi_{j_1,\ldots,j_m}(C_{ch}, Pol_{ch})$. In fact, in the policy-based setting, for an encrypted message with respect to a policy with disjunctions, there is more than one possible qualified set of credentials that can be used to perform the decryption. That is, forbidding the adversary from making decryption queries on the challenge tuple (C_{ch}, Pol_{ch}), as in the* IND-ID-CCA *model, is not sufficient anymore. Indeed, we may have tuples such that $(C, Pol) \neq (C_{ch}, Pol_{ch})$ while $\varphi_{j_1,\ldots,j_m}(C, Pol) = \varphi_{j_1,\ldots,j_m}(C_{ch}, Pol_{ch})$. Decryption queries on such tuples should then be forbidden as well.*

In the following, we provide a formal definition for IND-Pol-CCA secure POLBE schemes.

Definition 1.3 *The advantage of an adversary $\mathcal{A}$ in the* IND-Pol-CCA *game is defined to be the quantity $Adv_{\mathcal{A}} = |Pr[b = b'] - \frac{1}{2}|$. A* POLBE *scheme is* IND-Pol-CCA *secure if no probabilistic polynomial time adversary has a non-negligible advantage in the* IND-Pol-CCA *game.*

Now that we have formally defined policy-based encryption and the related security model, we propose in the following an elegant and relatively efficient policy-based encryption scheme the security of which is proved in the random oracle model.

1.4 A Pairing-Based Implementation

1.4.1 Description

Our POLBE scheme consists of the algorithms described below:

- **Setup**. On input of the security parameter *k*, do the following:

1. Run algorithm BDH-Setup (Definition 0.33) to obtain a tuple $(q, \mathbb{G}_1, \mathbb{G}_T, e, P)$
2. Let $\mathcal{M} = \{0,1\}^{n-n_0}$ and $\mathcal{C} = \mathbb{G}_1 \times (\{0,1\}^n)^*$ (for some $n, n_0 \in \mathbb{N}^*$ such that $n_0 \ll n$)
3. Define three hash functions: $H_0 : \{0,1\}^* \to \mathbb{G}_1$, $H_1 : \{0,1\}^* \to \mathbb{Z}_q^*$ and $H_2 : \{0,1\}^* \to \{0,1\}^n$
4. Let $\mathcal{P} = (q, \mathbb{G}_1, \mathbb{G}_T, e, P, n, n_0, H_0, H_1, H_2)$

- **Issuer-Setup**. Let $\mathcal{I} = \{I_1, \ldots, I_N\}$ be a set of credential issuers. Each credential issuer $I_\kappa \in \mathcal{I}$ picks at random a secret master key $s_\kappa \in \mathbb{Z}_q^*$ and publishes the corresponding public key $R_\kappa = s_\kappa \cdot P$.

- **CredGen**. On input of the public key R_κ of issuer $I_\kappa \in \mathcal{I}$ and assertion $A \in \{0,1\}^*$, this algorithm outputs the credential $\varsigma(R_\kappa, A) = s_\kappa \cdot H_0(A)$.

- **Encrypt**. On input of message $M \in \mathcal{M}$ and policy Pol, do the following:

  1. Pick at random $M_i \in \{0,1\}^{n-n_0}$ (for $i = 1, \ldots, m-1$), then set $M_m = M \oplus (\oplus_{i=1}^{m-1} M_i)$
  2. Pick at random $t_i \in \{0,1\}^{n_0}$ (for $i = 1, \ldots, m$)
  3. Compute $r = H_1(M_1 \| \ldots \| M_m \| t_1 \| \ldots \| t_m)$, then compute $U = r \cdot P$
  4. For $j = 1, \ldots, m_i$ and $i = 1, \ldots, m$, compute $\pi_{i,j} = \prod_{k=1}^{m_{i,j}} e(R_{\kappa_{i,j,k}}, H_0(A_{i,j,k}))$, then compute $\mu_{i,j} = H_2(\pi_{i,j}^r \| i \| j)$, and finally compute $v_{i,j} = (M_i \| t_i) \oplus \mu_{i,j}$
  5. Return $C = (U, [[v_{i,j}]_{j=1}^{m_i}]_{i=1}^m)$

- **Decrypt**. On input of ciphertext $C = (U, [[v_{i,j}]_{j=1}^{m_i}]_{i=1}^m)$ and the qualified set of credentials $\varsigma_{j_1,\ldots,j_m}(Pol)$, do the following:

  1. Compute $\tilde{\pi}_{i,j_i} = e(U, \sum_{k=1}^{m_{i,j_i}} \varsigma(R_{\kappa_{i,j_i,k}}, A_{i,j_i,k}))$ (for $i = 1, \ldots, m$)
  2. Compute $\tilde{\mu}_{i,j_i} = H_2(\tilde{\pi}_{i,j_i} \| i \| j_i)$, then compute $(M_i \| t_i) = v_{i,j_i} \oplus \tilde{\mu}_{i,j_i}$ (for $i = 1, \ldots, m$)
  3. Compute $r = H_1(M_1 \| \ldots \| M_m \| t_1 \| \ldots \| t_m)$
  4. If $U = r \cdot P$, then return the message $M = \oplus_{i=1}^m M_i$, otherwise return $\perp$

Remark 1.5 *The logic behind our encryption algorithm is based on the following arguments:*

1. *Each conjunction of conditions $\wedge_{k=1}^{m_{i,j}} \langle I_{\kappa_{i,j,k}}, A_{i,j,k} \rangle$ is first associated to a mask $\mu_{i,j}$ that depends on the different credentials related to the specified conditions.*
2. *The encrypted message M is split into m random shares $[M_i]_{i=1}^m$, then for each index $i \in \{1, \ldots, m\}$, the value $M_i \| t_i$ is associated to the disjunction $\vee_{j=1}^{m_i} \wedge_{k=1}^{m_{i,j}} \langle I_{\kappa_{i,j,k}}, A_{i,j,k} \rangle$, where t_i is a randomly chosen intermediate key.*
3. *Each value $M_i \| t_i$ is encrypted m_i times using each of the masks $\mu_{i,j}$. This way, it is sufficient to compute any one of the masks $\mu_{i,j}$ in order to be able to retrieve $M_i \| t_i$. In order to be able to retrieve the encrypted message, an entity needs to retrieve all the shares as well as all the intermediate keys t_i using a qualified set of credentials for policy Pol.*

Remark 1.6 *Our* POLBE *scheme is such that $\varphi_{j_1,\ldots,j_m}(C = (U, [[v_{i,j}]_{j=1}^{m_i}]_{i=1}^m), Pol)$ consists of the value U and the pairs $\{(v_{i,j_i}, \wedge_{k=1}^{m_{i,j_i}} \langle I_{\kappa_{i,j_i,k}}, A_{i,j_i,k} \rangle)\}_{i=1}^m$.*

1.4.2 Consistency and Efficiency

Our POLBE scheme satisfies the standard consistency constraint. In fact, thanks to the bilinearity property of bilinear pairings, the following holds:

$$\tilde{\pi}_{i,j_i} = e(r \cdot P, \sum_{k=1}^{m_{i,j_i}} s_{\kappa_{i,j_i,k}} \cdot H_0(A_{i,j_i,k})) = \prod_{k=1}^{m_{i,j_i}} e(s_{\kappa_{i,j_i,k}} \cdot P, H_0(A_{i,j_i,k}))^r = \pi_{i,j_i}^r$$

In table 1.1, we summarize the computational costs (in the worst case) of our POLBE scheme. We consider the computational costs of our algorithms in terms of the following notations: **pa** the pairing, $\mathbf{ad}_1$ the addition in the group $\mathbb{G}_1$, $\mathbf{mu}_1$ the scalar multiplication in the group $\mathbb{G}_1$, $\mathbf{mu}_T$ the multiplication in the group $\mathbb{G}_T$, $\mathbf{exp}_T$ the exponentiation in the group $\mathbb{G}_T$. Note that we ignore the costs of hash computations.

Table 1.1: Computational costs of our POLBE scheme

Encrypt	$(\sum_{i=1}^{m}\sum_{j=1}^{m_i} m_{i,j}).\mathbf{pa} + \mathbf{mu}_1$ $+(\sum_{i=1}^{m}\sum_{j=1}^{m_i}(m_{i,j}-1)).\mathbf{mu}_T + (\sum_{i=1}^{m} m_i).\mathbf{exp}_T$
Decrypt	$m.\mathbf{pa} + (\sum_{i=1}^{m}(\max(m_{i,j})-1)).\mathbf{ad}_1 + (m-1).\mathbf{mu}_1$

According to Table 1.1, our encryption algorithm requires as many pairing computations as the number of conditions specified by the policy according to which the encryption is performed. Although such operation can be optimized, as explained in [21, 66], it still has to be minimized. Observe that for all i, j, k, the pairing $e(R_{\kappa_{i,j,k}}, H_0(A_{i,j,k}))$ used in the Encrypt algorithm does not depend on the encrypted message. It can thus be pre-computed, cached and used in subsequent encryptions involving the condition $\langle I_{\kappa_{i,j,k}}, A_{i,j,k}\rangle$.

Let l_1 be the bit-length of an encoding of an element of $\mathbb{G}_1$, then the bit-length of a ciphertext produced by our POLBE scheme is equal to: $l_1 + (\sum_{i=1}^{m} m_i).n$. Note that the size of the ciphertexts does not depend on the number of the right-hand AND operator of the CDNF form according to which our policies are formalized i.e. the ciphertext size does not depend on the $m_{i,j}$ values as compared with the solution proposed in [42].

In Table 2.2, we compare the performance of our scheme with the one of the scheme proposed in [42]. Our comparison is based on two factors: pairing computations and ciphertext size. While the two encryption algorithms require the same amount of pairing computations, our decryption algorithm requires less pairing computations because $m_{i,j_i} \geq 1$ for $i = 1, \ldots, m$. Furthermore, our scheme leads to ciphertexts more compact than the ones given by the scheme of [42] thanks to the fact that our policies are written in standard normal forms, whereas the ones considered in [42] are written as general Boolean expressions.

Remark 1.7 *As for standard asymmetric encryption schemes, policy-based encryption schemes are much less efficient than symmetric encryption schemes. In practice, they should be used to*

Table 1.2: Performance of our scheme compared with the scheme proposed in [42]

	Encryption	Decryption	Ciphertext Size
Our scheme	$\sum_{i=1}^{m}\sum_{j=1}^{m_i} m_{i,j}$	m	$l_1+(\sum_{i=1}^{m} m_i).n$
The scheme of [42]	$\sum_{i=1}^{m}\sum_{j=1}^{m_i} m_{i,j}$	$\sum_{i=1}^{m} m_{i,j_i}$	$l_1+(\sum_{i=1}^{m}\sum_{j=1}^{m_i} m_{i,j}).n+n$

exchange the symmetric (session) keys that are used for bulk encryption. An illustration of this approach is given in Section 1.5.

1.4.3 Security

In the following, we study the security of our POLBE scheme.

Theorem 1.1 *Our* POLBE *scheme is* IND-Pol-CCA *secure in the random oracle model under the assumption that BDHP is hard.*

Proof As depicted in Figure 1.3, Theorem 1.1 follows from two reduction arguments:

1. Lemma 1.1 shows that an IND-Pol-CCA attack on our POLBE scheme can be converted into an IND-CCA attack on the BasicPub$^{hy_{II}}$ scheme (Definition 0.37).
2. Algorithm BasicPub$^{hy_{II}}$ is shown to be IND-CCA secure in the random oracle model under the assumption that BDHP is hard (See Section 0.3.5 for more details).

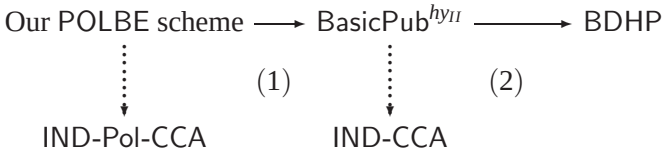


Figure 1.3: Reductionist Security for our POLBE Scheme

□

Lemma 1.1 *Let $\mathcal{A}^{\circ}$ be an* IND-Pol-CCA *adversary with advantage $Adv_{\mathcal{A}^{\circ}} \geq \varepsilon$ when attacking our* POLBE *scheme. Assume that $\mathcal{A}^{\circ}$ has running time $t_{\mathcal{A}^{\circ}}$ and makes at most q_c queries to oracle* CredGen-O*, q_d queries to oracle* Decrypt-O*, and q_0 queries to oracle H_0. Then, there exists an* IND-CCA *adversary $\mathcal{A}^{\bullet}$ the advantage of which, when attacking the* BasicPub^{hy_I} *scheme, is such that $Adv_{\mathcal{A}^{\bullet}} \geq \Psi(q_c,q_d,q_0,N,m_{\vee\wedge},m_{\vee},m_{\wedge}).\varepsilon$. Its running time is $t_{\mathcal{A}^{\bullet}} = O(t_{\mathcal{A}^{\circ}})$.*

Proof Let $\mathcal{A}^\circ$ be an IND-Pol-CCA adversary with advantage $\mathrm{Adv}_{\mathcal{A}^\circ} \geq \varepsilon$ when attacking our POLBE scheme. Assume that $\mathcal{A}^\circ$ has running time $t_{\mathcal{A}^\circ}$ and makes at most q_c queries to oracle CredGen-O, q_d queries to oracle Decrypt-O, and q_0 queries to oracle H_0. In the following, we construct an IND-CCA adversary $\mathcal{A}^\bullet$ that uses adversary $\mathcal{A}^\circ$ to mount an attack against the BasicPub^{hy_I} scheme.

The IND-CCA game, played between the challenger and algorithm $\mathcal{A}^\bullet$, starts with the Setup$^\bullet$ stage described below.

- **Setup$^\bullet$**. Given the security parameter k, the challenger does the following:
  1. Run the Setup algorithm of the BasicPub$^{hy_{II}}$ scheme, which generates the public parameters $\mathcal{P}^\star = (q, \mathbb{G}_1, \mathbb{G}_T, e, P, m^\star.n, m^\star.n_0, H_1, H_2)$, for some $m^\star \in \mathbb{N}^*$
  2. Run the KeyGen algorithm of the BasicPub$^{hy_{II}}$ scheme, which generates a private key $sk^\star \in \mathbb{Z}_q^*$ and the corresponding public key $pk^\star = (R^\star = sk^\star \cdot P, Q^\star)$
  3. Give the public parameters $\mathcal{P}^\star$ and the public key $pk^\star$ to adversary $\mathcal{A}^\bullet$, while keeping secret the private key $sk^\star$.

Before interacting with adversary $\mathcal{A}^\circ$, adversary $\mathcal{A}^\bullet$ does the following:

1. Let $m^\bullet = m^\star$, then choose $m_i^\bullet \in \{1,\ldots,m_\vee\}$ (for $i = 1,\ldots,m^\bullet$) and $m_{i,j}^\bullet \in \{1,\ldots,m_\wedge\}$ (for $i = 1,\ldots,m^\bullet$ and $j = 1,\ldots,m_i^\bullet$). In the rest of this proof, unless specified explicitly, the indices i, j, k are such that $i = 1,\ldots,m^\bullet$, $j = 1,\ldots,m_i^\bullet$ and $k = 1,\ldots,m_{i,j}^\bullet$.
2. Pick at random $\alpha_i^\bullet \in \{1,\ldots,m_{\vee\wedge}\}$ and $r_{\alpha_i^\bullet}, \omega_i^\bullet \in \mathbb{Z}_q^*$
3. Pick at random $\beta_{i,j}^\bullet \in \{1,\ldots,m_\vee\}$ and $r_{\beta_{i,j}^\bullet}, \upsilon_{i,j}^\bullet \in \mathbb{Z}_q^*$
4. Pick at random $l_{i,j,k}^\bullet \in \{1,\ldots,q_0\}$, $\gamma_{i,j,k}^\bullet \in \{1,\ldots,m_\wedge\}$ and $r_{\gamma_{i,j,k}^\bullet} \in \mathbb{Z}_q^*$
5. Pick at random $\theta_{i,j,k}^\bullet \in \mathbb{Z}_q^*$ (for $k \neq 1$), then compute $\theta_{i,j,1}^\bullet = \sum_{k=2}^{m_{i,j}^\bullet} \theta_{i,j,k}^\bullet$
6. Compute $\kappa_{i,j,k}^\bullet = ((\alpha_i^\bullet - 1).m_\vee + \beta_{i,j}^\bullet) - 1).m_\wedge + \gamma_{i,j,k}^\bullet$ and $r_{\kappa_{i,j,k}^\bullet} = r_{\gamma_{i,j,k}^\bullet} r_{\beta_{i,j}^\bullet} r_{\alpha_i^\bullet}$
7. Choose a hash function: $\bar{H}_2^\bullet : \{1,\ldots,m_{\vee\wedge}\} \to \{0,1\}^n$
8. Define the function $\Delta^\bullet : \{0,1\}^{m^\bullet.n} \times \{1\ldots,m^\bullet\} \to \{0,1\}^n$ which on input of (X,i) returns the i^{th} block of length n of the binary string X i.e. the bits from $(i-1).n+1$ to $i.n$ of X.
9. Define the functions $\xi_x^f : \{0,1\}^{>x} \to \{0,1\}^x$ and $\xi_x^l : \{0,1\}^{>x} \to \{0,1\}^x$ which on input a string X return, respectively, the first x bits of X and the last x bits of X.
10. Define the function $\Omega^\bullet : \{0,1\}^{m^\bullet.n} \times \{1\ldots,m^\bullet\} \to \{0,1\}^n$ which on input of a tuple (X,i) returns the binary $\Delta^\bullet(\xi^f_{m^\bullet.(n-n_0)}(X),i) \| \Delta^\bullet(\xi^l_{m^\bullet.n_0}(X),i)$.

Remark 1.8 *In our proof, we assume that adversaries $\mathcal{A}^\bullet$ and $\mathcal{A}^\circ$ are parameterized with the value $m^\star$. Besides, we assume that N, the number of available credential issuers, is such that $N \geq m_{\vee\wedge}m_\vee$. Our proof can be easily adapted to the case where $N \leq m_{\vee\wedge}m_\vee$.*

Remark 1.9 *Let $Pol_{cr} = \wedge_{i=1}^{m^\bullet} \vee_{j=1}^{m_i^\bullet} \wedge_{k=1}^{m_{i,j}^\bullet} \langle I_{\kappa_{i,j,k}^\bullet}, A_{l_{i,j,k}^\bullet} \rangle$. Policy Pol_{cr} is called the 'crucial' policy. Algorithm $\mathcal{A}^\bullet$ hopes that the 'target' policy Pol_{ch}, which will be chosen by adversary $\mathcal{A}^\circ$ in the* Challenge *stage of the* IND-Pol-CCA *game, is equal to policy Pol_{cr}.*

The interaction between algorithm $\mathcal{A}^\bullet$ (the challenger) and adversary $\mathcal{A}^\circ$ consists of five stages: Setup$^\circ$, Phase-1°, Challenge$^\circ$, Phase-2° and Guess$^\circ$, which we describe below.

- **Setup**$^\circ$. Algorithm $\mathcal{A}^\bullet$ does the following:
  1. Let $\mathcal{P}^\bullet = (q, \mathbb{G}_1, \mathbb{G}_T, e, P, n, n_0, H_0^\bullet, H_1, H_2^\bullet)$ be the public parameters, where the oracles $H_0^\bullet$ and $H_2^\bullet$ are controlled by algorithm $\mathcal{A}^\bullet$ and the tuple $(q, \mathbb{G}_1, \mathbb{G}_T, e, P, n, n_0, H_1)$ is taken from $\mathcal{P}^\star$. Algorithm $\mathcal{A}^\bullet$ controls $H_0^\bullet$ and $H_2^\bullet$ as follows:
     - For the random oracle $H_0^\bullet$, algorithm $\mathcal{A}^\bullet$ maintains a list of tuples $[A_\iota, H_{0,\iota}, \lambda_\iota]$ which we denote H_0^{list}. The list is initially empty. Assume that adversary $\mathcal{A}^\circ$ makes a query on assertion $A \in \{0,1\}^*$, then adversary $\mathcal{A}^\bullet$ responds as follows:
       (a) If A already appears on the list H_0^{list} in a tuple $[A_\iota, H_{0,\iota}, \lambda_\iota]$, then return $H_{0,\iota}$
       (b) If A does not appear on H_0^{list} and A is the $l_{i,j,1}^\bullet$-th distinct query to oracle $H_0^\bullet$, then compute $H_{0,l_{i,j,1}^\bullet} = r_{\gamma_{i,j,1}^\bullet}^{-1} \cdot ((r_{\beta_{i,j}^\bullet}^{-1} \upsilon_{i,j}^\bullet r_{\alpha_i^\bullet}^{-1} \omega_i^\bullet) \cdot Q^\star - \theta_{i,j,1}^\bullet \cdot P)$, return $H_{0,l_{i,j,1}^\bullet}$, and add $[A, H_{0,l_{i,j,1}^\bullet}, \mathsf{null}]$ to H_0^{list}
       (c) If A does not appear on H_0^{list} and A is the $l_{i,j,k}^\bullet$-th distinct query to oracle $H_0^\bullet$ (for $k > 1$), then compute $H_{0,l_{i,j,k}^\bullet} = (r_{\gamma_{i,j,k}^\bullet}^{-1} \theta_{i,j,k}^\bullet) \cdot P$, return $H_{0,l_{i,j,k}^\bullet}$, and add the entry $[A, H_{0,l_{i,j,k}^\bullet}, r_{\gamma_{i,j,k}^\bullet}^{-1} \theta_{i,j,k}^\bullet]$ to H_0^{list}
       (d) Otherwise, pick at random $\lambda \in \mathbb{Z}_q^*$ such that $\lambda \cdot P$ does not appear on the list H_0^{list}, return $\lambda \cdot P$, and add the entry $[A, \lambda \cdot P, \lambda]$ to H_0^{list}

       Note that the oracle $H_0^\bullet$ is such that $(r_{\beta_{i,j}^\bullet}^{-1} \upsilon_{i,j}^\bullet r_{\alpha_i^\bullet}^{-1} \omega_i^\bullet) \cdot Q^\star = \sum_{k=1}^{m_{i,j}^\bullet} r_{\gamma_{i,j,k}^\bullet} H_{0,l_{i,j,k}^\bullet}$.
     - For the random oracle $H_2^\bullet$, on input of a tuple (G, i, j), algorithm $\mathcal{A}^\bullet$ returns the quantity $\Omega^\bullet(H_2(G^{\upsilon_{i,j}^{\bullet\,-1} \omega_i^{\bullet\,-1}}) \oplus \bar{H}_2^\bullet(j), i)$.
  2. Define the set of credential issuers $\mathcal{I} = \{I_1, \ldots, I_N\}$ as follows:
     - For $\kappa \in \{\kappa_{i,j,k}^\bullet\}$, the public key of I_κ is $R_\kappa = r_\kappa \cdot R^\star = (r_\kappa sk^\star) \cdot P$
     - For $\kappa \in \{1, \ldots, N\} \setminus \{\kappa_{i,j,k}^\bullet\}$, the public key of I_κ is $R_\kappa = s_\kappa \cdot P$ for some randomly chosen $s_\kappa \in \mathbb{Z}_q^*$.
  3. Give the public parameters $\mathcal{P}^\bullet$ and the credential issuers' public keys $\{R_\kappa\}_{\kappa=1}^N$ to adversary $\mathcal{A}^\circ$.
- **Phase-1**$^\circ$. Adversary $\mathcal{A}^\circ$ performs a polynomial number of oracle queries adaptively.

- **Challenge**$^\circ$. Once adversary $\mathcal{A}^\circ$ decides that the stage Phase-1 is over, it outputs two equal length messages M_0 and M_1 as well as a policy Pol_{ch} on which it wishes to be challenged. Algorithm $\mathcal{A}^\bullet$ responds as follows:

  1. If $Pol_{\text{ch}} \neq Pol_{\text{cr}}$, then report failure and terminate (we refer to this event as $\mathcal{E}_{\text{ch}}$)
  2. Otherwise, for $d = 0,1$, pick at random $M_{d,i} \in \{0,1\}^{n-n_0}$ (for $i \neq m^\bullet - 1$) and set $M_{d,m^\bullet} = M_d \oplus (\oplus_{i=1}^{m^\bullet-1} M_{d,i})$. Then, give the messages $M_d^\bullet = M_{d,1} \| \ldots \| M_{d,m^\bullet}$ to the challenger who picks randomly $b \in \{0,1\}$ and returns a ciphertext $C^\star = (U, v^\star)$ representing the encryption of message $M_b^\bullet$ using the public key $\underline{pk}^\star$. Upon receiving the challenger's response, compute the values $v_{i,j} = \Omega^\bullet(v^\star \oplus H_2^\bullet(j), i)$, then return the ciphertext $C_{\text{ch}} = (U, [[v_{i,j}]_{j=1}^{m_i^\bullet}]_{i=1}^{m^\bullet})$.

- **Phase-2**$^\circ$. Adversary $\mathcal{A}^\circ$ performs a polynomial number of oracle queries adaptively.

- **Guess**$^\circ$. Algorithm $\mathcal{A}^\circ$ outputs a guess b' for b. Upon receiving b', algorithm $\mathcal{A}^\bullet$ outputs b' as its guess for b.

During Phase-1° and Phase-2°, adversary $\mathcal{A}^\circ$ can make the queries of its choice to two oracles, denoted by CredGen-O$^\circ$ and Decrypt-O$^\circ$, controlled by adversary $\mathcal{A}^\bullet$ as described below.

- **CredGen-O**$^\circ$. Assume that adversary $\mathcal{A}^\circ$ makes a query on a tuple (I_κ, A). Let $[A_\iota, H_{0,\iota}, \lambda_\iota]$ be the tuple from H_0^{list} such that $A_\iota = A$, then algorithm $\mathcal{A}^\bullet$ responds as follows:

  1. If $\iota = l_{i,j,1}^\bullet$ and $\kappa \in \{\kappa_{i,j,k}^\bullet\}_{i,j,k}$, then report failure and terminate (event $\mathcal{E}_{\text{cred}}$)
  2. If $\iota \neq l_{i,j,1}^\bullet$ and $\kappa \in \{\kappa_{i,j,k}^\bullet\}_{i,j,k}$, then return $(r_\kappa \lambda_\iota) \cdot R^\star = (r_\kappa s^\star) \cdot H_{0,\iota}$
  3. If $\kappa \in \{1, \ldots, N\} \setminus \{\kappa_{i,j,k}^\bullet\}_{i,j,k}$, then return $s_\kappa \cdot H_{0,\iota}$

- **Decrypt-O**$^\circ$. Assume that adversary $\mathcal{A}^\circ$ makes a query on tuple $(C, Pol, \{j_1, \ldots, j_m\})$, such that $\varphi_{j_1,\ldots,j_m}(C, Pol) = \varphi_{j_1,\ldots,j_m}(C_{\text{ch}}, Pol_{\text{ch}})$. Then, algorithm $\mathcal{A}^\bullet$ responds as follows:

  1. If $Pol \neq Pol_{\text{ch}}$ and Pol involves a condition $\langle I_\kappa, A \rangle$ such that $\kappa \in \{\kappa_{i,j,k}^\bullet\}$ and $A \in \{A_{l_{i,j,1}^\bullet}\}$, then report failure and terminate (event $\mathcal{E}_{\text{dec}}$)
  2. If $Pol \neq Pol_{\text{ch}}$ and Pol does not involve any condition $\langle I_\kappa, A \rangle$ such that $\kappa \in \{\kappa_{i,j,k}^\bullet\}$ and $A \in \{A_{l_{i,j,1}^\bullet}\}$, then do the following: (1) Run oracle CredGen-O$^\circ$ multiple times until obtaining the qualified set of credentials $\varsigma_{j_1,\ldots,j_m}(Pol)$, (2) Run algorithm Decrypt on input the tuple $(C, Pol, \varsigma_{j_1,\ldots,j_m}(Pol))$ and return the resulting output back to adversary $\mathcal{A}^\circ$
  3. If $Pol = Pol_{\text{ch}}$, then do the following: let $C = (U, [[v_{i,j}]_{j=1}^{m_i^\bullet}]_{i=1}^{m^\bullet})$, then compute the values $v_i^\bullet = v_{i,j_i} \oplus H_2^\bullet(j_i)$, and make a decryption query to the challenger on ciphertext $C^\bullet = (U, \xi_{n-n_0}^f(v_1^\bullet) \| \ldots \| \xi_{n-n_0}^f(v_{m^\bullet}^\bullet) \| \xi_{n_0}^l(v_1^\bullet) \| \ldots \| \xi_{n_0}^l(v_{m^\bullet}^\bullet))$. Upon receiving the challenger's response, forward it to adversary $\mathcal{A}^\circ$

Remark 1.10 *Without loss of generality, we assume that adversary $\mathcal{A}^\circ$ always makes the appropriate query on A to the random oracle $H_0^\bullet$ before making any query involving A to oracles* CredGen-O$^\circ$ *and* Decrypt-O$^\circ$.

We assume that adversary $\mathcal{A}^\circ$ respects the following restrictions when performing oracle queries during Phase-1° and Phase-2°:

1. It does not try to obtain a qualified set of credentials for policy Pol_{ch}
2. It does not make a query to oracle Decrypt-O$^\circ$ on a tuple $(C, Pol, \{j_1, \ldots, j_m\})$ such that $\varphi_{j_1,\ldots,j_m}(C, Pol) = \varphi_{j_1,\ldots,j_m}(C_{\text{ch}}, Pol_{\text{ch}})$. In other words, if $C = (U, [[v_{i,j}]_{j=1}^{m_i}]_{i=1}^{m})$ and $C_{\text{ch}} = (U^{\text{ch}}, [[v_{i,j}^{\text{ch}}]_{j=1}^{m_i}]_{i=1}^{m})$, then we should not have

$$\begin{cases} U = U^{\text{ch}} \\ \{(v_{i,j_i}, \wedge_{k=1}^{m_{i,j_i}} \langle I_{\kappa_{i,j_i,k}}, A_{i,j_i,k}\rangle) = (v_{i,j_i}^{\text{ch}}, \wedge_{k=1}^{m_{i,j_i}} \langle I_{\kappa_{i,j_i,k}}, A_{i,j_i,k}\rangle)\}_{i=1}^{m} \end{cases}$$

In the following, we analyze the simulation described above:

If algorithm $\mathcal{A}^\bullet$ does not report failure during the simulation, then the view of algorithm $\mathcal{A}^\circ$ is identical to its view in the real attack. In fact, observe first that the responses of algorithm $\mathcal{A}^\bullet$ to all queries of adversary $\mathcal{A}^\circ$ to oracle $H_0^\bullet$ are uniformly and independently distributed in group $\mathbb{G}_1$ as in the real IND-Pol-CCA attack. Second, all the responses of algorithm $\mathcal{A}^\bullet$ to queries made by adversary $\mathcal{A}^\circ$ to oracles CredGen-O$^\circ$ and Decrypt-O$^\circ$ are consistent. Third, the ciphertext C_{ch} given to adversary $\mathcal{A}^\circ$ corresponds to the encryption according to Pol_{ch} of M_b for some random $b \in \{0,1\}$, as shown in Remark 1.11.

Remark 1.11 *For adversary $\mathcal{A}^\circ$, C_{ch} is a well-formed ciphertext resulting from the encryption of message M_b with respect to policy Pol_{ch} according to our* POLBE *scheme. In fact, the ciphertext $C^\star$ is such that $U = H_1(M_b\|t) \cdot P$ (for some randomly chosen $t \in \{0,1\}^{m^\bullet \cdot n}$), and $v^\star = (M_b\|t) \oplus H_2(g^r)$ where $g = e(R^\star, Q^\star)$. Let $t_i = \Omega^\bullet(t,i)$, then the following holds*

$$\begin{aligned} v_{i,j} &= \Omega^\bullet((M_b\|t) \oplus H_2(e(R^\star, Q^\star)^r) \oplus \bar{H}_2^\bullet(j), i) \\ &= \Omega^\bullet(M_b\|t, i) \oplus \Omega^\bullet(H_2([e((r_{\beta_{i,j}^\bullet} r_{\alpha_i^\bullet}) \cdot R^\star, (r_{\beta_{i,j}^\bullet}^{-1} \upsilon_{i,j}^\bullet r_{\alpha_i^\bullet}^{-1} \omega_i^\bullet) \cdot Q^\star)^r]^{{\upsilon_{i,j}^\bullet}^{-1} {\omega_i^\bullet}^{-1}}) \oplus \bar{H}_2^\bullet(j), i) \\ &= (M_i\|t_i) \oplus H_2^\bullet(e((r_{\beta_{i,j}^\bullet} r_{\alpha_i^\bullet}) \cdot R^\star, \sum_{k=1}^{m_{i,j}^\bullet} r_{\gamma_{i,j,k}^\bullet} H_{0,l_{i,j,k}^\bullet})^r, i, j) \\ &= (M_i\|t_i) \oplus H_2^\bullet([\prod_{k=1}^{m_{i,j}^\bullet} e(R_{\kappa_{i,j,k}^\bullet}, H_{0,l_{i,j,k}^\bullet})]^r, i, j) \end{aligned}$$

Algorithm $\mathcal{A}^{\bullet}$ reports failure if either event $\mathcal{E}_{\mathsf{ch}}$, event $\mathcal{E}_{\mathsf{cred}}$ or event $\mathcal{E}_{\mathsf{dec}}$ occurs during the simulation. Since events $\mathcal{E}_{\mathsf{cred}}$ and $\mathcal{E}_{\mathsf{dec}}$ are independent, the following statement holds

$$\begin{aligned} \mathrm{Adv}_{\mathcal{A}^{\bullet}} &\geq Pr[\neg\mathcal{E}_{\mathsf{cred}} \wedge \neg\mathcal{E}_{\mathsf{ch}} \wedge \neg\mathcal{E}_{\mathsf{dec}}].\varepsilon \\ &\geq \underbrace{Pr[\neg\mathcal{E}_{\mathsf{ch}}|\neg\mathcal{E}_{\mathsf{cred}} \wedge \neg\mathcal{E}_{\mathsf{dec}}].Pr[\neg\mathcal{E}_{\mathsf{cred}}].Pr[\neg\mathcal{E}_{\mathsf{dec}}]}_{\Psi(q_c,q_d,q_0,N,m_{\vee\wedge},m_{\vee},m_{\wedge})}.\varepsilon \end{aligned} \tag{1.2}$$

From the simulation described above, we have

$$Pr[\mathcal{E}_{\mathsf{cred}}] \leq \frac{q_c m_{\vee\wedge} m_{\vee}}{N q_0} \tag{1.3}$$

Adversary $\mathcal{A}^{\bullet}$ picks the challenge policy from a set of $\Upsilon(Nq_0, m_{\vee\wedge}, m_{\vee}, m_{\wedge})$ distinct policies. Then, the following statement holds

$$Pr[\neg\mathcal{E}_{\mathsf{ch}}|\neg\mathcal{E}_{\mathsf{cred}} \wedge \neg\mathcal{E}_{\mathsf{dec}}] \geq \frac{1}{\Upsilon(Nq_0, m_{\vee\wedge}, m_{\vee}, m_{\wedge})} \tag{1.4}$$

The total number of policies, distinct from policy Pol_{ch}, that may be specified by adversary $\mathcal{A}^{\bullet}$ during queries to oracle Decrypt-O$^{\circ}$, and that involve at least one of the conditions $\langle I_{\kappa}, A\rangle$ such that $\kappa \in \{\kappa^{\bullet}_{i,j,k}\}$ and $A \in \{A_{l^{\bullet}_{i,j,1}}\}$ could be upper bounded by the function

$$\Upsilon'(Nq_0, m_{\vee\wedge}, m_{\vee}, m_{\wedge}) = \Upsilon(Nq_0, m_{\vee\wedge}, m_{\vee}, m_{\wedge}) - \Upsilon(Nq_0 - (m_{\vee\wedge}m_{\vee})^2, m_{\vee\wedge}, m_{\vee}, m_{\wedge}) - 1 \tag{1.5}$$

Then, the following statement holds

$$Pr[\mathcal{E}_{\mathsf{dec}}] \leq \frac{q_d \Upsilon'(Nq_0, m_{\vee\wedge}, m_{\vee}, m_{\wedge})}{\Upsilon(Nq_0, m_{\vee\wedge}, m_{\vee}, m_{\wedge})} \tag{1.6}$$

Finally, statements (1.2), (1.3), (1.4) and (1.6) lead to the function $\Psi(q_c, q_d, q_0, N, m_{\vee\wedge}, m_{\vee}, m_{\wedge})$.

The function $\Psi(.)$ has a rather unaesthetic expression. Computing $\Psi(q_c, q_d, q_0, N, m_{\vee\wedge}, m_{\vee}, m_{\wedge})$ relies on computing the quantity $\Upsilon(X, m_{\vee\wedge}, m_{\vee}, m_{\wedge})$, which is defined to be the total number of 'minimal' policies written in CDNF, given the upper-bounds $(m_{\vee\wedge}, m_{\vee}, m_{\wedge})$ and X possible credential-based conditions. Computing $\Upsilon(X, m_{\vee\wedge}, m_{\vee}, m_{\wedge})$ is similar, but not exactly the same as the problems of computing the number of monotone Boolean functions of n variables (Dedekind's Problem [100]) and computing the number of antichains on a set $\{1, \ldots, n\}$ [97]. As opposed to these problems, the order of the terms must be taken into consideration when dealing with our policies. This is a typical, yet interesting, 'counting' problem. However, as we do not address exact security in this thesis, we do not elaborate more on the details.

Remark 1.12 *In the case where* $N = m_{\vee\wedge} = m_{\vee} = m_{\wedge} = 1$*, we have* $\Upsilon'(Nq_0, m_{\vee\wedge}, m_{\vee}, m_{\wedge}) = 0$ *and* $\Upsilon(Nq_0, m_{\vee\wedge}, m_{\vee}, m_{\wedge}) = q_0$*. In this case, the security of our* POLBE *scheme is equivalent to the security of the* FullIdent *scheme of [38]. Note that our results match Result 5 of [81]. In fact, our reductionist security proof follows a strategy that is similar but not exactly the same as the one proposed by Galindo in [81].*

□

Now that we have formalized the concept of policy-based encryption and proposed a provably secure policy-based encryption scheme using bilinear pairings over elliptic curves, we present in the remaining sections three possible applications of our encryption primitive: the first application is in the context of classical access control, the second application is in the context of privacy policy enforcement, while the third application is in the context of ad-hoc community establishment.

1.5 Controlling Access to Released XML Documents

The most intuitive application of the policy-based encryption primitive is in the context of access control. In this section, we present a comprehensive framework for controlling access to released XML documents over the Internet. The choice of the eXtensible Markup Language (XML) is justified by the fact that XML is playing a crucial role in today's web-based applications, since it is rapidly becoming a de facto standard for document representation and exchange. Some of the well-recognized benefits of using the XML language as data container are its simplicity, openness and extensibility as well as its endorsement by almost all software industry market leaders. We refer the reader to [129] for more details about XML and its benefits.

An increasing number of XML documents released over the Internet contain information of different sensitivity degrees that is to be shared by a large number of users (data consumers). Consequently, it is necessary for the releasing entity (data owner) to put in place a suitable protection for each document before its release. A protection consists of a set of authorization policies each of which defines the conditions under which a data consumer is allowed to have access to a specific portion of the document. Once a protection is defined for a specific document, it must be implemented by an effective and efficient protection enforcement mechanism.

A protection for an XML document has to fulfill three basic requirements that have been widely discussed in the literature. In the following, we briefly discuss each of these requirements:

1. A protection has to cope with a **large and dynamic population of data consumers**. The different policies specified by the protection should therefore be based on assertion-based authorization rather than on identity-based authorization. In other words, access to the different portions of a given document should be defined with respect to the attributes, properties, capabilities of data consumers, which are more relevant than their identities.

2. A protection has to support **content-based access control**, which allows one to express authorization conditions that take document contents into account. Instead of being judged according to their compliance with specific assertions, data consumers are judged with respect to their knowledge of certain data values. Also referred to as knowledge-based authorization, content-based access control offers more flexibility to the protection associated to an XML document [118].

3. A protection has to support authorization policies with **different granularity levels**. Indeed, in some cases, the same policy may apply to the entire document. In other cases, different authorization policies may apply to different components of the same document. Given an XML document, the granularity of a policy can be limited, for instance, to a specific element and its sub-elements.

There are mainly three different approaches in the literature to the enforcement of a protection associated to an XML document to be released over the Internet:

- **centralized access control**: a protection can be enforced by keeping the document on a secure server without publishing it. The server is responsible for checking the compliance of each data consumer with the authorization policy associated to the portion of the document the data consumer wants to have access to. This approach corresponds to standard access control and does not fit very well with the characteristics of data-intensive web-based applications.

- **multiple views**: because data consumers can be authorized to have access to different, selected portions of the same document, a protection can be enforced by publishing multiple views of the document, one for each data consumer or class of data consumers. This is the current approach for controlling access to published XML documents. This approach suffers from at least two drawbacks: 1) the number of views may become very large and even proportional to the total number of data consumers, 2) there might exist information duplication between the different released views, which results in a network overload.

- **cryptography-based access control**: to avoid information duplication, an increasingly popular approach for protection enforcement consists in using cryptography. The basic idea is that the data owner, instead of releasing multiple views of the document in cleartext, it releases a single ciphertext version resulting from the encryption of the different portions of the document with respect to the associated authorization policies. The encryption is performed in such a way that a data consumer is able to only decrypt the portions of the document to which it is authorized to have access.

Henceforth, our discussion focuses on the cryptography-based enforcement mechanism.

A cryptography-based framework for controlling access to XML documents typically addresses the two aspects discussed below:

- **protection specification**: for each portion of a given XML document, the data owner needs to specify a policy. The policy specifies the conditions under which a data consumer is authorized to have access to the considered portion of the document. The specification should be done according to a protection model defining the syntax and semantics of the policies to be used (policy model) and the granularity levels at which they are applied. Naturally, the adopted protection model needs to be understandable by data consumers.

- **protection enforcement**: once a protection is specified for a given document, the data owner needs to apply a provably secure mechanism for encrypting the document and managing the encryption keys so that a data consumer is able to decrypt a portion of the document if and only if it is compliant with the associated authorization policy. The performance of the enforcement mechanism depends on the efficiency of the used encryption algorithm and essentially on the number of keys used to encrypt the different portions of the document. Its security relies on the security of the encryption algorithm and on the effectiveness of the key distribution strategy.

As mentioned before, controlling access to XML documents using cryptography is an increasingly popular approach. In this line of research, several solutions are proposed in the literature, each of which addresses one or different aspects of protection specification and enforcement [103, 119, 60, 33, 41, 80]. One of the most relevant solutions is the one recently proposed by Miklau et al. in [119]. In their paper, the authors propose a comprehensive framework for cryptography-based access control on published XML documents. In the following, we briefly discuss the main features of their framework:

- **protection specification**: Miklau et al. consider the tree representation of an XML document. They define a protection to be a tree where each node is associated to a specific authorization policy. The latter specifies the conditions under which a data consumer is allowed to have access to the node and its sub-nodes. The authors define a query-based language for specifying the authorization policies to be associated to the different tree nodes. Their policy model considers two types of authorization rules: on one hand, authorization rules whose validity is directly verified by the data owner and, on the other hand, authorization rules whose validity is constrained by the knowledge of the data consumer of an information contained in the document. A query engine allows to convert their policies into guard formulae, which are policies that consist of conjunctions and disjunctions of conditions where each condition is fulfilled by a specific symmetric key. A data consumer should be able to have access to a specific node (and sub-nodes) if and only if it is compliant with the guard formula associated to the node i.e. if and only if the data consumer has access to a set of keys fulfilling the combination of key-based conditions specified by the guard formula. The considered keys are either issued by the data owner after checking the compliance of the data consumer with a set of authorization rules or directly derived from the content of the document.

- **protection enforcement**: the enforcement mechanism proposed by Miklau et al. consists of three stages. First, a set of normalization and optimization re-write rules are used to

transform the protection associated to an XML document into an equivalent protection that consists of atomic guard formulae. An atomic guard formula consists of a single condition fulfilled by a single key. This transformation stage implies the extension of the original document by adding a set of metadata nodes. Second, each node (and sub-nodes) is encrypted using the key fulfilling the associated guard formula. The normalization stage is in fact necessary because the standard symmetric-key encryption schemes used by Miklau et al. do not allow to handle the conjunctions and disjunctions of keys. Third, the different keys used to encrypt the document are released by the data owner in a way such that a data consumer obtains only the keys that allow it to decrypt the portions of the document corresponding to the policies it is compliant with.

The key distribution strategy proposed by Miklau et al. is such that the data owner is the only entity who is responsible for carefully distributing the keys used to encrypt a released XML document. The main advantage of this strategy is that the data owner has a full control on the data consumers who want to access his documents. However, this key distribution method suffers from the two shortcomings discussed below:

- a centralized key distribution strategy is cumbersome to the data owner, especially with a large number of data consumers. In fact, before releasing the decryption keys, the data owner has often to check the validity of the credentials of each data consumer and verify his compliance with the policies associated to the portions of the documents to which it wants to have access. It would be interesting for the data owner to delegate all or part of the management of decryption keys to trusted third parties. Once an encrypted document is released, the data owner would not have to deal with data consumers, while being sure that his protection will be enforced.
- as a consequence of the centralized key distribution strategy, the data owner knows exactly who wants to have access to its documents and what policies it fulfills. The data owner knows even what specific conditions of a given policy are fulfilled by a data consumers. From a privacy perspective this approach is not satisfactory. In fact, the main concern of the data owner is to ensure that its documents are effectively protected according to the associated access constraints. The data owner should not know which data consumers want or are able to have access to its documents as long as it has the assurance that only authorized data consumers can have access to the content of the documents in cleartext.

In this section, we present a solution that allows to overcome the two shortcomings described above. The key features of our framework are described below:

- **protection specification**: as for the approach presented by Miklau et al. in [119], we consider the tree representation of an XML document and define a protection to be a tree where each node is associated to a specific authorization policy. Our authorization policies are similar to their guard formulae in that the policies associated to the tree nodes are

formalized as monotone Boolean expressions. A policy consists of conjunctions and disjunctions of conditions, where each condition is fulfilled either by a credential issued by a credential issuer trusted by the data owner or by a data-value key. Our policy model takes therefore into account (as required) both assertion-based and content-based authorization.

- **protection enforcement**: we first define an algorithm that allows to reduce the size of the policies associated to the tree nodes. This allows to reduce the number of keys that will be used to encrypt the document and therefore decrease the oversize added by the encryption process. As for the framework of Miklau et al., the encryption of a document is performed according to the standard XML Encryption recommendation [1]. Instead of being managed by the data owner, the keys used to encrypt the document are in their turn encrypted using a policy-based encryption algorithm. The resulting encrypted keys are sent together with the released encrypted document. A data consumer that wants to have access to some portions of the document does not need to contact the data owner. It has to collect qualified sets of keys for the policies associated to the portions of the document to which it wants to have access. To do so, the data consumer either has to request credentials from the specified credential issuers or has to know some data-values contained in the encrypted document.

The rest of this section is organized as follows: we first provide a brief description of the XML data model in Section 1.5.1. Then, we present our policy model in Section 1.5.2, and discuss our tree protection model in Section 1.5.3. In Sections 1.5.4 and 1.5.5, we describe our protection enforcement mechanism. We provide a formal description as well as an XML-based description of our approach. Our formal description allows to avoid the lengthy XML syntax and is thus more suitable for a scientific discussion.

1.5.1 The XML Data Model

An XML document consists of elements and attributes. An element contains a portion of the document delimited by two tags: the start tag, at the beginning of the element, with the form <tag-name>, and the end tag, at the end of the element, with the form </tag-name>, where <tag-name> indicates the type of the element. Elements can be nested at any depth and can contain other elements, called sub-elements. The start tag of each element can specify a list of attributes. An attribute is of the form name=value, where name is a label and value is a string delimited by quotes. There are different types of attributes, the most common and simple of which are the string type attributes, which allow to provide additional (textual) information about the element. For an illustration, we describe a sample XML document in Example 1.2.

Example 1.2 *A hospital at a medical school stores the collected information about its patients and staff members in XML-structured documents. The sample XML document shown in Figure 1.4 is modeled through the* <hospital-record> *root element which consists of multiple* <patient-record> *sub-elements. Each* <patient-record> *element represents a portion of the document providing both general information (name, age, address, health insurance, billing,* etc.*)*

and medical information (diagnosis, therapy, etc.*) about a patient.* <admin-info>*,* <address> *and* <diagnosis> *are examples of elements at different depth in the document.* <patient-record> *is an example of element with sub-elements in that it contains the elements* <personal-info>*,* <admin-info> *and* <medical-info>*.* <doctor-id> *is an example of element containing text (data value). Empty elements, such as* <out>*, are characterized only by a start tag and do contain neither data value nor sub-elements.* <room> *and* <record> *are examples of elements with lists of attributes. The corresponding element expressions are, respectively,* {room,number=110,bed=3} *and* {record,ref=M1YFSNA}*.*

```
<hospital-record>
  <patient-record patient-id="URJ83OMN">...  </patient-record>

  <patient-record patient-id="TY3567HEN">
    <personal-info>
      <name>Alice James</name>
      <age>50</age>
      <sex>Female</sex>
      <address>01, XML Street</address>
      <ssn>N12345</ssn>
    </personal-info>
    <admin-info>
      <room number="111" bed="3">
        <in>2006-01-06</in>
        <out></out>
      </room>
      <billing-info>...</billing-info>
    </admin-info>
    <medical-info>
      <record ref="M1YFSNA">
        <doctor-id>S1RSH</doctor-id>
        <nurse-id>S3SHJ</nurse-id>
        <diagnosis date="2006-01-05">
          diagnosis' description
        </diagnosis>
        <therapy>
          therapy's description
        </therapy>
      </record>
    </medical-info>
  </patient-record>

  <patient-record patient-id="XVGD89H5">  ...  </patient-record>

  <patient-record patient-id="LPO90TV2">...  </patient-record>
</hospital-record>
```

Figure 1.4: A sample XML document

Formally, an XML document is modeled as a node-labeled tree t, with internal nodes labeled from an alphabet of element expressions, and leaves (external nodes) labeled from a set of data values. An element expression contains typically the name of an element and a list of attribute (name,value) pairs. Each node in the tree is associated with a unique identifier obtained by applying some numbering scheme. The tree representation of the XML document shown in Figure 1.4 is given in Figure 1.5.

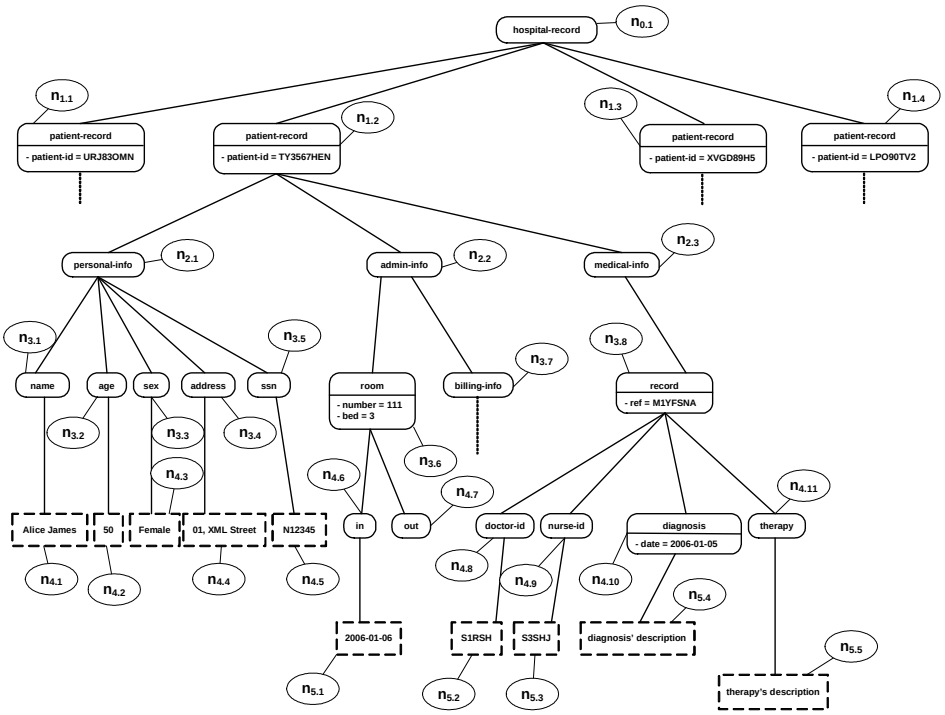


Figure 1.5: The tree representation of the XML document of Figure 1.4

In the sequel of this section, we use the following notations:

- the set of nodes (represented by their identifiers) is denoted by $\mathcal{N}^t$, and the value of a leaf node $n_x \in \mathcal{N}^t$ is denoted by $value(n_x)$.
- given two nodes $n_x, n_y \in \mathcal{N}^t$, we write $n_x \prec n_y$ when n_x is an ancestor of n_y, and $n_x \preceq n_y$ for the ancestor-or-self relation i.e. if $n_x \prec n_y$ or $n_x = n_y$.
- the depth of a node $n_x \in \mathcal{N}^t$ is the length of the path from the root to the node. The set of all nodes at a given depth is called a *level* of the tree. The subset of nodes located at level l is denoted by $\mathcal{N}^t(l)$.
- the height of a node $n_x \in \mathcal{N}^t$ is the length of the path from the node n_x to its furthest descendant (a leaf node). The height of the tree t, denoted by h^t, is the height of its root node. Therefore, if the root of the tree is located at level $l = 0$, while the deepest node is located at level $l = h^t$.

Example 1.3 *Given Figure 1.5, we illustrate in the following the terminology and notations defined above:*

- *the root of the tree is* $n_{0.1} =$ *{hospital-record}. It is located at level* $l = 0$.

- *given* $n_{1.2}$ = *{patient-record,patient-id=TY3567HEN},* $n_{2.3}$ = *{medical-info} and* $n_{3.8}$ = *{record,ref=M1YFSNA}, we have* $n_{1.2} \prec n_{2.3}$ *and* $n_{2.3} \prec n_{3.8}$.
- *the height of the tree is* $h^t = 5$ *and the node* $n_{4.10}$ = *{diagnosis,date=2006-01-05} is at level* $l = 4$ *i.e.* $n_{4.10} \in \mathcal{N}^t(4)$.
- *given the node* $n_{3.1}$ = *{name}, we have* $value(n_{3.1})$ = *"Alice James".*

1.5.2 Policy Model

Given the tree representation, denoted by t, of an XML document to be released, the data owner starts by stating for each node $n_x \in \mathcal{N}^t$ a policy, denoted by Pol_{n_x}, specifying the conditions under which a data consumer is allowed to have access to the node n_x and its sub-nodes. In other words, Pol_{n_x} defines the constraints that have to be fulfilled by a data consumer in order for him to be authorized to have access to the element represented by the node n_x, its attributes and its sub-elements. The policy Pol_x can have one of the following forms:

- $Pol_{n_x} = true$: in this case, any data consumer is allowed to have access to n_x and its sub-nodes. In this case, Pol_{n_x} is called an 'allow-all' policy.
- $Pol_{n_x} = false$: in this case, no data consumer is allowed to have access to n_x and its sub-nodes. In this case, Pol_{n_x} is called a 'deny-all' policy.
- Pol_{n_x} consists of conjunctions (AND / $\wedge$) and disjunctions (OR / $\vee$) of conditions, where each condition is fulfilled by a specific key: in this case, a data consumer is compliant with a policy if and only if it has access to a qualified set of keys for the policy i.e. a set of keys fulfilling the logical combination of conditions specified by the policy. We distinguish two types of conditions:
  - **credential-based conditions**: each condition is defined through a pair, denoted by $\langle I_\kappa, A\rangle$, specifying an assertion $A \in \{0,1\}^*$ and a credential issuer $I_\kappa \in \mathcal{I}$ that is trusted to check and certify the validity of A. A data consumer fulfills the condition $\langle I_\kappa, A\rangle$ if and only if it has been issued the credential $\varsigma(R_\kappa, A)$. Here, we assume the existence of a set of credential issuers $\mathcal{I} = \{I_1, \ldots, I_N\}$, where each issuer $I_\kappa \in \mathcal{I}$ is trusted by the data consumer to issue specific credentials.
  - **data-value conditions**: each condition is defined through a pair, denoted by $\langle \aleph, A\rangle$, specifying a data value $A \in \{0,1\}^*$ and a symbol $\aleph$. The data value A can be any text, number, date, passphrase, *etc.* that can normally occur in an XML document. Here, the symbol $\aleph$ is used to differentiate the data-value conditions from the credential-based conditions, which specify a credential issuer $I_\kappa \in \mathcal{I}$ that is trusted to check and certify the validity of the content of A. A data consumer fulfills the condition $\langle \aleph, A\rangle$ if and only if it knows the data-value A. Equivalently, a data consumer fulfills the condition $\langle \aleph, A\rangle$ if and only if it knows the value $\varsigma(\aleph, A) = f(A)$, where f is a publicly known one-way function such as a hash function.

In the following example, we illustrate the two considered types of conditions.

Example 1.4 *Assume that the hospital considered in Example 1.2 plans to publish its records so that patients can read their personal information, staff members can refer to patients' records, and other departments at the same school (or partners) can conduct related research. Because some information is sensitive, it should be protected so that it is accessible only to authorized users. Assume that there exists two credential issuers: I_1 represents the hospital that is trusted for issuing credentials to the hospital staff members (doctors, nurses,* etc*), and I_2 represents the research department that is trusted for issuing credentials to research staff members (researchers, technicians,* etc*). Before publishing a record, the hospital defines the policies protecting its different portions. In the following, we give examples of such policies, while referring to the sample XML document described in Example 1.2 and its tree representation shown in Figure 1.5.*

1. *access to a patient's medical information is conditioned by the possession of either a doctor credentials issued by the hospital or a researcher credential issued by the research department.*

   i.e. $Pol_{n_{2.3}} = \langle I_1, doctor\rangle \vee \langle I_2, researcher\rangle$

2. *access to the name of a patient is conditioned by the possession of a doctor credential and the doctor-in-charge's credential.*

   i.e. $Pol_{n_{3.1}} = \langle I_1, doctor\rangle \wedge \langle I_1, S1RSH\rangle$

3. *access to the social security number (SSN) of a patient is conditioned by the knowledge of its name and the possession of a hospital staff member.*

   i.e. $Pol_{n_{3.5}} = \langle \aleph, Alice\ James\rangle \wedge \langle I_1, staff\rangle$

   Note that the data-value "Alice James" should not appear in the public representation of the policy $Pol_{n_{3.5}}$. Further details on policy representation are given later in this section.

As for the policy model adopted for policy-based encryption schemes (Section 1.3.1), our policies can be written in the CDNF form i.e.

$$Pol_{n_x} = \wedge_{i=1}^{m}[\vee_{j=1}^{m_i}[\wedge_{k=1}^{m_{i,j}}\langle L_{\kappa_{i,j,k}}, A_{i,j,k}\rangle]], \text{ where } L_{\kappa_{i,j,k}} \in \mathcal{I} \cup \{\aleph\} \text{ and } A_{i,j,k} \in \{0,1\}^*$$

For the sake of readability, we consider the following notations:

- $\varsigma(Pol_{n_x})$ denotes the set of all the keys corresponding to the different conditions specified by policy Pol_{n_x} i.e. $\varsigma(Pol_{n_x}) = \{\{\varsigma(R_{\kappa_{i,j,k}}, A_{i,j,k})\}_{k=1}^{m_{i,j}}\}_{j=1}^{m_i}\}_{i=1}^{m}$. Note that for $Pol_{n_x} = true$, we have $\varsigma(Pol_{n_x}) = \emptyset$, while for $Pol_{n_x} = false$, $\varsigma(Pol_{n_x})$ does not exist
- $\check{\varsigma}(Pol_{n_x})$ denotes the power set of $\varsigma(Pol_{n_x})$ i.e. the set of all the subsets of $\varsigma(Pol_{n_x})$

- For some $\{j_i \in \{1,\ldots,m_i\}\}_{i=1}^{m}$, $\varsigma_{j_1,\ldots,j_m}(Pol) = \{\{\varsigma(R_{\kappa_{i,j_i,k}}, A_{i,j_i,k})\}_{k=1}^{m_{i,j_i}}\}_{i=1}^{m}$, where $R_{\kappa_{i,j_i,k}}$ is either the public key of the credential issuer $I_{\kappa_{i,j_i,k}} \in \mathcal{I}$ or the symbol $\aleph$

- For a subset of credentials $\rho \subset \check{\varsigma}(Pol_{n_x})$, '$\rho \vDash Pol_{n_x}$' denotes the fact that ρ is a qualified set of credentials for policy Pol_{n_x}

According to the notation defined above, the following equivalence holds:

$$\forall\ \rho \in \check{\varsigma}(Pol_{n_x}):\ \ \rho \vDash Pol \Leftrightarrow \exists\{j_i \in \{1,\ldots,m_i\}\}_{i=1}^{m}\ \text{s.th.}\ \rho = \varsigma_{j_1,\ldots,j_m}(Pol_{n_x}) \tag{1.7}$$

Remark 1.13 *Our policy model extends the one defined in Section 1.3.1 for the policy-based encryption primitive. The added feature is the data-value conditions which can be 'informally' viewed as credential-based conditions for which the secretness of the credential is not based on the secretness of the issuer's master key, but on the secretness of the assertion itself. The definition of the policy-based encryption primitive can be easily extended to support data-value conditions.*

An illustration of our policy model is given in the following example.

Example 1.5 *Assume that there exists a set of credential issuers $\mathcal{I} = \{I_1, I_2, I_3\}$, where issuer I_κ is trusted by the data owner to issue credentials A_κ and A'_κ, for $\kappa = 1,2,3$ respectively.*

- *assume that the data owner requires that a data consumer has been issued either the two credentials $\varsigma(R_1, A_1)$ and $\varsigma(R_2, A_2)$ or the two credentials $\varsigma(R_3, A_3)$ and $\varsigma(R_3, A'_3)$, to be authorized to access a node $n_x \in \mathcal{N}^t$ and its sub-nodes. In this case, we have*

$$Pol_{n_x} = (\langle R_1, A_1\rangle \wedge \langle R_2, A_2\rangle) \vee (\langle R_3, A_3\rangle \wedge \langle R_3, A'_3\rangle)$$

Given the CDNF*-based notation defined above, policy Pol_{n_x} is such that*

$$\begin{cases} m = 1, m_1 = 2 \\ m_{1,1} = 2 : \langle L_{1,1,1}, A_{1,1,1}\rangle = \langle R_1, A_1\rangle, \langle L_{1,1,2}, A_{1,1,2}\rangle = \langle R_2, A_2\rangle \\ m_{1,2} = 2 : \langle L_{1,2,1}, A_{1,2,1}\rangle = \langle R_3, A_3\rangle, \langle L_{1,2,2}, A_{1,2,2}\rangle = \langle R_3, A'_3\rangle \end{cases}$$

- *assume that the data owner requires that a data consumer has been issued either the credential $\varsigma(R_1, A'_1)$ or the two credential $\varsigma(R_2, A'_2)$, and additionally knows a registration password A_0, to be authorized to access a node $n_y \in \mathcal{N}^t$. In this case, we have*

$$Pol_{n_y} = (\langle R_1, A'_1\rangle \vee \langle R_2, A'_2\rangle) \wedge \langle \aleph, A_0\rangle$$

Given the CDNF*-based notation defined above, policy* Pol_{n_y} *is such that*

$$\begin{cases} m = 2, m_1 = 2, m_2 = 1 \\ m_{1,1} = 1 : \langle L_{1,1,1}, A_{1,1,1} \rangle = \langle R_1, A'_1 \rangle \\ m_{1,2} = 1 : \langle L_{1,1,2}, A_{1,1,2} \rangle = \langle R_2, A'_2 \rangle \\ m_{2,1} = 1 : \langle L_{1,2,1}, A_{1,2,1} \rangle = \langle \aleph, A_0 \rangle \end{cases}$$

Now that we have described our formal policy model, we present a basic XML-based representation of our policies in the remaining of this section.

In addition to document representation and exchange, XML appears as a natural choice for policy representation, thanks to its human and machine readability and the ease with which its syntax and semantics can be extended. Several XML-based policy languages can be found in the literature such as the XML Access Control Language (XACL) [103] and the eXtensible Access Control Markup Language (XACML) [73]. We refer to [149] for a comprehensive survey on XML-based policy languages.

Four of the XML-based policy languages that have been proposed as the basis for a new standard are based on Boolean combination of predicates [6]. These languages are: the Web Services Policy Framework (WS-Policy) [144], the Web Services Description Language (WSDL) [143] with the addition of compositors [53], the XACML profile for Web Services (WSPL) [74], and a language outline from IONA Technologies [124]. While these four languages have been proposed in the context of Web Services, they can be naturally used in the general context of XML-structured documents. In particular, WS-Policy and WSPL can be used as containers for our policy model. These languages are complex and support more features than the ones required in the context of our framework. For the sake of readability, we define a simplified XML-based policy syntax, which we briefly describe below.

In our basic XML-based policy language, a policy is modeled through a <Policy> root element. A shorthand XML schema[1] of the <Policy> element is shown below in Figure 1.6. The different components of the <Policy> root element are defined below:

- <Condition>: this element is the container of our conditions. Its content depends on the type of condition as follows:
  - in the case of credential-based conditions, the <Condition> element contains two sub-elements: an <Assertion> element and an <Issuer> element. The <Assertion> element contains the assertion to be fulfilled in order to fulfill the <Condition> element. The <Assertion> element should support any type of assertion representation ranging from simple text-based assertions to advanced SAML assertions [75].

[1] In the XML shorthand scheme, the '+' sign declares that the sub-element must occur one or more times inside the element, while the '?' sign declares that the sub-element can occur zero or one time inside the element.

The <issuer> element provides the name or the public key of the credential issuer that is trusted to check and certify the validity of the statements contained in the <Assertion> element.

– in the case of data-value conditions, the <Condition> element contains one sub-element denoted by <Data-value>. A natural way to reference the data-values within an XML document is to use the XPath language [142], which is a widely used standard technology that allows to address different portions of an XML document. XPath considers the tree representation of an XML document and forms a path expression that selects a set of tree-nodes. The path expression, called location path, is composed of a sequence of location steps where each location step has three parts: axis, node test, and optional predicate list.

- <All>: this element corresponds to the right-hand AND Boolean operator in the CDNF form. It contains one or multiple <Condition> sub-elements. All the <Condition> elements must be fulfilled in order for the <All> element to be fulfilled.

- <AtLeastOne>: this element corresponds to the OR Boolean operator in the CDNF form. It contains one or multiple <All> sub-elements. At least one of the <All> elements must be fulfilled in order for the <AtLeastOne> element to be fulfilled.

- <Policy>: this root-element contains one or multiple <AtLeastOne> sub-elements. The different <AtLeastOne> elements must be fulfilled in order for the hole policy to be fulfilled. This corresponds to the left-hand AND Boolean operator in the CDNF form. Besides, this element may contain an optional identifier specified by the ID attribute.

```
<Policy>
   (<AtLeastOne>
      (<All>
          (<Condition>
               <Data-value>?
               <Assertion>?
               <Issuer>?
           </Condition>)+
       </All>)+
    </AtLeastOne>)+
</Policy>
```

Figure 1.6: A shorthand schema for the <Policy> element

An illustration of our XML-based policy representation is given below.

Example 1.6 *Consider the policy $Pol_{n_{3.5}}$ specified in Example 1.4. The XML-based representation for this policy is given in Figure 1.7. The policy states that access to the social security number (SSN) of a patient is conditioned by the knowledge of its name and the possession of a hospital staff member. The XPath expression specified by the* <Data-value> *element refers to the*

content of the <name> *element in the sub-tree of the* <hospital-record> *document corresponding to the patient whose identifier is* "TY3567HEN". *The* Assertion *element contains a text-based assertion that can be signed by the credential issuer of the hospital to generate a credential that is given the staff members. A more sophisticated representation of assertions such as SAML assertions [75] should be considered in real-world scenarios. The is also the case of the* <Issuer> *where the reference to the credential issuer and its public key should be more elaborated than the simplistic text-based URL reference used in our illustrative example.*

```
<Policy>
   <AtLeastOne>
     <All>
       <Condition>
          <Data-value>
 /hospital-record/patient-record[patient-id="TY3567HEN"]/personal-info/name/
          </Data-value>
       </Condition>
       <Condition>
          <Assertion>staff-2006</Assertion>
          <Issuer>www.hospital.com</Issuer>
       </Condition>
     </All>
   </AtLeastOne>
</Policy>
```

Figure 1.7: A sample <Policy> element

1.5.3 Protection Model

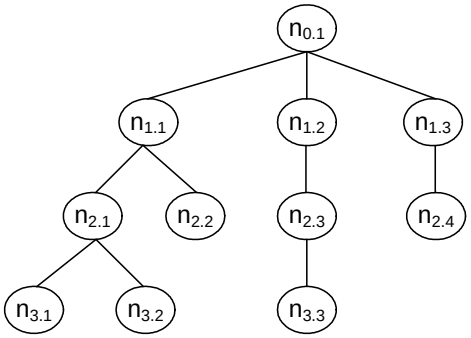
A tree protection for an XML document groups the different policies associated by the data owner to the different portions of the document. Formally, it is defined as follows:

Definition 1.4 *A* **tree protection** Π^t *over a node-labeled tree* t *is a function that associates to each node* $n_x \in \mathcal{N}^t$ *a policy* $Pol_{n_x} = \Pi^t(n_x)$ *defining the conditions under which a data consumer is allowed to have access to the set of nodes* $\{n_y \in \mathcal{N}^t \mid n_x \preceq n_y\}$. *Thus, the tree protection enforcement mechanism is such that, in order to reach a node* $n_y \in \mathcal{N}^t$ *from the root, the data consumer has to satisfy all the policies* $Pol_{n_x} = \Pi^t(n_x)$, *for all* $n_x \preceq n_y$. *In other words, the data consumer must have access to a set of credentials fulfilling all the policies* Pol_{n_x}, *for all* $n_x \preceq n_y$.

Example 1.7 *An example of a tree protection over a node-labeled tree is given in Figure 1.8.*

Consider a node-labeled tree t, an associated tree protection Π^t and a node $n_x \in \mathcal{N}^t$. Assume that there exists a condition $\langle L, A\rangle$, for $L \in \mathcal{I} = \{I_1, \ldots, I_N\} \cup \{\aleph\}$ and $A \in \{0,1\}^*$, such that the fulfillment of policy $Pol_{n_x} = \Pi^t(n_x)$ requires the fulfillment of condition $\langle L, A\rangle$. Because of the downward propagation of the policies specified by Π^t, it is not necessary to include the

condition $\langle L,A\rangle$ in the policies $Pol_{n_y} = \Pi^t(n_y)$, for $n_x \prec n_y \in \mathcal{N}^t$. In other words, replacing the condition $\langle L,A\rangle$ by the *true* logical statement in the policies $Pol_{n_y} = \Pi^t(n_y)$, for $n_x \prec n_y \in \mathcal{N}^t$, does not affect the restrictiveness of the tree protection Π^t. If the condition $\langle L,A\rangle$ is specified by policy Pol_{n_y}, we say that the tree protection contains a protection redundancy at node n_y.



$n_x \in \mathcal{N}^t$	$\Pi^t(n_x)$
$n_{0.1}$	$true$
$n_{1.1}$	$(\langle R_1,A_1\rangle \wedge \langle R_2,A_2\rangle) \vee (\langle R_3,A_3\rangle \wedge \langle R_3,A'_3\rangle)$
$n_{1.2}$	$(\langle R_1,A'_1\rangle \vee \langle R_2,A'_2\rangle) \wedge \langle \aleph,A_0\rangle$
$n_{1.3}$	$\langle R_3,A_3\rangle$
$n_{2.1}$	$\langle R_1,A_1\rangle \wedge \langle R_2,A_2\rangle$
$n_{2.2}$	$true$
$n_{2.3}$	$(\langle R_1,A'_1\rangle \vee \langle \aleph,A_0\rangle) \wedge \langle R_3,A_3\rangle$
$n_{2.4}$	$\langle R_1,A_1\rangle \vee (\langle R_2,A_2\rangle \wedge \langle R_3,A_3\rangle)$
$n_{3.1}$	$false$
$n_{3.2}$	$\langle R_1,A_1\rangle \wedge \langle R_3,A_3\rangle$
$n_{3.3}$	$\langle \aleph,A_0\rangle$

Figure 1.8: A tree protection over a node-labeled tree

Our protection enforcement mechanism (described later in Section 1.5.4) is such that the generated overhead (with respect to the original cleartext data) is proportional to the size of the policies as well as to the number of distinct policies specified by the tree protection (see Remark 1.16 for further details). It is therefore necessary to reduce the size of the policies specified by the tree protection Π^t by eliminating the possible protection redundancies. Examples of protection redundancies within a tree protection is given below.

Example 1.8 *Consider the tree protection shown in Figure 1.8. We describe in the following three cases of protection redundancy:*

- *the fulfillment of policy $Pol_{n_{2.1}} = \Pi^t(n_{2.1})$ requires the fulfillment of condition $\langle R_1,A_1\rangle$. Therefore, condition $\langle R_1,A_1\rangle$ specified by $Pol_{n_{3.2}} = \Pi^t(n_{3.2})$ becomes obsolete. In other words, setting $Pol_{n_{3.2}} = true \wedge \langle R_3,A_3\rangle = \langle R_3,A_3\rangle$ does not affect the restrictiveness of the tree protection Π^t, while reducing the size of policy $Pol_{n_{3.2}}$.*
- *the fulfillment of policy $Pol_{n_{1.2}} = \Pi^t(n_{1.2})$ requires the fulfillment of condition $\langle \aleph,A_0\rangle$. Therefore, condition $\langle \aleph,A_0\rangle$ specified by $Pol_{n_{3.3}} = \Pi^t(n_{3.3})$ becomes obsolete. In other words, setting $Pol_{n_{3.3}} = true$ does not affect the restrictiveness of the tree protection Π^t, while reducing the number of distinct policies specified by Π^t.*
- *the fulfillment of policy $Pol_{n_{1.3}} = \Pi^t(n_{1.3})$ requires the fulfillment of condition $\langle R_3,A_3\rangle$. Therefore, condition $\langle R_3,A_3\rangle$ specified by $Pol_{n_{2.4}} = \Pi^t(n_{2.4})$ becomes obsolete. In other words, setting $Pol_{n_{2.4}} = \langle R_1,A_1\rangle \vee (\langle R_2,A_2\rangle \wedge true) = \langle R_1,A_1\rangle \vee \langle R_2,A_2\rangle$ does not affect the restrictiveness of the tree protection Π^t, while reducing the size of policy $Pol_{n_{2.4}}$.*

To formally address the reduction of tree protections, we first introduce the notion of protection refinement.

The notion of policy refinement, as defined in [13], is fundamental for many situations in policy management. Intuitively, one policy refines another if using the first policy automatically fulfills the second policy. In other words, the second policy is at least as restrictive as the first policy. In the following, we adapt the notion of refinement to our tree protection model.

To formalize our notion of protection refinement, we consider the following notations:

- $\varsigma(\Pi^t) = \bigcup_{n_x \in \mathcal{N}^t} \varsigma(\Pi^t(n_x))$ i.e. $\varsigma(\Pi^t)$ denotes the set of all the keys corresponding to the different conditions specified by the policies associated, through the tree protection $\varsigma(\Pi^t)$, to the nodes of t
- $\check{\varsigma}(\Pi^t)$ denotes the power set of $\varsigma(\Pi^t)$ i.e. the set of all the subsets of $\varsigma(\Pi^t)$
- for a node $n_y \in \mathcal{N}^t$ and a subset of credentials $\rho \subset \varsigma(\Pi^t)$, '$\rho \overset{\Pi^t}{\rightsquigarrow} n_x$', denotes the fact that, for all $n_x \preceq n_y \in \mathcal{N}^t$, there exists $\rho_{n_x} \subset \rho$ such that $\rho_{n_x} \vDash Pol_{n_x} = \Pi^t(n_x)$ i.e. with respect to the tree protection Π^t, a data consumer having access to the set of credentials ρ is able to reach the node n_x.

Definition 1.5 *Given a node-labeled tree t, let Π_1^t and Π_2^t be two tree protections over t. Then, Π_2^t is a* **refinement** *of Π_1^t, written $\Pi_2^t \geqslant \Pi_1^t$, if and only if, for all nodes $n_x \in \mathcal{N}^t$, if a data consumer is not able to reach a node n_x according to the tree protection Π_2^t, then it cannot reach n_x according the tree protection Π_1^t. This can be expressed as follows:*

$$\forall\ \rho \in \check{\varsigma}(\Pi_1^t) \cup \check{\varsigma}(\Pi_2^t)\ \ ,\ \ \forall\ n_x \in \mathcal{N}^t :\ \ \rho \overset{\Pi_2^t}{\rightsquigarrow} n_x\ \ \Rightarrow\ \ \rho \overset{\Pi_1^t}{\rightsquigarrow} n_x \tag{1.8}$$

In the following, we define the notion of equivalence between two tree protections.

Definition 1.6 *Given a node-labeled tree t, let Π_1^t and Π_2^t be two tree protections over t. Then, Π_1^t and Π_2^t are* **equivalent**, *written $\Pi_2^t \leqslant\!\!\geqslant \Pi_2^t$, if and only if $\Pi_2^t \geqslant \Pi_1^t$ and $\Pi_1^t \geqslant \Pi_2^t$.*

Now that the notions of protection refinement and equivalence have been defined, we define a reduced tree protection as follows:

Definition 1.7 *Given a tree protection Π^t over a node-labeled tree t, it is called a* **reduced tree protection** *over the tree t if and only if, for all $n_x \prec n_y \in \mathcal{N}^t$, there exists no condition defined by policy $Pol_{n_y} = \Pi^t(n_y)$ whose fulfillment if required for the fulfillment of policy $Pol_{n_x} = \Pi^t(n_x)$.*

In the following, we present an algorithm that allows converting any tree protection Π^t over a node-labeled tree t into an equivalent reduced tree protection $\overline{\Pi}^t$. To formalize our reduction algorithm, we consider the following notations:

- for $n_x \in \mathcal{N}^t$, $\varsigma^{\cap}(Pol_{n_x}) = \bigcap_{\{j_i \in \{1,\ldots,m_i\}\}_{i=1}^{m}} \varsigma_{j_1,\ldots,j_m}(Pol_{n_x})$ i.e. $\rho_{n_x}^{\cap}$ represents the intersection of all the qualified sets of credentials for policy Pol_{n_x}

- for $n_x, n_y \in \mathcal{N}^t$, $\varsigma^{\cap}(Pol_{n_x}, Pol_{n_y}) = \varsigma^{\cap}(Pol_{n_x}) \cap \varsigma(Pol_{n_y})$ i.e. $\rho_{n_x \rightsquigarrow n_y}$ denotes the set of credentials that are required for the fulfillment of policy Pol_{n_x} and which fulfill a condition specified by Pol_{n_y}.

Input : A node-labeled tree t , A tree protection Π^t ;
Output : A (reduced) tree protection $\overline{\Pi}^t$;

$l \leftarrow 1$;
while $l \leq h^t$ **do**
  foreach $n_y \in \mathcal{N}^t(l)$ **do**
    foreach $n_x \prec n_y$ **do**
      if $\varsigma^{\cap}(\overline{\Pi}^t(n_x), \Pi^t(n_y)) \neq \emptyset$ **then**
        let Pol_{n_y} be the policy obtained by rewriting $\Pi^t(n_y)$, while replacing each condition fulfilled by a credential included in $\varsigma^{\cap}(\overline{\Pi}^t(n_x), \Pi^t(n_y))$ by the *true* logical statement **in**
          $\overline{\Pi}^t(n_y) \leftarrow \overline{Pol}_{n_y}$;
      else
        $\overline{\Pi}^t(n_y) \leftarrow \Pi^t(n_y)$;
      end
  $l \leftarrow l+1$;
end
return $\overline{\Pi}^t$;

Algorithm 1.1: Tree protection reduction algorithm

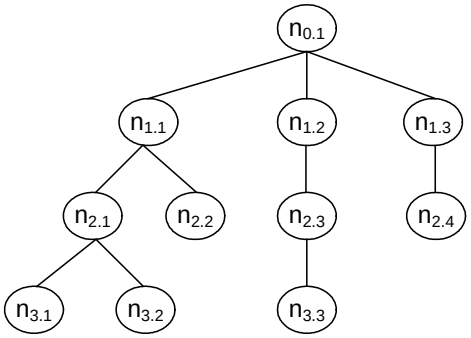
In the following, we give a step-by-step description of the execution of Algorithm 1.1.

Example 1.9 *On input of the node-labeled tree and the associated tree protection given in Figure 1.8, Algorithm 1.1 does the following:*

1. *As* $\varsigma^{\cap}(\overline{\Pi}^t(n_{0.1}), \Pi^t(n_{1.j})) = \emptyset$ *(for* $j = 1,2,3$*), the policies associated to nodes* $n_{1.1}$, $n_{1.2}$ *and* $n_{1.3}$ *remain unchanged*
2. *As* $\varsigma^{\cap}(\overline{\Pi}^t(n_{0.1}), \Pi^t(n_{2.j})) = \emptyset$ *and* $\varsigma^{\cap}(\overline{\Pi}^t(n_{1.1}), \Pi^t(n_{2.j})) = \emptyset$ *(for* $j = 1,2$*), the policies associated to nodes* $n_{2.1}$ *and* $n_{2.2}$ *remain unchanged*
3. *Whereas* $\varsigma^{\cap}(\overline{\Pi}^t(n_{0.1}), \Pi^t(n_{2.3})) = \emptyset$, $\varsigma^{\cap}(\overline{\Pi}^t(n_{1.2}), \Pi^t(n_{2.3})) = \{\varsigma(\aleph, A_0)\}$. *Therefore, the policy associated to node* $n_{2.3}$ *is rewritten, while replacing the condition* $\langle \aleph, A_0 \rangle$ *by the true logical statement i.e.* $\overline{\Pi}^t(n_{n_{2.3}}) = (\langle R_1, A_1' \rangle \vee true) \wedge \langle R_3, A_3 \rangle = \langle R_3, A_3 \rangle$.
4. *Whereas* $\varsigma^{\cap}(\overline{\Pi}^t(n_{0.1}), \Pi^t(n_{2.4})) = \emptyset$, $\varsigma^{\cap}(\overline{\Pi}^t(n_{1.3}), \Pi^t(n_{2.4})) = \{\varsigma(R_3, A_3)\}$. *Therefore, the policy associated to node* $n_{2.4}$ *is rewritten, while replacing the condition* $\langle R_3, A_3 \rangle$ *by the true logical statement i.e.* $\overline{\Pi}^t(n_{n_{2.4}}) = \langle R_1, A_1 \rangle \vee (\langle R_2, A_2 \rangle \wedge true) = \langle R_1, A_1 \rangle \vee \langle R_2, A_2 \rangle$.
5. *As* $\varsigma^{\cap}(\overline{\Pi}^t(n_{i.1}), \Pi^t(n_{3.1})) = \emptyset$ *(for* $i = 0,1,2$*), the policy associated to node* $n_{3.1}$ *remains unchanged*
6. *Whereas* $\varsigma^{\cap}(\overline{\Pi}^t(n_{i.1}), \Pi^t(n_{3.2})) = \emptyset$ *(for* $i = 0,1$*),* $\varsigma^{\cap}(\overline{\Pi}^t(n_{2.1}), \Pi^t(n_{3.2})) = \{\langle R_1, A_1 \rangle\}$. *Therefore, the policy associated to node* $n_{3.2}$ *is rewritten, while replacing the condition* $\langle R_1, A_1 \rangle$ *by the true logical statement i.e.* $\overline{\Pi}^t(n_{n_{3.2}}) = true \wedge \langle R_3, A_3 \rangle = \langle R_3, A_3 \rangle$.

7. *Whereas* $\varsigma^{\cap}(\overline{\Pi}^t(n_{0.1}),\Pi^t(n_{3.3})) = \emptyset$ *and* $\varsigma^{\cap}(\overline{\Pi}^t(n_{2.3}),\Pi^t(n_{3.3})) = \{\varsigma(R_3,A_3)\}$, *we have* $\varsigma^{\cap}(\overline{\Pi}^t(n_{1.2}),\Pi^t(n_{3.3})) = \{\varsigma(\aleph,A_0)\}$. *Therefore, the policy associated to node* $n_{3.3}$ *is rewritten, while replacing the condition* $\langle\aleph,A_0\rangle$ *by the* true *logical statement.*

The reduced tree protection output by Algorithm 1.1 is summarized in Figure 1.9.



$n_x \in \mathcal{N}^t$	$\overline{\Pi}^t(n_x)$
$n_{0.1}$	*true*
$n_{1.1}$	$(\langle R_1,A_1\rangle \wedge \langle R_2,A_2\rangle) \vee (\langle R_3,A_3\rangle \wedge \langle R_3,A'_3\rangle)$
$n_{1.2}$	$(\langle R_1,A'_1\rangle \vee \langle R_2,A'_2\rangle) \wedge \langle\aleph,A_0\rangle$
$n_{1.3}$	$\langle R_3,A_3\rangle$
$n_{2.1}$	$\langle R_1,A_1\rangle \wedge \langle R_2,A_2\rangle$
$n_{2.2}$	*true*
$n_{2.3}$	$\langle R_3,A_3\rangle$
$n_{2.4}$	$\langle R_1,A_1\rangle \vee \langle R_2,A_2\rangle$
$n_{3.1}$	*false*
$n_{3.2}$	$\langle R_3,A_3\rangle$
$n_{3.3}$	*true*

Figure 1.9: A reduced tree protection over a node-labeled tree

Note that in addition to reducing the size of the policies, our reduction algorithm allows to potentially reduce the number of distinct policies in the tree protection associated to the document to be protected (8 distinct policies for Π^t *vs. 5 distinct policies for* $\overline{\Pi}^t$*). This will have a positive impact on the overhead caused by our cryptography-based enforcement mechanism.*

The correctness and completeness of Algorithm 1.1 are stated below.

Theorem 1.2 *Let* $\overline{\Pi}^t$ *be the tree protection obtained by running Algorithm 1.1 on input of a tree protection* Π^t *over a node-labeled tree* t. *Then, the following statements hold: 1) Algorithm 1.1 is complete, 2)* $\overline{\Pi}^t$ *is a reduced tree protection, and 3)* Π^t *and* $\overline{\Pi}^t$ *are equivalent i.e.* $\Pi^t \lesseqgtr \overline{\Pi}^t$.

Proof In the following, we prove the three statements of Theorem 1.2:

1. For each node in the node-labeled tree t, Algorithm 1.1 processes each of its ancestors. As the total number of nodes and the height of the tree (and therefore the number of ancestors for each node) are finite, Algorithm 1.1 is complete.

2. Algorithm 1.1 parses all the nodes of the tree t. For each node $n_y \in \mathcal{N}^t$, the algorithm detects all the possible protection redundancies by checking the policies associated to the ancestors of n_y by the tree protection Π^t. Then, the algorithm redefines the policies in the protection $\overline{\Pi}^t$ by eliminating the detected redundancies. Therefore, the obtained tree protection is free from protection redundancies.

3. Consider a node $n_x \in \mathcal{N}^t$ and a set of credentials $\rho \in \breve{\varsigma}(\Pi^t) \cup \breve{\varsigma}(\overline{\Pi}^t)$. First, note that, by construction, we have $\breve{\varsigma}(\overline{\Pi}^t) \subset \breve{\varsigma}(\Pi^t)$, which means that $\breve{\varsigma}(\Pi^t) \cup \breve{\varsigma}(\overline{\Pi}^t) = \breve{\varsigma}(\Pi^t)$.

According to the processing of Algorithm 1.1, policy $\bar{\Pi}^t(n_x)$ corresponds to policy $\Pi^t(n_x)$ rewritten while potentially replacing some conditions by the *true* logical statement. In other words, the tree protection Π^t is at least as restrictive as the tree protection $\bar{\Pi}^t$ at node n_x. As this is valid for all the nodes $n_x \in \mathcal{N}^t$, we have $\Pi^t \geqslant \bar{\Pi}^t$. Proving that $\bar{\Pi}^t \geqslant \Pi^t$ returns to proving that the statement $\mathcal{S}_{n_x} : \rho \overset{\bar{\Pi}^t}{\rightsquigarrow} n_x \;\Rightarrow\; \rho \overset{\Pi^t}{\rightsquigarrow} n_x$ holds for all nodes $n_x \in \mathcal{N}^t$, which we do recursively as described below:

- for $l = 0$, for all nodes $n_x \in \mathcal{N}^t(0)$, the statement $\mathcal{S}_{n_x}$ is valid. In fact, there is only one node at level $l = 0$ (corresponding to the root of the tree) and for that node, Algorithm 1.1 does not change the policy specified by Π^t.
- assume that for some $l < h^t$, the statement $\mathcal{S}_{n_x}$ is valid for all nodes $n_x \in \mathcal{N}^t(l)$ (recursion assuption). We prove that the statement $\mathcal{S}_{n_x}$ is valid for all nodes $n_x \in \mathcal{N}^t(l+1)$ as follows: Consider a node $n_y \in \mathcal{N}^t(l+1)$ and assume that $\rho \overset{\bar{\Pi}^t}{\rightsquigarrow} n_y$. Then, for all $n_x \prec n_y$, we have $\rho \overset{\bar{\Pi}^t}{\rightsquigarrow} n_y$. Therefore, according to the recursion assumption, we have $\rho \overset{\Pi^t}{\rightsquigarrow} n_y$, for all $n_x \prec n_y$. This means that for all $n_x \prec n_y$, we have $\varsigma^{\cap}(\bar{\Pi}^t(n_x), \Pi^t(n_y)) \subset \rho$. In fact, the difference between policy $\bar{\Pi}^t(n_y)$ and policy $\Pi^t(n_y)$ is that the conditions in $\Pi^t(n_y)$ which are fulfilled by the credentials contained in $\varsigma^{\cap}(\bar{\Pi}^t(n_x), \Pi^t(n_y))$, for all $n_x \prec n_y$, are replaced by *true*. As $\rho \overset{\bar{\Pi}^t}{\rightsquigarrow} n_y$ and $\varsigma^{\cap}(\bar{\Pi}^t(n_x), \Pi^t(n_y)) \subset \rho$ for all $n_x \prec n_y$, we have $\rho \overset{\Pi^t}{\rightsquigarrow} n_y$.

□

Once a reduced tree protection is specified for the document to be protected, it is enforced using the protection enforcement mechanism described in the following section.

1.5.4 Protection Enforcement: Formal Description

Given a node-labeled tree, denoted by t, and an associated reduced tree protection $\bar{\Pi}^t$, our protection enforcement mechanism allows the data owner to generate a node-labeled tree $t^{\bar{\Pi}^t}$ representing the encryption of the original tree t according to the tree protection $\bar{\Pi}^t$. The intuition behind our encryption mechanism is as follows: the encrypted tree $t^{\bar{\Pi}^t}$ is first initialized with the original tree t. It is then processed according to the associated tree protection $\bar{\Pi}^t$. Because of the downward propagation of the policies specified by $\bar{\Pi}^t$, the processing of $t^{\bar{\Pi}^t}$ proceeds by an upward traversal of $t^{\bar{\Pi}^t}$ (from leaf nodes to the root node). At each level of the tree $t^{\bar{\Pi}^t}$, each node $n_x \in \mathcal{N}^{t^{\bar{\Pi}^t}}$ is processed according to the following rules:

- if $\bar{\Pi}^t(n_x) = \mathit{false}$, then the node n_x together with all its children and descendants must not appear in $t^{\bar{\Pi}^t}$ i.e. the subtree of t whose root is the node n_x must be removed from $t^{\bar{\Pi}^t}$
- if $\bar{\Pi}^t(n_x) = \mathit{true}$, then the node n_x together with all its children and descendants must appear without any modification (in plaintext) in $t^{\bar{\Pi}^t}$.

- otherwise, the node n_x together with its descendants is encrypted according to the policy $\overline{\Pi}^t(n_x)$. This is performed as follows:

  1. A randomly chosen symmetric key is associated by the data owner to policy $\overline{\Pi}^t(n_x)$
  2. The subtree of t whose root is the node n_x is encrypted with the chosen symmetric key using a symmetric encryption algorithm specified by the data owner
  3. The symmetric key is encrypted according to policy $\overline{\Pi}^t(n_x)$ using a policy-based encryption algorithm specified by the data owner

More formally, our protection enforcement mechanism consists of three stages: Initialization, Tree-Encrypt and Key-Wrap, which are defined as follows:

- **Initialization**. The data owner does the following:

  1. Choose two encryption schemes: a symmetric encryption scheme, denoted by $\mathsf{E}^{\mathrm{sym}}$, and a policy-based encryption scheme, denoted by POLBE.
  2. Define a policy-key mapping function $\Omega^{\overline{\Pi}^t}$, which associates to each distinct policy defined by $\overline{\Pi}^t$, a randomly chosen symmetric key for the $\mathsf{E}^{\mathrm{sym}}$ scheme.
  3. Define a three-column table $\mathcal{T}^{\overline{\Pi}^t}$, called *protection table*. The first column of $\mathcal{T}^{\overline{\Pi}^t}$ contains the different policies specified by the tree protection $\overline{\Pi}^t$. Each row of $\mathcal{T}^{\overline{\Pi}^t}$ contains a distinct policy *Pol* (in the first column), the symmetric key $\Omega^{\overline{\Pi}^t}(Pol)$ (in the second column), and the set of nodes $\{n_x \in \mathcal{N}^t \mid \overline{\Pi}^t(n_x) = Pol\}$ (in the third column).

- **Tree-Encrypt**. The data owner generates a node-labeled tree $t^{\overline{\Pi}^t}$ by running Algorithm 1.2 on input of the node-labeled tree t and the protection table $\mathcal{T}^{\overline{\Pi}^t}$.

- **Key-Wrap**. The data owner replaces each symmetric key in the protection table $\mathcal{T}^{\overline{\Pi}^t}$ by the ciphertext resulting from its encryption according to the corresponding policy using the POLBE scheme.

The output of our protection enforcement mechanism consists thus of an encrypted tree $t^{\overline{\Pi}^t}$ and a protection table $\mathcal{T}^{\overline{\Pi}^t}$ containing the different encryption keys, where each key is encrypted with respect to the corresponding policy. The encrypted document and the protection table should be naturally represented in a way understandable to data consumers. In other words, a data consumer should be able to know what are the credentials and the data values it needs to collect in order to be able to successfully decrypt a specific portion of the document.

Upon receiving the encrypted tree $t^{\overline{\Pi}^t}$ and the associated protection table $\mathcal{T}^{\overline{\Pi}^t}$, a data consumer that has access to a set of credentials and data-value keys can lookup the protection table to either find the policies it is compliant with or check whether it is compliant with the policy associated to a specific portion of the document it wishes to have access to. The data consumer

Input : A node-labeled tree t , A protection table $\mathcal{T}^{\overline{\Pi}^t}$;
Output : A node-labeled tree $t^{\overline{\Pi}^t}$;
$t^{\overline{\Pi}^t} \leftarrow t$;
$l \leftarrow h^{t^{\overline{\Pi}^t}}$;
while $l \geq 0$ **do**
  foreach $n_x \in \mathcal{N}^{t^{\overline{\Pi}^t}}(l)$ **do**
    let $t^{\overline{\Pi}^t}_{n_x}$ be the node-labeled subtree of $t^{\overline{\Pi}^t}$ whose root is the node n_x **in**
    if $\overline{\Pi}^t(n_x) = false$ **then**
      remove $t^{\overline{\Pi}^t}_{n_x}$ from $t^{\overline{\Pi}^t}$;
    else
      if $\overline{\Pi}^t(n_x) = true$ **then**
        keep $t^{\overline{\Pi}^t}_{n_x}$ as it is ;
      else
        replace $t^{\overline{\Pi}^t}_{n_x}$ by a subtree representing the encryption of $t^{\overline{\Pi}^t}_{n_x}$ using $\Omega^{\overline{\Pi}^t}(\overline{\Pi}^t(n_x))$;
      end
    end
  $l \leftarrow l - 1$;
end
return $t^{\overline{\Pi}^t}$;

Algorithm 1.2: Tree encryption algorithm

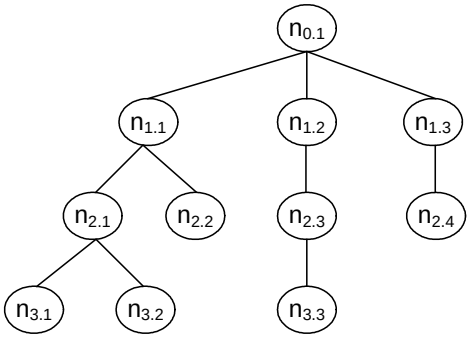
is able to access an element in cleartext if an only if it is able to access all its ancestors. In the case where the data consumer does not have access to a qualified set of credentials for a specific policy, it needs to get the missing credentials from the credential issuers specified by the data owner. Note that the decryption of certain elements within the document may allow the data consumer to update its set of data-value keys.

Remark 1.14 *As for the policy model, the formal definition of policy-based encryption schemes and the* IND-Pol-CCA *security model have to be extended to support data-value conditions. Our pairing-based implementation of policy-based encryption described in Section 1.4.1 can be easily adapted to support the new form of conditions. In fact, by setting* $\varsigma(\aleph, A) = H_0(A)$*, a data-value condition can be simply seen as a credential-based condition fulfilled by a credential for which the assertion is the data-value A and the master key of the credential issuer is equal to the identity element* $1_{\mathbb{Z}_q^*}$*. In contrast with the credentials whose secretness is based on the secretness of the issuers' master keys, the secretness of the data-value keys is based on the knowledge of the data-value A. Our reductionist security proof can also be easily adapted to include the new form of decryption keys while maintaining the same security level i.e. indistinguishability against chosen ciphertext attacks.*

An illustration of our protection enforcement mechanism is given in Example 1.10.

Example 1.10 *Given the node-labeled tree t and the associated reduced tree protection* $\overline{\Pi}^t$ *shown in Figure 1.9, the protection enforcement mechanism consists of the following stages:*

- **Initialization**. *The data owner first chooses the symmetric and policy-based encryption schemes. Then, it defines the policy-key mapping function and constructs the protection table which is shown in Figure 1.10.*



Policy	Key	Nodes
$(\langle R_1,A_1\rangle \wedge \langle R_2,A_2\rangle) \vee (\langle R_3,A_3\rangle \wedge \langle R_3,A'_3\rangle)$	k_1	$n_{1.1}$
$(\langle R_1,A'_1\rangle \vee \langle R_2,A'_2\rangle) \wedge \langle \aleph,A_0\rangle$	k_2	$n_{1.2}$
$\langle R_3,A_3\rangle$	k_3	$n_{1.3}$, $n_{2.3}$, $n_{3.2}$
$\langle R_1,A_1\rangle \wedge \langle R_2,A_2\rangle$	k_4	$n_{2.1}$
$\langle R_1,A_1\rangle \vee \langle R_2,A_2\rangle$	k_5	$n_{2.4}$
$true$	–	$n_{0.1}$, $n_{2.2}$, $n_{3.3}$
$false$	–	$n_{3.1}$

Figure 1.10: A protection table over a node-labeled tree

- **Tree-Encrypt**. *The data owner encrypts the tree according to the associated protection table as follows:*

  1. *For $l = 3$, Algorithm 1.2 removes nodes $n_{3.1}$, encrypts node $n_{3.2}$ with key k_3 and keeps node $n_{3.2}$ as it is.*
  2. *For $l = 2$, Algorithm 1.2 encrypts node $n_{2.1}$ with key k_4, node $n_{2.3}$ with key k_3, node $n_{2.4}$ with key k_5, and keeps node $n_{2.2}$ as it is.*
  3. *For $l = 1$, Algorithm 1.2 encrypts node $n_{1.1}$ with key k_1, node $n_{1.2}$ with key k_2, and node $n_{1.3}$ with key k_3.*

- **Key-Wrap**. *Each of the keys k_i (for $i = 1,\ldots,5$) is encrypted with respect to the policy to which it is associated using a policy-based encryption algorithm.*

In the following section, we describe a simple XML-based representation of the information output by our protection enforcement mechanism. In fact, the W3C Recommendation on XML Encryption Syntax and Processing (XML Encryption) specifies a standardized schema for encrypting data and representing the result in XML [1]. We leverage the processing rules of XML Encryption, while adapting its syntax to cope with the features of our protection enforcement mechanism.

1.5.5 Protection Enforcement: XML Representation

The output of our protection enforcement mechanism consists of two complementary components: on one hand, an XML document where each element is encrypted using the symmetric key corresponding to the associated authorization policy. On the other hand, the protection

table defined by the document's owner, according to which the different elements of the document are encrypted. In the following, we discuss the XML representation of each of these two components, respectively.

Our protection enforcement mechanism starts with the Initialization stage during which the data owner chooses a symmetric encryption scheme (to be used in the Tree-Encrypt stage) and a policy-based encryption scheme (to be used in the Key-Wrap stage). The XML Encryption specification supports an 'extensible' list of algorithms, each of which is defined through a brief name, an identifying URI and a level of implementation requirement. The supported algorithms are classified in different categories, among which two are of interest to us:

- **block encryption algorithms**: are symmetric encryption schemes designed for encrypting and decrypting data in fixed size, multiple octet blocks. Examples of these encryption schemes are 3DES and AES-(128-192-256). The encryption scheme chosen by the data owner should be one of the supported block encryption algorithms. For example, the data owner chooses the AES-128 algorithm, which is referenced as follows:
  - REQUIRED AES-128
    http://www.w3.org/2001/04/xmlenc#aes128-cbc
- **key transport algorithms**: are public-key schemes specified for encrypting and decrypting the symmetric keys used by block encryption algorithms. Examples are RSA-v1.5 and RSA-OAEP. To this category of algorithms we can add policy-based encryption schemes as they are used for the same purpose. The policy-based encryption scheme chosen by the data owner should be referenced and supported by the XML encryption and decryption engines. For example, the data owner can choose our POLBE scheme (described in Section 1.4.1), which can be referenced as follows:
  - REQUIRED POLBE-v1.0
    http://www.eurecom.fr/2006/01/xmlenc#polbe-v1_0

For each distinct policy defined by the protection tree, the data owner generates at random a key for the chosen symmetric encryption scheme (AES-128). Once the different keys are generated, the document is encrypted (during the Tree-Encrypt stage) according to Algorithm 1.2. That is, the different elements of the XML document are processed recursively in a way such that an element is processed only after the processing of all its descendants. An XML element is processed according to the following rules:

- if the element is associated to a 'deny-all' policy, then it is removed (with its descendants)
- if the element is associated to an 'allow-all' policy, then it is kept unchanged
- if the element is associated to a customized policy then it is encrypted using the symmetric key associated by the data owner to the policy.

Concretely, the encryption of an element using a symmetric key consists in replacing it (together with its descendants) by an <EncryptedData> element that contains two relevant components:

- ID: is an attribute that is used as a unique reference to the encrypted element in the protection table.
- <CipherData>: is a sub-element that provides the encrypted data. It contains the encrypted data represented as a Base-64 encoded text within a <CipherValue> sub-element. Base-64 is a standard encoding for transmitting binary data such as keys or digital credentials in printable textual form [1]. The encrypted data is obtained as follows: because block encryption algorithms expect input represented as a stream of bytes (octets), the data owner first converts the XML element to octets, as specified in [145] (this operation is called *serialization*). Then, the data owner runs the chosen block encryption algorithm on input of the resulting octets and the encryption key.

A example of an encrypted XML element is given in Figure 1.11.

```
<EncryptedData ID="n_{3.5}">
  <CipherData>
     <CipherValue>A23B45C56</CipherValue>
  </CipherData>
</EncryptedData>
```

Figure 1.11: An <EncryptedData> element

Remark 1.15 *The data included in the* <CipherValue> *element corresponds to the encryption of an XML element and its sub-elements. Because of the downward propagation of policies over the tree-structured XML document and the upward processing of the different elements, it might happen that the encrypted element is itself an* <EncryptedData> *element. This case is referred to as* nested encryption.

Remark 1.16 *The size of a ciphertext output by a symmetric encryption algorithm is equal to the size of the corresponding plaintext. In the case of XML encryption, the plaintext uses a text encoding with 8 bits per character (UTF-8), while the ciphertext is binary data that is represented using 6 bits per character (Base-64 encoding). As discussed in [119], this results in a blow-up of around 33% in the case of the encryption of a plaintext XML element. This becomes more problematic in the case of nested encryption, since the inflated representation of the ciphertext becomes the cleartext for another round of encryption. It is therefore critical, given a protection specification, to minimize the number of keys that need to be used and the number of nested encryptions that need to be performed in order to enforce the protection specification. In our framework, this is achieved thanks to our protection reduction algorithm presented in Section 1.5.3.*

In the Key-Wrap stage, each key in the protection table is encrypted according to the corresponding policy using the chosen policy-based encryption scheme. The standard XML Encryption recommendation defines the XML structure of an encrypted key as follows: the encrypted key is represented by an <EncryptedKey> element that contains a <CipherData> sub-element. The latter contains the encrypted symmetric key represented as a Base-64 encoded text.

```
<EncryptedKey>
   <CipherData>
      <CipherValue>C287D41C36</CipherValue>
   </CipherData>
</EncryptedKey>
```

Figure 1.12: A sample <EncryptedKey> element

Remark 1.17 *In our description of the* <EncryptedData> *and* <EncryptedKey> *elements, it is assumed that data consumers know the symmetric encryption scheme used to encrypt the different elements of the document and the policy-based encryption scheme used to encrypt the symmetric keys. In general, these elements should explicitly specify the associated encryption algorithms and the corresponding parameters. This is achieved through an* <EncryptionMethod> *sub-element which contains a reference to the used algorithm.*

The different <EncryptedKey> elements are grouped in the protection table which can be modeled in XML by a <Protection> root-element. A simplified shorthand XML schema of our <Protection> element is given in Figure 1.13.

```
<Protection>
   (<Row>
      <Policy>
      <EncryptedKey>
      (<ReferenceList>
          <ReferenceData>+
       </ReferenceList>)
    </Row>)+
</Protection>
```

Figure 1.13: A shorthand schema for the <Protection> element

The <Protection> element contains one or multiple <Row> sub-elements, each of which corresponds to a row in the considered protection table and contains the following sub-elements:

- <Policy>: this element contains a policy that can be represented as proposed in the shorthand schema depicted in Figure 1.6.
- <EncryptedKey>: this element represents a symmetric key encrypted with respect to the policy contained in the <Policy> element.

- <ReferenceList>: this element contains one or more <DataReference> sub-elements. Each <DataReference> element either contains a URI attribute or specifies a location path (XPath) pointing to an element of the XML document the access to which is regulated by the policy specified by the <Policy> element. In other words, the <ReferenceList> element groups the different portions of the document to which is associated the policy. For example, the <DataReference> element that points to the <EncryptedData> element shown in Figure 1.11 is as follows: <DataReference URI="n_{3.5}"/>.

Given an encrypted XML document and the associated <Protection> element, a data consumer that has access to a set of credentials and data-value keys can parse the different <Row> sub-elements to either find the policies it is compliant with or check whether it is compliant with the policy associated to a specific portion of the document referenced by a <DataReference> element. The data consumer is able to only decrypt the <EncryptedKey> elements associated to the <Policy> elements it is compliant with. In the case where the data consumer does not have access to a qualified set of credentials for a specific policy, it needs to get the missing credentials from the credential issuers specified by the policy. Recall that the decryption of certain elements within the XML document may allow the data consumer to update its set of data-value keys.

1.5.6 Summary

To summarize, we presented in this section a novel framework for controlling access to XML documents through policy-based encryption. Our framework allows a data owner to delegate the power of authorization to trusted third parties while enhancing the privacy of data consumers. The proposed features cannot be realized using the conventional access control models and key distribution strategies found in the literature. Our framework is presented in the context of tree-structured documents such as XML documents. However, as any document can be structured as a tree with a single root element, the proposed approach can be generally applied to any type of documents released over the Internet.

The application presented in this section illustrates the most intuitive usage of the policy-based encryption primitive i.e. as an enforcement mechanism for access control policies. In the following section, we present an application of the policy-based encryption in the context of privacy policy enforcement, which is similar but not exactly the same as classical access control. In this second application, two properties of the policy-based encryption primitive are of interest to us: the stickiness of policies and the support for cryptographic workflow.

1.6 The Sticky Privacy Policy Paradigm

In e-commerce, an increasing number of transactions cannot be achieved without revealing some privacy-sensitive information such as shipping address, billing information, personal or professional e-mail, product preference, *etc.* In a complex network like the Internet, information flows between many actors and the protection of the exchanged data against possible threats becomes a hard management problem. In this context, it is critical to define fine-grained privacy policies and to put in place effective and provably secure enforcement mechanisms. Informally, a privacy policy is an access control policy that takes into account advanced authorization features such as the purpose for which and the context during which an action is performed on a sensitive resource. In this section, we show how policy-based encryption can be used in a particular aspect of enterprise privacy policy enforcement, called the *sticky policy paradigm.*

A company can publish, through its website, a set of privacy promises to its customers explaining what data is collected, how it is used, and what other enterprises may use it. The simplest way to publish privacy promises is obviously through textual privacy statements, as shown in Figure 1.14. However, the most popular and well-adopted one consists in using the Platform for Privacy Preferences (P3P) [59]. Whereas privacy promises represent a primary approach to build privacy-aware systems, they remain not sufficient to ensure the effective protection of the exchanged data. Indeed, although P3P captures common elements of privacy policies, it does not provide the technical mechanisms that guarantee the application of the P3P statements. Such mechanisms depend on the enterprise's actual privacy practices, which are defined by the enterprise's chief privacy officer.

```
If you request something from our company's web site, for example, a product
or service, a callback, or specific marketing materials, we will use the
information you provide to fulfill your request. To help us do this, we may
share information, with others, for instance, other divisions of our company,
business partners, financial institutions, shipping companies, postal or
government authorities involved in fulfillment. We may also contact you as
part of our customer satisfaction surveys or for market research purposes.
```

Figure 1.14: Example of a text-based privacy statement

In [98], Karjoth et al. define the Platform for Enterprise Privacy Practices (E-P3P), which consists of a set of mechanisms for a privacy-enforced and fine-grained management of personally identifiable information. E-P3P allows for formalizing privacy practices in a machine readable format that can be automatically enforced within the enterprise. Moreover, it allows for identifying the specific preferences of each data owner so that these preferences could be taken into account during the privacy enforcement. Recently, IBM defined the Enterprise Privacy Authorization Language (EPAL), which is an XML-based language whose abstract syntax is close to the one defined by E-P3P [8]. The EPAL specification has lately been submitted to the World Wide Web Consortium (W3C) for public comments and possible subsequent input to standardization.

As for classical access control policies, once a privacy policy has been defined for some privacy-sensitive information, it must be enforced using an effective policy enforcement mechanism. Typically, a centralized policy engine is deployed by an enterprise. Given a request to access some privacy sensitive data, the policy engine checks whether such request is allowed or denied by the rules specified by the policy and what are the potential actions that must be fulfilled before and after the authorization decision is determined.

The *sticky policy paradigm* was first defined by Karjoth et al. in [98]. It is based on the following observation: with increasingly dynamic e-business, data is exchanged between enterprises and enterprise boundaries change due to mergers, acquisitions or virtual organizations. After transferring data from the realm of one policy into another (where the transfer must of course be permitted by the first policy), the second realm must enforce the first policy. Accordingly, the sticky policy paradigm states that once a policy is defined for some privacy-sensitive data, it must 'stick' to it and be enforced during the life cycle of the data.

The rest of this section is organized as follows: in order to make the reader more familiar with the context of enterprise privacy policies, we provide a brief overview of the EPAL policy language in Section 1.6.1. In Section 1.6.2, we discuss the notion of policy refinement and describe the current approach for implementing the sticky policy paradigm. Finally, in Section 1.6.3, we describe our approach for implementing this paradigm through policy-based encryption.

1.6.1 An Overview of EPAL

An EPAL policy is modeled by an <epal-policy> element whose relevant components are depicted in Figure 1.15. An <epal-element> consists of one or multiple <rule> elements, each of which consists of the attributes and sub-elements briefly described below:

```
<epal-policy>
   (<rule ruling>
      <data-user>+
      <data-category>+
      <purpose>+
      <action>+
      <obligation>*
      <container>*
      <condition>*
    </rule>)+
</epal-policy>
```

Figure 1.15: A shorthand schema for the <epal-policy> element

- ruling: this attribute can have three string values: 'allow', 'deny' or 'obligate'.

- <data-user>: this element identifies either an individual or a group of individuals accessing or receiving the privacy-sensitive data protected by the policy. For example, in the context of medical records, this element can contain different categories of actors within a hospital such as patients, doctors, nurses, *etc.*
- <data-category>: this element identifies a type or a sub-type of the protected data. For example, this element can contain different types of information: personal, administrative, medical, *etc.*
- <action>: this element identifies an operation that might be performed on the data. Examples are: create, store, write, disclose, read, *etc.*
- <purpose>: this element identifies a reason for which the operation defined by <action> element is performed. For example, this element may contain the following purposes: medical follow-up, treatment, research, medical record update, *etc.*
- <obligation>: this element identifies an additional action that could be mandated by the policy whenever an operation on the data is authorized. Examples of obligations are: notify the data owner, sign modification, delete within 3 months *etc.*
- <container>: this element identifies a set of external contextual information. Examples of context attributes are: urgency, user inside a surgery room, user is physician in charge*etc.*
- <condition>: this element refers to a logical combination of <container> elements. Note that before the condition contained in this element is evaluated, the input container data are validated with the container definitions. Only if all conditions in a rule are satisfied, the rule can be used. Otherwise, it is ignored.

According to the shorthand schema shown in Figure 1.15, an EPAL policy consists of one or more rules. Each rule specifies one or more users or categories of users, one or more categories of data, one or more operations on the data, and one or more purposes for which the specified actions are performed on the data. A rule applies if and only if one or more conditions depending on a logical combination of contextual information is fulfilled. Moreover, a rule specifies one or more actions, called obligations, which have to be performed once the rule is applied.

In an EPAL policy, each rule can be either an allow-rule, a deny-rule or an obligate-rule. An allow-rule or a deny-rule states that one or more users are respectively allowed or denied to perform one or more actions on data of given categories for one or more purposes whenever zero or more conditions are satisfied. An obligate-rule allows for defining a set of actions that have to be performed independently of whether the request is allowed or denied. Usually, the rules specified by the policy are ordered by precedence. The first rule has highest precedence, while the last has the lowest.

In EPAL, user's categories, data's categories and purposes are structured in hierarchies. This improves the expressiveness of rules and consequently allows to reduce the number of rules in an EPAL policy thanks to rules' inheritance. Whereas allow-rules and obligate-rules are

inherited downwards in a hierarchy, deny-rules are inherited both downwards and upwards. The reason is that hierarchies are considered groupings; if access is forbidden to an element of a group, it is also forbidden for the group as a whole.

The semantics of EPAL describe the behavior of a given EPAL policy. Given a request that consists of a quadruple (data-user, data-category, purpose, action) together with a set of context attributes, the goal is to check whether such request is allowed or denied by the policy and what are the obligations that must be fulfilled once the authorization decision is determined.

Now that we have briefly presented EPAL as an illustrative example of enterprise privacy policy languages, we address in the following section the notion of policy refinement, which is closely related to the sticky policy paradigm.

1.6.2 Privacy Policy Refinement

After transferring data from the realm of one policy into another, the policy associated to the data in the second realm must refine the first policy. Here, one policy refines another if using the first policy automatically fulfills the second policy. In other words, the second policy is at least as restrictive as the first policy. Although well-established in theory, the problem of how to efficiently check whether one policy refines another has been left open in the privacy policy literature. Recently, a practical algorithm for this task was proposed in [13]. As mentioned by the authors, the proposed algorithm deals with EPAL policies and concentrates on those aspects that make refinement of privacy policies more difficult than the refinement of classical access control policies, such as a more sophisticated treatment of deny rules and a suitable way for dealing with obligations and conditions on context information.

Given a refinement checking algorithm, a possible implementation of the sticky policy paradigm is depicted in the sample scenario shown in Figure 1.16. The transfer of privacy-sensitive information from the realm of one policy (Enterprise 1) into another (Enterprise 2) involves the following interactions:

1. Enterprise 2 requests the data from Enterprise 1.
2. Enterprise 1 asks Enterprise 2 for the policy that will be applied to the requested data.
3. Upon receiving the policy of Enterprise 2, Enterprise 1 runs the refinement checking algorithm to verify that the received policy refines the original policy associated to the requested data.
4. If the policy of Enterprise 2 refines the original policy associated to the requested data, then Enterprise 1 discloses the data. Otherwise, a fault message is returned.

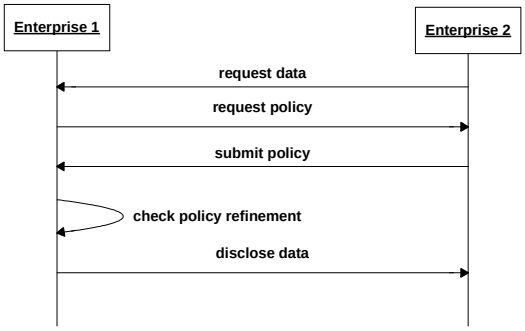


Figure 1.16: Policy enforcement prior to data disclosure

The implementation of the sticky policy paradigm described above suffers from the following shortcoming:

- **accountability**: it is hard for Enterprise 1 to make Enterprise 2 accountable for its behavior with respect to the disclosed data. Once the protected data is disclosed, Enterprise 1 has no choice other than to trust Enterprise 2 for respecting its commitment by using the advertised policy to protect the data.
- **refinement check**: the refinement checking algorithm is run by Enterprise 1. Although this approach gives Enterprise 1 a full control over the disclosure of its data, it might seem as an overhead to Enterprise 1.
- **data confidentiality**: data is transferred in cleartext from Enterprise 1 to Enterprise 2. Unless an additional encryption algorithm is used by Enterprise 1, the transferred data can be intercepted by unauthorized parties.
- **stickiness**: the proposed approach can be seen as a 'weak' implementation of the sticky policy paradigm as the policy is not strongly attached to the data during its transfer from one realm into another.

In the following section, we propose a cryptography-based implementation of the sticky policy paradigm that overcomes the shortcomings discussed above.

1.6.3 Sticky Policy through Policy-Based Cryptography

Our cryptography-based approach for implementing the sticky policy paradigm is based on the two ideas discussed below:

- **trusted authorities for refinement check**: Enterprise 2 can be made more accountable thanks to the mediation of autonomous and independent trusted third parties, which we

simply call trusted authorities. The idea here is that the refinement check is now performed by the trusted authorities, instead of being performed by the data owner. In fact, given the scenario described in the previous section, Enterprise 1 can rely on different trusted authorities, such as the Better Business Bureau (BBB) or the International Chamber of Commerce (ICC), each of which is responsible for checking the refinement of a specific portion of the privacy policy associated to the data to be disclosed. Given a specific portion of the policy of Enterprise 1 and the policy advertised by Enterprise 2, a trusted authority typically runs the refinement checking algorithm. If the refinement is valid, then the trusted authority issues a credential to Enterprise 2 certifying the validity of the refinement relationship. The issued credential is basically the signature of the trusted authority on an assertion that contains the portion of the policy of Enterprise 1 that is refined by the policy of Enterprise 2, an identifier of Enterprise 2, and additional optional information such as the date of the credential issuance. Finally, each trusted authority can trace and store all the information exchanged during these interactions in audit-trails, as evidence for future contentions or forensic analysis.

- **policy-based encryption of data**: privacy-sensitive data should never be disclosed in cleartext. The disclosed data should be encrypted in a way enforcing the privacy policy defined for the data. Again, given the scenario described in the previous section, the idea is that Enterprise 1 defines a sort of policy that consists of conjunctions and disjunctions of conditions, where each condition is fulfilled by a specific credential. Each credential is issued by a specific trusted authority that certifies the validity of a refinement relationship between the policy associated to the credential and the policy advertised by Enterprise 2. For example, assume that the EPAL policy associated by Enterprise 1 to the data to be disclosed consists of three rules: rule-1, rule-2 and rule-3. The policy can be represented as a conjunction of two EPAL policies EPAL_1 and EPAL_2, where EPAL_1 contains the rules rule-1 and rule-2 while EPAL_2 contains the rule rule-3. Enterprise 1 can require that the refinement of EPAL_1 by the policy of Enterprise 2 must be verified by either a trusted authority denoted by TA_1 or another trusted authority denoted by TA_2, while the refinement of EPAL_2 must be verified by a trusted authority denoted by TA_3. In this case, the data transferred to Enterprise 2 is encrypted (using a policy-based encryption scheme) with respect to a policy of the form:

$$(\langle \mathsf{TA}_1, \text{Enterprise 2} : \mathsf{EPAL}_1 \rangle \vee \langle \mathsf{TA}_2, \text{Enterprise 2} : \mathsf{EPAL}_1 \rangle) \wedge \langle \mathsf{TA}_3, \text{Enterprise 2} : \mathsf{EPAL}_2 \rangle$$

Given the principles discussed above, the sticky policy paradigm can be implemented using the policy-based encryption primitive as depicted in the sample scenario shown in Figure 1.17. The transfer of privacy-sensitive information from one realm (Enterprise 1) into another (Enterprise 2) involves the following interactions:

1. Enterprise 2 requests the data from Enterprise 1.
2. Enterprise 1 first encrypts the requested data with respect to the associated privacy policy using a policy-based encryption algorithm. The resulting ciphertext is then returned to Enterprise 2, instead of the data in cleartext.

In order to be able to decrypt the received ciphertext Enterprise 2 needs to have access to qualified set of credentials for the policy according to which the data is encrypted. The issuance of a credential involves the following interactions:

(a) Enterprise 2 submits a request for a credential to Trusted Authority together with its privacy policy.

(b) Trusted Authority runs the refinement checking algorithm to verify that the policy of Enterprise 2 refines the policy associated to the requested credential.

Given the example presented above, Enterprise 2 needs to send a request to TA_1 (or to TA_2) that consists of its policy and the policy EPAL_1. Assume that the refinement relationship is valid, then TA_1 returns the credential $\varsigma(\mathsf{TA}_1, \text{Enterprise 2} : \mathsf{EPAL}_1)$. Similarly, Enterprise 2 needs to get the credential $\varsigma(\mathsf{TA}_3, \text{Enterprise 2} : \mathsf{EPAL}_2)$ from TA_3.

3. Once Enterprise 2 collects a qualified set of credentials for the policy according to which the data is encrypted, it uses these credentials to decrypt the data.

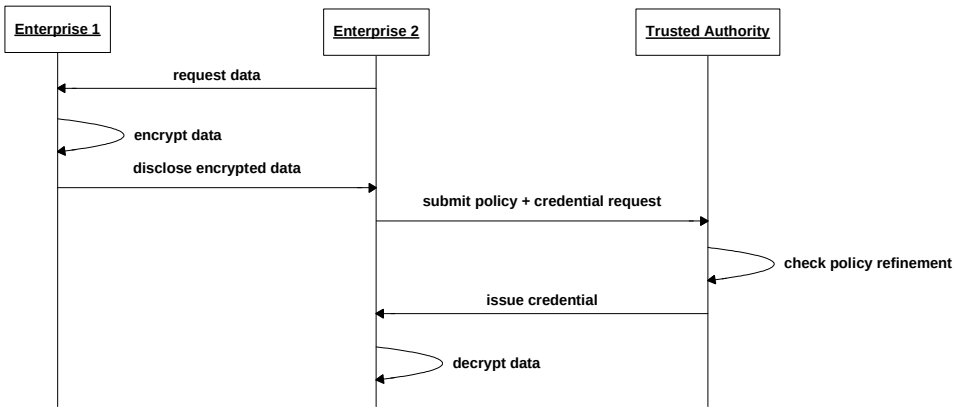


Figure 1.17: Policy enforcement after data disclosure

Our approach for implementing the sticky policy paradigm relies on two fundamental properties: on one hand, the ability to encrypt some data with respect to a policy so that only an entity that is compliant with the policy is able to decrypt the data. On the other hand, the ability to encrypt some data using public information before the private information that is required to decrypt the data is generated. Also referred to as, the support for cryptographic workflow, the second property translates the fact that, in policy-based encryption, the encryption process is independent from the generation of credentials.

The topic of privacy protection has gained an increasing interest among the research community in the last decade. In addition to privacy policy enforcement, several aspects of privacy are addressed in the literature such as anonymity, unlinkability, unobservability, pseudonymity, *etc.* In the following section, we show how the policy-based encryption primitive allows to establish ad-hoc communities with respect to the privacy principle of data minimization, according to which only strictly necessary information should be collected for a given purpose [72].

1.7 Establishment of Ad-Hoc Communities

With the advancement in wireless technologies, a new concept of networking has emerged. This is known as ad-hoc networking, where potential mobile users arrive within the common perimeter of radio link and participate in setting up the network topology for communication. Interactions within ad-hoc networks often occur between users coming from different security domains and having no pre-existing trust relationships. It is therefore a primary need to set up a security framework to ensure trustworthy communications in ad-hoc networks. In this section, we show how the policy-based encryption primitive can be used in this context.

Let's start by giving the meaning of the term 'ad-hoc communities'. A community commonly refers, in socio-economic studies, to *a group of people, that interact with each other, who have common interests or characteristics, and who live in the same locality under the governance of a set of laws* [58]. In the context of ad-hoc networking, a network can be perceived as a community of autonomous devices that can share resources with each other, provide services to each other, and collaborate in order to achieve a common goal. In order to ensure trustworthy communications within ad-hoc communities, one needs to specify a set of rules governing the different interactions within these communities. In particular, for a given community, one first needs to define a policy specifying the conditions under which an entity can be admitted as a member of the community. This is referred to as the community establishment problem.

In [99], Keoh et al. propose a comprehensive policy-based security framework supporting the establishment, evolution and management of ad-hoc networks. In Section 1.7.1, we provide an overview of their approach. Then, we leverage their policy-based trust establishment model in Section 1.7.2 and show, through the description of an application scenario, how the policy-based encryption primitive can be used to achieve a privacy-enhanced secure establishment of ad-hoc communities.

1.7.1 Policy-Based Establishment of Ad-Hoc Communities

In [99], an ad-hoc network is perceived as a community of interconnected autonomous devices providing services and resources to each other. More precisely, ad-hoc communities are defined as follows:

Definition. *An ad-hoc community interconnects a group of devices, maintains membership and ensures that only entities, i.e., users or computing services, which possess certain credentials, attribute information and characteristics can join the community (common characteristics). The members of the community rely upon each other to provide services and share resources (interactions). These interactions are regulated through a set of well-defined rules and policies (law) that govern the access to the services and resources in the community.*

With regard to their definition of ah-hoc communities, Keoh et al. introduce a community specification, called *doctrine*. A doctrine specifies a set of roles that can be associated to the participants in the community, the characteristics that participants must exhibit in order to be eligible to play a specific role, as well as the authorization and obligation policies governing the behavior of the participants within the community depending on their roles. Based on the doctrine, a set of security protocols is proposed to bootstrap the community, manage the membership (joining and leaving the community), and govern the access to the services provided by the participants.

The characteristics that a participant must fulfill in order to be eligible to play a specific role in a community are expressed in terms of a credential-based policy, called *user-role policy*, which is formalized, as for the policy-based encryption primitive, as a monotone Boolean expression. The policy is defined by the entity that initiates the bootstrapping of the community, and is broadcasted (flooded) to the other participants. The credentials considered in [99] are public-key certificates (X.509 certificates) issued by certification authorities and attribute certificates (SPKI/SDSI) issued by trusted attribute authorities.

As argued in [99], the idea of the proposed approach is not to establish trusted authorities in mobile ad-hoc networks. On the contrary, it is assumed that the participants have been already issued various certificates during their past connections to the wired environment. Such assumption is admissible in a wide range of application scenarios. For example, consider the case where the laptops and PDAs of different persons interact in an ad-hoc business meeting. Typically, the interacting devices belong to individuals from multiple domains: employees of their institutions or companies, members of collaborative projects, *etc*. In each domain, the individuals obtain credentials certifying their attributes within the domain. Recall that a credential is the signature of the credential issuer on an assertion that binds an identifier (a public key or a pseudonym) of the credential owner to the set of statements/attributes whose validity is checked and certified by the credential issuer.

Remark 1.18 *The policy model in [99] is limited to policies written in the Disjunctive Normal Form (*DNF*) but can naturally be extended to support the Conjunctive Normal Form (*CNF*). Besides, the trust model relies on a security infrastructure that consists of well-established trusted authorities in the Internet. We can extend this model to support any entity, including the participants themselves (e.g. friends, colleagues,* etc*), that is trusted to check and certify the validity of specific credentials. Finally, note that it is assumed that the entity that defines the user-role policy have access to trusted values of the public keys of the different credential issuers which are referenced in the policy.*

The community bootstrapping and community joining protocols described in [99] necessitate the verification of the compliance of the participants that want to join the community with the user-role policies associated to the roles they wish to play within the community. Such verification, as described in [99], involves the exchange of credentials, checking their validity as well as their compliance with the user-role policies. For an illustration, consider the scenario described below:

Scenario. *Alice is on a business trip for the collaborative project P. On the train there might be other colleagues from different companies working on the same project. Alice has some documents she is willing to share and possibly discuss only with the members of the project that are either from company X or from company Y.*

Following the approach proposed in [99], Alice defines a community with a role, denoted by $r_p = partner$, that allows having access to the proposed documents as well as initiating a private discussion with the different members of Alice's community. The user-role policy associated to the role r_p can be formally written, using the formalism used for the policy-based encryption in Section 1.3.1, as follows:

$$Pol_{r_p}^{ID} = [\langle I_X \ , \ ID : Employee \rangle \vee \langle I_Y \ , \ ID : Employee \rangle] \wedge \langle I_P \ , \ ID : Member \rangle$$

In policy $Pol_{r_p}^{ID}$, I_X refers to the credential issuer of company X, I_Y refers to the credential issuer of company Y, I_P refers to the credential issuer of the collaborative project P, and, finally, the attribute ID refers to the identifier of the entity that wants to join the community.

Assume that all the messages exchanged between the members of the community defined by the admission policy $Pol_{r_p}^{ID}$ and the role r_p are encrypted using a symmetric key k_{r_p} that is randomly chosen by Alice. The bootstrapping of Alice's community consists of the stages described below:

1. Alice initiates the bootstrapping of her community by flooding her policy $Pol_{r_p}^{ID}$ as well as the privileges granted by role r_p.
2. Assume that Bob, who is an employee of company X working for the collaborative project P, is interested in joining the community advertised by Alice in order to have access to the proposed documents and potentially discuss their content. In order to do so, Bob sends a join request to Alice supported by a set of credentials proving its compliance with policy $Pol_{r_p}^{ID_{Bob}}$ i.e. Bob sends his employee credential $\varsigma(I_X \ , \ ID_{Bob} : Employee)$ and his project membership credential $\varsigma(I_P \ , \ ID_{Bob} : Member)$. Upon receiving Bob's joining request, Alice first checks the validity of his credentials using the public keys of the credential issuers, then she checks that the received credentials effectively fulfill her policy. Once the admission conditions are validated, Bob is given the community's secret key k_{r_p} through a secure channel.
3. All the messages exchanged between the members of Alice's community are encrypted using the secret key k_{r_p}. A user that intercepts the messages is able to decrypt them if and only if it has been issued the secret key k_{r_p} i.e. it has been admitted to Alice's community.

The interaction between Alice and Bob is summarized in Figure 1.18.

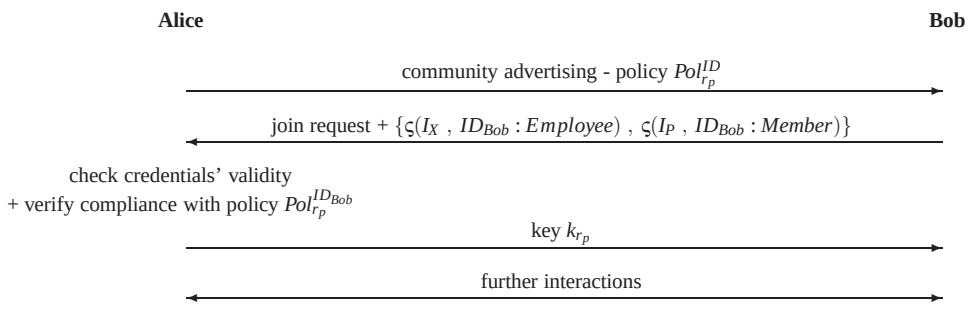


Figure 1.18: Bootstrapping of an ad-hoc community according to the approach of [99]

In the following, we discuss the shortcomings of the community establishment mechanism of [99] from the angle of privacy and propose an alternative solution using the policy-based encryption primitive.

1.7.2 Community Establishment using Policy-Based Encryption

In the scenario described above, the main concern of Alice is to ensure that the participants that are not compliant with the user-role policy $Pol_{r_p}^{ID}$ associated to role r_p cannot have the privileges given by r_p i.e. they cannot read the documents proposed by Alice and cannot initiate a private discussion within her community. In other words, the main concern of Alice is to be sure that her user-role policy is effectively enforced. Consider the two scenarios described below:

1. Assume that Bob is interested in reading the documents proposed by Alice, but it is not willing to have further interactions with the members of her community. According to the privacy principle of data minimization, the policy enforcement mechanism should not allow Alice to know whether Bob is compliant with her policy.

2. Assume that Bob is interested in reading the documents proposed by Alice as well as in having further interactions with the members of her community in order to discuss some research issues. Alice will know anyway that Bob is compliant with her policy. However, according to the privacy principle of data minimization, the policy enforcement mechanism should not allow Alice to know for which specific company Bob is working i.e. Alice should not know whether Bob is from company X or from company Y.

In the two cases described above, the standard approach proposed in [99] for policy enforcement cannot meet the privacy requirement of data minimization. In fact, because Bob must provide the credentials proving his compliance with $Pol_{r_p}^{ID_{Bob}}$, Alice will know anyway whether his is compliant with her policy and from which specific company it comes from. More generally, as long as the policy enforcement mechanism involves the exchange of digital credentials, the privacy principle of data minimization cannot be satisfied in our scenarios.

The policy-based encryption primitive can be used to overcome the shortcomings of the standard approach described above from the angle of data minimization. In the following, we describe a simple mechanism that illustrates our approach:

1. Upon receiving the join request of Bob (including an identifier specified ID_{bob} by Bob), Alice encrypts the community's secret key k_{r_p} with respect to the policy $Pol^{ID_{Bob}}$ using a policy-based encryption algorithm. Then, Alice sends the resulting ciphertext to Bob. Here, policy $Pol^{ID_{Bob}}$ is such that $Pol_{r_p}^{ID_{Bob}} = [\langle I_X \ , \ ID_{Bob} : Employee\rangle \vee \langle I_Y \ , \ ID_{Bob} : Employee\rangle] \wedge \langle I_P \ , \ ID_{Bob} : Member\rangle$.

2. Upon receiving the ciphertext, Bob uses his credentials to decrypt the community's secret key k_{r_p}. As it has access to a qualified set of credentials for the policy according to which the key was encrypted i.e. $\{\varsigma(I_X \ , \ ID_{Bob} : Employee), \varsigma(I_P \ , \ ID_{Bob} : Member)\}$, it is able to get the key k_{r_p}, which it then can use to decrypt the different documents and messages exchanged within Alice's community. In the case where Bob do not have any further interaction with the members of Alice's community, there is no way for Alice to know that Bob fulfills her policy.

3. In the case where Bob is willing to start a private discussion with the members of Alice's community, he just broadcasts his message encrypted with k_{r_p}. All the member's of Alice's community, including Alice, will be able to decrypt Bob's message. Alice will know that Bob fulfills his policy. However, as she does not know which specific credentials were used to decrypt the symmetric key k_{r_p}, she cannot know which company Bob is working for.

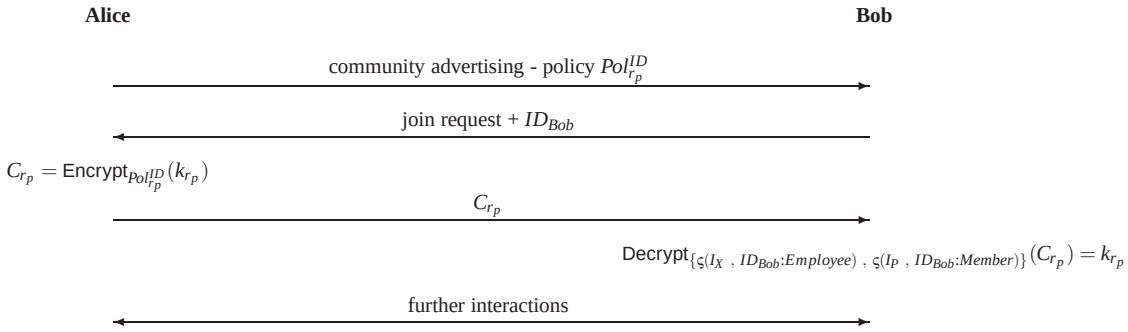


Figure 1.19: Bootstrapping of an ad-hoc community through policy-based encryption

As shown in the simple mechanism described above (summarized in Figure 1.19), the policy-based encryption primitive allows to enforce the communities' user-role policies, while adhering to the privacy principle of data minimization. This is enabled by the fact that, in contrast with the standard approach where the credentials need to be exchanged, the digital credentials are used as decryption keys in policy-based encryption.

1.8 Conclusion

In this chapter, we formalized our policy-based encryption primitive. Once we formally defined policy-based encryption and the related security model, we proposed an implementation of this primitive using bilinear pairings over elliptic curves. The functionality of policy-based encryption can be achieved using some encryption schemes found in the literature. Our scheme is not only more efficient than the existing schemes, but also provably secure under a security model that is adapted to the particular features of policy-based cryptography. The second part of this chapter is dedicated to the description of three applications of the policy-based encryption primitive. In the first application, we show how the policy-based encryption primitive can be elegantly used as a policy enforcement mechanism in the context of tree-structured documents. In the second application, we show how the policy-based encryption primitive can be used in conjunction with other tools to implement the sticky privacy policy paradigm. Finally, we show how the policy-based encryption primitive can be used to establish ad-hoc communities while adhering to the privacy principle of data minimization.

The policy-based encryption primitive presented in this chapter may suffer from collusion attacks when applied in certain contexts. In the next chapter, we propose a variant of this original encryption primitive that allows to overcome this problem.

CHAPTER 2

Collusion-Free Policy-Based Encryption

2.1 Introduction

In Chapter 1, we studied the policy-based encryption primitive that allows to encrypt a message with respect to a credential-based policy in a way such that only an entity having access to a qualified set of credentials for the policy can decrypt the message. In this chapter, we propose a variant of the policy-based encryption primitive, which we call policy-based public-key encryption. This new primitive allows to overcome the collusion problem that is inherent to the original policy-based encryption primitive.

Indeed, the policy-based encryption primitive presented in Chapter 1 relies on two trust assumptions: first, the credential issuers are not interested in spying the messages exchanged between end users. Second, end users are not willing to share their credentials with other users. While these two assumptions can be accepted in certain contexts, especially in one-to-n communication scenarios, they cannot be satisfactory in environments where the security requirements are stricter. In fact, in such environments one may consider two types of attacks against policy-based encryption:

- **collusion between credential issuers**: in addition to the legitimate holder of a qualified set of credentials, any collusion of credential issuers who collaborate to form a qualified set of credentials for the policy can also decrypt the message.

- **collusion between end users**: two or more end users can pool their credentials and decrypt a message to which neither one fulfills the policy according to which the message was encrypted.

In order to avoid collusions between end users, an intuitive solution may consist in systematically binding each issued credential to a verifiable identifier of the legitimate holder. In other words, each assertion signed by a trusted credential issuer contains, in addition to a set of statements and a set of optional details (validity period, signature algorithm, *etc*), a mandatory identifier of the entity that requests the credential from the credential issuer. The policy according to which a message is encrypted is thus such that the different assertions include a verifiable identifier of the intended recipient. Here, 'verifiable identifier' means that there exists a protocol allowing the entity that encrypts the message (the sender) to verify that the identifier specified by the policy according to which the message will be encrypted effectively corresponds to the intended recipient. In this context, it should be assumed that each identifier corresponds to one entity, whereas an entity can have multiple identifiers. Furthermore, it should be assumed that no entity is willing to share its identifiers with other entities. In other words, there exists no entity that can impersonate other entities. Here, we stress the fact that the confidentiality is not based on the identifier of the recipient the message is intended for as in identity-oriented encryption schemes but on his compliance with the policy according to which the message is encrypted. Identifiers are just used to ensure the uniqueness of the issued credentials. In the following, we describe two possible identification strategies:

- **pseudonym-based identification**: the participating entities can be identified through pseudonyms (e.g. local name, IP address, random identifier, *etc*). As in many credential systems [114, 43, 46], one may assume the existence of a pseudonym authority that controls the assignment of pseudonyms to the different entities. The pseudonym authority can play the role of a particular credential issuer that issues pseudonym credentials, where a pseudonym credential represents the signature of the pseudonym authority on the pseudonym assigned to an entity. In this case, the authenticity of an entity identified by a certain pseudonym is constrained by the possession of the corresponding pseudonym credential, which is secretly kept by the legitimate holder. An elegant approach for combining recipient authentication with policy-based encryption consists in systematically adding a condition fulfilled by the pseudonym credential of the intended recipient to the policy according to which a message has to be encrypted.
- **key-based identification**: another approach consists in assuming that each entity holds a pair of a randomly generated private key and the corresponding public key. In this case, an entity can be simply identified by its public key. Thus, the authenticity of an entity identified by a public key is constrained by the possession of the corresponding private key. The latter is assumed to be valuable, and therefore is never disclosed by its owner. ...

Binding each issued credential to a verifiable identifier of the legitimate holder is not sufficient to overcome potential collusions between credential issuers. Indeed, in order to address this

kind of attacks, in addition to the qualified set of credentials for the policy according to which a message is encrypted, the policy-based decryption algorithm must involve a secret element that is held only by the legitimate recipient. From this perspective, a private key that is secretly held by the legitimate recipient can play the role of such secret element.

Here, we present a collusion-free policy-based encryption primitive that allows to overcome the shortcomings of the original policy-based encryption primitive introduced in Chapter 1. The new primitive, called policy-based public-key encryption, combines the properties of policy-based encryption and public-key encryption. It therefore allows to encrypt a message with respect to a credential-based policy and a public key in a way such that only an entity having access not only to a qualified set of credentials for the policy but also to the private key associated to the used public key is able to decrypt the message.

An illustration of our policy-based public-key encryption primitive is shown in Figure 2.1.

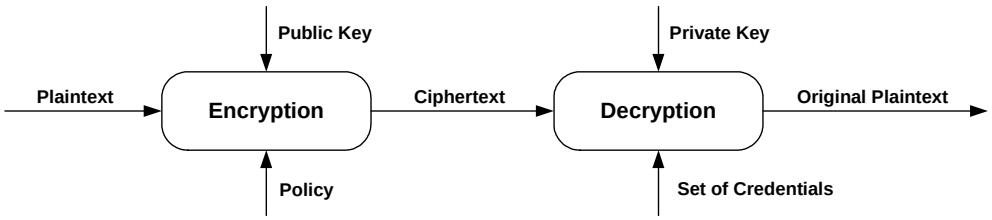


Figure 2.1: Policy-Based Public-Key Encryption

In this chapter, we propose a concrete implementation of the policy-based public-key encryption scheme using bilinear pairings over elliptic curves. While the proposed scheme is intuitively similar to the policy-based encryption scheme described in Chapter 1, a special care has to be given in order to keep the same security level i.e. semantic security against chosen ciphertext attacks, when adding the public-key functionality. The rest of the chapter is organized as follows: in Section 2.2, we discuss related work. In Section 2.3, we first present our policy model and the related terminology. Then, we formally define the policy-based public-key encryption primitive. Finally, we present the related model for indistinguishability against chosen ciphertext attacks. In Section 2.4, we first describe our pairing-based policy-based public-key encryption scheme. Then, we discuss its consistency and efficiency before proving its security in the random oracle model under the BDH assumption. Finally, in Section 2.5, we show how the policy-based public-key encryption primitive can be used in the context of automated trust negotiation. We basically suggest to replace the encryption scheme presented in [92], which suffers (as for the original policy-based encryption primitive) from the collusion property, by our policy-based public-key encryption primitive.

2.2 Related Work

The functionality of policy-based public-key encryption can be achieved using the encryption scheme presented in [4]. Besides, policy-based public-key encryption is for policy-based encryption what the concept of certificateless public-key encryption, first formalized in [5], is for identity-based encryption. These two remarks are discussed below.

Certificateless Public-Key Encryption

The identity-based encryption primitive, first defined by Shamir in [137] and recently implemented by Boneh and Franklin in [38], allows to remove the need for a public key infrastructure, replacing it by the need for a key generation center that computes and issues end users' private keys. This approach is more efficient than standard public-key encryption in terms of key management, but suffers from some shortcomings as well. The fact that the key generation center is in charge of computing the private key of an end user means that it is able to decrypt all the messages sent to that user. Thus, an honest-but-curious key generation center can read the messages of every end user in the system. This is referred to as the *key-escrow* problem. In some contexts, such as disaster recovery applications or within organizations where it is important to supervise e-mail communications, this escrow facility may be useful. However, for other applications this escrow facility is undesirable.

In [5], Al-Riyami et al. formalize the concept of certificateless public-key encryption, which allows to overcome the key-escrow problem faced by the original identity-based encryption primitive. The main idea is to combine the functionality of public-key encryption and the functionality of identity-based encryption: the encryption of a message using a certificateless public-key encryption scheme is performed with respect to the identity of the recipient as well as with respect to his public key. The authors present a pairing-based certificateless public-key encryption which is a variant of the Boneh-Franklin IBE scheme [38]. Furthermore, they define adequate security models for their encryption primitive and provide security arguments for their scheme. Our work on policy-based public-key encryption is inspired by the work presented in [5]. In fact, the encryption primitive proposed in [5] can be seen as a policy-based public-key encryption scheme for which the policies are restricted to one condition fulfilled by a credential delivered by a single credential issuer. The concept of certificateless encryption schemes is gaining an increasing interest in the research community. We refer to [61] for a survey of certificateless encryption schemes and related security models.

The concept of certificateless public-key encryption is intuitively similar to the concept of self-certified public keys, first introduced by Girault in [84] and further discussed and developed by Petersen and Horster in [125] and Saeednia in [133, 134]. The basic setting of self-certified keys is as follows: an entity first generates its private/public key pair (sk, pk). Upon receiving the entity's public key pk, a trusted authority combines it with the entity's identity to generate

a so called witness *w*. This witness *w* may correspond to the signature of the trusted authority on some combination of *pk* and the entity's identity as in [84], part of a signature as in [125], or the result of inverting a trapdoor one-way function based on *pk* and the entity's identity as in [133]. The witness *w*, which can be issued only by the trusted authority, is such that given *w*, the public key of the trusted authority and the entity's identifier, it is easy to extract the entity's public key *pk*. As for certificateless public-key encryption, self-certified public keys enable the use of public-key cryptography without certificates. However, it can be argued that the witness in a self-certified scheme is just a lightweight certificate linking an entity's identity to its public key. We refer the reader to Section 4.3 of [3] for a detailed comparison between these two concepts.

Escrow-Free Encryption Supporting Cryptographic Workflow

In [4], Al-Riyami et al. consider general access structures and use a technique similar to the one used for certificateless encryption to achieve the policy-based encryption functionality while avoiding the collusion property. The proposed scheme could be seen as the collusion-free variance of the encryption scheme proposed in [42]. They underline the fact that their scheme supports cryptographic workflow, which is a feature inherited from the Boneh-Franklin encryption primitive and naturally supported by our policy-based encryption primitive as well. Furthermore, they define formal security models to support their encryption primitive. Their 'recipient security model' considers indistinguishability against chosen plaintext attacks, where the adversary does not have access to the decryption oracle. Security against chosen ciphertext attacks was left as an open research problem. Besides, they consider policies formalized as monotone Boolean expressions represented as general conjunctions and disjunctions of atomic terms. The size of the resulting ciphertexts linearly depends on the number of these terms, whereas the normal forms considered by policy-based public-key encryption schemes, as will be shown later in this chapter, substantially reduce the size of the produced ciphertexts in addition to improving the computational cost.

2.3 Formal Definitions

2.3.1 Policy Model

In the following, we describe our policy model for policy-based public-key encryption, which is similar but not exactly the same as the one defined for policy-based encryption in Section 1.3.

On one hand, we consider a public key infrastructure where each end user holds a pair of keys (pk_u, sk_u). An end user is identified by his public key pk_u, and his private key is kept in a tamper-proof storage system such as a smart card. On the other hand, we consider a set of

credential issuers $\mathcal{I} = \{I_1, \ldots, I_N\}$, where the public key of I_κ, for $\kappa \in \{1, \ldots, N\}$, is denoted by R_κ while the corresponding master key is denoted by s_κ. We assume that a trustworthy value of the public key of each of the credential issuers is known by the end users. Any credential issuer $I_\kappa \in \mathcal{I}$ may be asked by an end user to issue a credential corresponding to a set of statements about the end user. The requested credential is the digital signature of the credential issuer on an assertion denoted by A^{pk_u}. The assertion contains, in addition to the set of statements, the end user's public key pk_u as well as a set of additional information such as the validity period of the credential.

Upon receiving a request for generating a credential on assertion A^{pk_u}, a credential issuer I_κ first checks the fact that the requester has access to the private key sk_u associated to pk_u. Then, the credential issuer checks the validity of the assertion A^{pk_u}. If it is valid, then I_κ executes a credential generation algorithm and returns a credential denoted by $\varsigma(R_\kappa, A^{pk_u})$. Otherwise, I_κ returns an error message. Upon receiving the credential $\varsigma(R_\kappa, A^{pk_u})$, the end user may check its integrity using the public key R_κ of issuer I_κ. Here, in contrast with the credentials used in policy-based encryption, a secure channel is not required for the transmission of credentials from the issuers to the requesters (unless they are considered as privacy-sensitive by the requesters). In fact, as in policy-based encryption, the credentials will play the role of decryption keys in policy-based public-key encryption. However, as will be shown later in this section, they can be used only in conjunction with the corresponding private keys which are considered to be securely held by their legitimate owners.

Remark 2.1 *As for policy-based encryption, the different assertions will be simply encoded as binary strings. Their content, representation and validation are of the scope of this section.*

A policy is formalized as monotone Boolean expressions involving conjunctions (AND / $\wedge$) and disjunctions (OR / $\vee$) of credential-based conditions. A credential-based condition is defined through a pair $\langle I_\kappa, A^{pk_u} \rangle$ specifying an assertion $A^{pk_u} \in \{0,1\}^*$ (about an end user whose public key is pk_u) and a credential issuer $I_\kappa \in \mathcal{I}$ that is trusted to check and certify the validity of A^{pk_u}. An end user whose public key is pk_u fulfills the condition $\langle I_\kappa, A^{pk_u} \rangle$ if and only if he has been issued the credential $\varsigma(R_\kappa, A^{pk_u})$.

As for policy-based encryption, we consider policies written in the conjunctive-disjunctive normal form (CDNF) i.e. a policy denoted by Pol^{pk_u} is written as follows:

$$Pol^{pk_u} = \wedge_{i=1}^{m} [\vee_{j=1}^{m_i} [\wedge_{k=1}^{m_{i,j}} \langle I_{\kappa_{i,j,k}}, A_{i,j,k}^{pk_u} \rangle]], \text{ where } I_{\kappa_{i,j,k}} \in \mathcal{I} \text{ and } A_{i,j,k}^{pk_u} \in \{0,1\}^*$$

As in Section 1.3, we define the following notations:

- $\varsigma(Pol^{pk_u})$ denotes the set of all the keys corresponding to the different conditions specified by policy Pol^{pk_u} i.e. $\varsigma(Pol^{pk_u}) = \{\{\varsigma(R_{\kappa_{i,j,k}}, A_{i,j,k}^{pk_u})\}_{k=1}^{m_{i,j}}\}_{j=1}^{m_i}\}_{i=1}^{m}$
- $\check{\varsigma}(Pol^{pk_u})$ denotes the power set of $\varsigma(Pol^{pk_u})$ i.e. the set of all the subsets of $\varsigma(Pol^{pk_u})$

- For some $\{j_i \in \{1,\ldots,m_i\}\}_{i=1}^{m}$, $\varsigma_{j_1,\ldots,j_m}(Pol^{pk_u}) = \{\{\varsigma(R_{\kappa_{i,j_i,k}}, A_{i,j_i,k}^{pk_u})\}_{k=1}^{m_{i,j_i}}\}_{i=1}^{m}$
- For a subset of credentials $\rho \subset \xi(Pol^{pk_u})$, '$\rho \vDash Pol^{pk_u}$' denotes the fact that ρ is a qualified set of credentials for policy Pol^{pk_u}

According to the notation defined above, the following equivalence holds:

$$\forall\ \rho \in \xi(Pol^{pk_u}):\ \ \rho \vDash Pol^{pk_u} \Leftrightarrow \exists\{j_i \in \{1,\ldots,m_i\}\}_{i=1}^{m}\ \text{s.th.}\ \rho = \varsigma_{j_1,\ldots,j_m}(Pol^{pk_u}) \qquad (2.1)$$

We refer to Example 1.1 for an illustration of our policy model. In the following we give our formal definition for policy-based public-key encryption schemes.

2.3.2 Policy-Based Public-Key Encryption

A formal definition of policy-based public-key encryption is given below:

Definition 2.1 *A **policy-based public-key encryption scheme**, denoted in short* POLBE$_{PK}$*, is specified by six algorithms:* Setup, Issuer-Setup, KeyGen, CredGen, Encrypt *and* Decrypt, *which we describe below.*

- **Setup**. *On input of a security parameter k, this algorithm generates the public parameters $\mathcal{P}$ which specify the different parameters, groups and public functions that will be referenced by subsequent algorithms. Furthermore, it specifies a public key space $\mathcal{K}$, a message space $\mathcal{M}$ and a ciphertext space $\mathcal{C}$.*
- **Issuer-Setup**. *This algorithm generates a random master key s_κ and the corresponding public key R_κ for credential issuer $I_\kappa \in \mathcal{I}$.*
- **KeyGen**. *This algorithm is run by an end user to generate at random a private key sk_u and the corresponding public key pk_u.*
- **CredGen**. *On input of the public key R_κ of a credential issuer $I_\kappa \in \mathcal{I}$ and an assertion $A^{pk_u} \in \{0,1\}^*$, this algorithm returns the credential $\varsigma(R_\kappa, A^{pk_u})$.*
- **Encrypt**. *On input of a message $M \in \mathcal{M}$, a public key pk_u and a policy Pol^{pk_u}, this algorithm returns a ciphertext $C \in \mathcal{C}$ representing the encryption of M with respect to policy Pol^{pk_u} and public key pk_u.*
- **Decrypt**. *On input of a ciphertext $C \in \mathcal{C}$, a private key sk and a set of credentials ρ, this algorithm returns either a message $M \in \mathcal{M}$ or $\perp$ (for '*error*').*

Remark 2.2 *In the* Decrypt *algorithm defined above, whenever* $sk \neq sk_u$ *or the set of credentials* ρ *is such that* $\rho \not\models Pol^{pk_u}$*, the output of the algorithm is* $\perp$*. To avoid this trivial case, we consider, from now on, that* $sk = sk_u$ *and the set of credentials* ρ *is such that* $\rho \models Pol^{pk_u}$*. In other words, algorithm* Decrypt *takes as input, in addition to the ciphertext C, the private key* sk_u *and a qualified set of credentials* $\varsigma_{j_1,\ldots,j_m}(Pol^{pk_u})$*, for some set of indices* $\{j_i \in \{1,\ldots,m_i\}\}_{i=1}^{m}$.

A $\mathsf{POLBE_{PK}}$ scheme has to satisfy the standard consistency constraint i.e.

$$C = \mathsf{Encrypt}_{Pol^{pk_u},pk_u}(M) \;\Rightarrow\; \mathsf{Decrypt}_{\varsigma_{j_1,\ldots,j_m}(Pol^{pk_u}),sk_u}(C) = M, \text{ for some } \{j_i \in \{1,\ldots,m_i\}\}_{i=1}^{m}$$

Remark 2.3 *We let* $\varphi_{j_1,\ldots,j_m}(C, pk_u, Pol^{pk_u})$ *be the information from C that is required to correctly perform the decryption of C with respect to policy* Pol^{pk_u} *and public key* pk_u *using the qualified set of credentials* $\varsigma_{j_1,\ldots,j_m}(Pol^{pk_u})$*. A concrete example of* $\varphi_{j_1,\ldots,j_m}(C, pk_u, Pol^{pk_u})$ *will be given when describing our pairing-based* POLBE_{PK} *scheme. Basically, the information* $\varphi_{j_1,\ldots,j_m}(C, pk_u, Pol^{pk_u})$ *will be referenced in our definition of the security model associated to* POLBE_{PK} *schemes.*

In the following, we describe our model of semantic security against chosen ciphertext attacks for $\mathsf{POLBE_{PK}}$ schemes.

2.3.3 Security Model

A $\mathsf{POLBE_{PK}}$ scheme is such that a user, be it an end user or a credential issuer, must not be able to decrypt a message if one of the two following cases occurs: i) he does not fulfill the policy according to which the message was encrypted, ii) he does not have access to the private key corresponding to the public key used to encrypt the message. Assume, for instance, that a user Alice wants to send a sensitive message to a user Bob whose public key is pk_b. Moreover, assume that Alice wants to be sure that Bob is compliant with a specific policy Pol^{pk_b} in order for Bob to be able to read the message. Thus, Alice uses a $\mathsf{POLBE_{PK}}$ scheme to encrypt her message according to her policy Pol^{pk_b} and Bob's public key pk_b. Two attack scenarios should be considered:

1. A third-party, say Charlie, who has 'somehow' access to a qualified set of credentials for Pol^{pk_b} tries to decrypt the intercepted message. For example, Charlie may represent a collusion of the different credential issuers specified by Pol^{pk_b}. As Charlie has not access to Bob's private key sk_b, he must not be able to successfully achieve the decryption. Because Charlie is not the legitimate recipient of the message he will be called *Outsider*.

2. Bob does not have access to a qualified set of credentials for policy Pol^{pk_b} and tries to illegally decrypt the message. As Bob does not fulfill Alice's policy, he must not be able

to successfully decrypt the message, although he has access to the private key sk_b. As opposed to the Outsider adversary, Bob will be called *Insider*.

According to the two scenarios described above, an Insider adversary against a $\mathsf{POLBE}_{\mathsf{PK}}$ scheme is equivalent to an adversary attacking a POLBE scheme, while an Outsider adversary against a $\mathsf{POLBE}_{\mathsf{PK}}$ scheme is equivalent to an adversary attacking a standard PKE scheme. Thus, we define indistinguishability against chosen ciphertext attacks for $\mathsf{POLBE}_{\mathsf{PK}}$ schemes in terms of an interactive game played between a challenger and an adversary. The game is denoted by $\mathsf{IND\text{-}Pol\text{-}CCA}^{\mathsf{X}}_{\mathsf{PK}}$, where $\mathsf{X} = \mathsf{I}$ for Insider adversaries and $\mathsf{X} = \mathsf{O}$ for Outsider adversaries.

A formal definition of the $\mathsf{IND\text{-}Pol\text{-}CCA}^{\mathsf{X}}_{\mathsf{PK}}$ game is given below.

Definition 2.2 *The* $\mathbf{IND\text{-}Pol\text{-}CCA}^{\boldsymbol{X}}_{\boldsymbol{PK}}$ *game consists of five stages:* Setup, Phase-1, Challenge, Phase-2 *and* Guess, *which we describe below.*

- **Setup**. *On input of a security parameter k, the challenger does the following:*
  1. *Run algorithm* Setup *to obtain the system parameters* $\mathcal{P}$ *which are given to the adversary*
  2. *Run algorithm* Issuer-Setup N *times to obtain a set of credential issuers* $\mathcal{I} = \{I_1, \ldots, I_N\}$
  3. *Run algorithm* KeyGen *to obtain a public/private key pair* (pk_{ch}, sk_{ch}).
  4. *Depending on the type of adversary, the challenger does the following:*
     (a) *If* $\mathsf{X} = \mathsf{O}$, *then the challenger gives to the adversary the public keys as well as the master keys of the credential issuers included in* $\mathcal{I}$. *Furthermore, the challenger gives the public key* pk_{ch} *to the adversary while keeping secret the private key* sk_{ch}.
     (b) *If* $\mathsf{X} = \mathsf{I}$, *then the challenger just gives to the adversary, in addition to the pair of keys* (pk_{ch}, sk_{ch}), *the public keys of the credential issuers included in* $\mathcal{I}$ *while keeping secret the corresponding master keys.*
- **Phase-1**. *The adversary performs a polynomial number of oracle queries adaptively i.e. each query may depend on the replies to the previously performed queries.*
- **Challenge**. *This stage occurs when the adversary decides that the* Phase-1 *stage is over. The adversary gives to the challenger two equal length messages* M_0, M_1 *and a policy* $Pol_{ch}^{pk_{ch}}$ *on which he wishes to be challenged. The challenger picks at random* $b \in \{0,1\}$, *then runs algorithm* Encrypt *on input of the tuple* $(M_b, pk_{ch}, Pol_{ch}^{pk_{ch}})$, *and returns the resulting ciphertext* C_{ch} *to the adversary.*
- **Phase-2**. *The adversary performs again a polynomial number of adaptive oracle queries.*
- **Guess**. *The adversary outputs a guess* b', *and wins the game if* $b = b'$.

During Phase-1 *and* Phase-2*, the adversary may perform queries to two oracles controlled by the challenger. On one hand, a credential generation oracle denoted* CredGen-O*. On the other hand, a decryption oracle denoted* Decrypt-O*. While the oracles are executed by the challenger, their input is specified by the adversary. The two oracles are defined as follows:*

- **CredGen-O**. *On input of the public key R_κ of a credential issuer $I_\kappa \in \mathcal{I}$ and an assertion A^{pk_u}, run algorithm* CredGen *on input of the tuple (R_κ, A^{pk_u}) and return the resulting credential $\varsigma(R_\kappa, A^{pk_u})$.*
- **Decrypt-O**. *On input of a ciphertext $C \in \mathcal{C}$, a policy $Pol^{pk_{ch}}$, run algorithm* CredGen *once or multiple times to obtain the qualified set of credentials $\varsigma_{j_1,\ldots,j_m}(Pol^{pk_{ch}})$, then run algorithm* Decrypt *on input of the tuple $(C, sk_{ch}, \varsigma_{j_1,\ldots,j_m}(Pol^{pk_{ch}}))$, and return the resulting output to the adversary.*

The oracle queries made by the adversary during Phase-1 *and* Phase-2 *are subject to two restrictions:*

1. *If $X = I$, the adversary is not allowed to obtain a qualified set of credentials for the policy $Pol_{ch}^{pk_{ch}}$ which he is challenged on. Note that if $X = O$, the adversary does not need to perform queries to this oracle as he has access to the credential issuers' master keys.*
2. *For both $X = I$ and $X = O$, the adversary is not allowed to perform a query to oracle* Decrypt-O *on a tuple $(C, Pol^{pk_{ch}}, \{j_1, \ldots, j_m\})$ such that $\varphi_{j_1,\ldots,j_m}(C, pk_{ch}, Pol^{pk_{ch}}) = \varphi_{j_1,\ldots,j_m}(C_{ch}, pk_{ch}, Pol_{ch}^{pk_{ch}})$ (as for* POLBE *schemes in Section 1.3.3).*

In the following, we provide a formal definition for IND-Pol-CCA$^X_{PK}$ secure POLBE$_{PK}$ schemes.

Definition 2.3 *The advantage of an adversary $\mathcal{A}^X$ in the* IND-Pol-CCA$^X_{PK}$ *game is defined to be the quantity $Adv_{\mathcal{A}^X} = |Pr[b = b'] - \frac{1}{2}|$. A* POLBE$_{PK}$ *scheme is* IND-Pol-CCA$^X_{PK}$ *secure if no probabilistic polynomial time adversary has a non-negligible advantage in the* IND-Pol-CCA$^X_{PK}$ *game.*

Now that we have formally defined policy-based public-key encryption and the related security model, we propose in the following an elegant and relatively efficient policy-based public-key encryption scheme the security of which is proved in the random oracle model.

2.4 A Pairing-Based Implementation

2.4.1 Description

Our POLBE$_{PK}$ scheme consists of the algorithms described below:

- **Setup**. On input of a security parameter k, do the following:
  1. Run algorithm BDH-Setup to obtain a tuple $(q, \mathbb{G}_1, \mathbb{G}_T, e, P)$
  2. Let $\mathcal{M} = \{0,1\}^n$, $\mathcal{K} = \mathbb{G}_1$ and $\mathcal{C} = \mathbb{G}_1 \times (\{0,1\}^n)^* \times \{0,1\}^n$ (for some $n \in \mathbb{N}^*$)
  3. Define four hash functions: $H_0 : \{0,1\}^* \to \mathbb{G}_1$, $H_1 : \{0,1\}^* \to \mathbb{Z}_q^*$, $H_2 : \{0,1\}^* \to \{0,1\}^n$ and $H_3 : \{0,1\}^* \to \{0,1\}^n$
  4. Let $\mathcal{P} = (q, \mathbb{G}_1, \mathbb{G}_T, e, P, n, H_0, H_1, H_2, H_3)$.
- **Issuer-Setup**. Let $\mathcal{I} = \{I_1, \ldots, I_N\}$ be a set of credential issuers. Each issuer $I_\kappa \in \mathcal{I}$ picks at random a secret master key $s_\kappa \in \mathbb{Z}_q^*$ and publishes the corresponding public key $R_\kappa = s_\kappa \cdot P$.
- **KeyGen**. This algorithm picks at random a private key $sk_u \in \mathbb{Z}_q^*$ and computes the corresponding public key $pk_u = sk_u \cdot P$.
- **CredGen**. On input of the public key R_κ of a credential issuer $I_\kappa \in \mathcal{I}$ and assertion $A^{pk_u} \in \{0,1\}^*$, this algorithm outputs $\varsigma(R_\kappa, A^{pk_u}) = s_\kappa \cdot H_0(A^{pk_u})$.
- **Encrypt**. On input of a message $M \in \mathcal{M}$, a public key pk_u and a policy Pol^{pk_u}, do the following:
  1. Pick at random $t_i \in \{0,1\}^n$ (for $i = 1, \ldots, m$)
  2. Compute $r = H_1(M \| t_1 \| \ldots \| t_m)$, then compute $U = r \cdot P$ and $K = r \cdot pk_u$
  3. Compute $\pi_{i,j} = \prod_{k=1}^{m_{i,j}} e(R_{\kappa_{i,j,k}}, H_0(A_{i,j,k}^{pk_u}))$ (for $j = 1, \ldots, m_i$ and $i = 1, \ldots, m$)
  4. Compute $\mu_{i,j} = H_2(K \| \pi_{i,j}^r \| i \| j)$ (for $j = 1, \ldots, m_i$ and $i = 1, \ldots, m$)
  5. Compute $v_{i,j} = t_i \oplus \mu_{i,j}$ (for $j = 1, \ldots, m_i$ and $i = 1, \ldots, m$)
  6. Compute $W = M \oplus H_3(t_1 \| \ldots \| t_m)$
  7. Return $C = (U, [[v_{i,j}]_{j=1}^{m_i}]_{i=1}^{m}, W)$
- **Decrypt**. On input of ciphertext $C = (U, [[v_{i,j}]_{j=1}^{m_i}]_{i=1}^{m}, W)$, the private key sk_u and the qualified set of credentials $\varsigma_{j_1,\ldots,j_m}(Pol^{pk_u})$, do the following:
  1. Compute $\tilde{\pi}_{i,j_i} = e(U, \sum_{k=1}^{m_{i,j_i}} \varsigma(R_{\kappa_{i,j_i,k}}, A_{i,j_i,k}^{pk_u}))$ (for $i = 1, \ldots, m$),
  2. Compute $\tilde{K} = sk_u \cdot U$
  3. Compute $\tilde{\mu}_{i,j_i} = H_2(\tilde{K} \| \tilde{\pi}_{i,j_i} \| i \| j_i)$, then compute $t_i = v_{i,j_i} \oplus \tilde{\mu}_{i,j_i}$ (for $i = 1, \ldots, m$)
  4. Compute $M = W \oplus H_3(t_1 \| \ldots \| t_m)$, then compute $r = H_1(M \| t_1 \| \ldots \| t_m)$
  5. If $U = r \cdot P$, then return the message M, otherwise return $\perp$

Remark 2.4 *The intuition behind our* Encrypt *and* Decrypt *algorithms is as follows:*

1. *Each conjunction of conditions* $\wedge_{k=1}^{m_{i,j}} \langle I_{\kappa_{i,j,k}}, A_{i,j,k}^{pk_u} \rangle$ *is first associated to a mask* $\mu_{i,j}$ *that depends not only on the credentials related to the specified conditions but also on the public key* pk_u.

2. *For each index $i \in \{1,\ldots,m\}$, a randomly chosen intermediate key t_i is associated to the disjunctive expression $\vee_{j=1}^{m_i} \wedge_{k=1}^{m_{i,j}} \langle I_{\kappa_{i,j,k}}, A_{i,j,k}^{pk_u} \rangle$.*

3. *Each intermediate key t_i is encrypted m_i times using each of the masks $\mu_{i,j}$. This way, it is sufficient to compute any one of the masks $\mu_{i,j}$ in order to be able to retrieve t_i. In order to be able to retrieve the encrypted message, an entity needs to retrieve all the intermediate keys t_i using not only a qualified set of credentials for policy Pol^{pk_u}, but also the private key sk_u corresponding to pk_u.*

Remark 2.5 *In the case of our* POLBE$_{PK}$ *scheme, $\varphi_{j_1,\ldots,j_m}(C = (U, [[v_{i,j}]_{j=1}^{m_i}]_{i=1}^{m}, W), Pol^{pk_u})$ consists of the values U and W as well as the pairs $\{(v_{i,j_i}, \wedge_{k=1}^{m_{i,j_i}} \langle I_{\kappa_{i,j_i,k}}, A_{i,j_i,k}^{pk_u} \rangle)\}_{i=1}^{m}$.*

2.4.2 Consistency and Efficiency

Our POLBE$_{\mathsf{PK}}$ scheme satisfies the consistency constraint thanks to the following statements:

- $\tilde{K} = sk_u \cdot U = sk_u \cdot (r \cdot P) = r \cdot (sk_u \cdot P) = r \cdot pk_u$
- $\tilde{\pi}_{i,j_i} = e(r \cdot P, \sum_{k=1}^{m_{i,j_i}} s_{\kappa_{i,j_i,k}} \cdot H_0(A_{i,j_i,k}^{pk_u})) = \prod_{k=1}^{m_{i,j_i}} e(s_{\kappa_{i,j_i,k}} \cdot P, H_0(A_{i,j_i,k}^{pk_u}))^r = \pi_{i,j_i}^r$

In table 2.1, we summarize the computational costs (in the worst case) of our POLBE$_{\mathsf{PK}}$ scheme. As in Section 1.4, we consider the computational costs of our algorithms in terms of the following notations: **pa** the pairing, $\mathbf{ad}_1$ the addition in the group $\mathbb{G}_1$, $\mathbf{mu}_1$ the scalar multiplication in the group $\mathbb{G}_1$, $\mathbf{mu}_T$ the multiplication in the group $\mathbb{G}_T$, $\mathbf{exp}_T$ the exponentiation in the group $\mathbb{G}_T$. Note that we ignore the costs of hash computations.

Table 2.1: Computational costs of our POLBE$_{\mathsf{PK}}$ scheme

Encrypt	$(\sum_{i=1}^{m}\sum_{j=1}^{m_i} m_{i,j}).\mathbf{pa} + 2.\mathbf{mu}_1$ $+(\sum_{i=1}^{m}\sum_{j=1}^{m_i}(m_{i,j}-1)).\mathbf{mu}_T + (\sum_{i=1}^{m} m_i).\mathbf{exp}_T$
Decrypt	$m.\mathbf{pa} + (\sum_{i=1}^{m}(\max(m_{i,j})-1)).\mathbf{ad}_1 + m.\mathbf{mu}_1$

According to Table 2.1, our encryption algorithm requires as many pairing computations as the number of conditions specified by the policy according to which the encryption is performed. Although such operation can be optimized, as explained for example in [21, 66], it still has to be minimized. Observe that for all i, j, k, the pairing $e(R_{\kappa_{i,j,k}}, H_0(A_{i,j,k}^{pk_u}))$ used in the Encrypt algorithm does not depend on the encrypted message. It can thus be pre-computed, cached and used in subsequent encryptions involving the condition $\langle I_{\kappa_{i,j,k}}, A_{i,j,k}^{pk_u} \rangle$.

Let l_1 be the bit-length of an encoding of an element of $\mathbb{G}_1$, then the bit-length of a ciphertext produced by our $\mathsf{POLBE}_{\mathsf{PK}}$ scheme is equal to: $l_1 + (\sum_{i=1}^{m} m_i).n + n$.

In Table 2.2, we compare the performance of our $\mathsf{POLBE}_{\mathsf{PK}}$ scheme with the performance of the encryption scheme proposed in [4], when it is applied to policies written in standard normal forms. Our comparison is based on two parameters: the number of pairing computations and the size of the produced ciphertexts. While the encryption algorithms require the same amount of pairing computations, our decryption algorithm is more efficient than the one proposed in [4] because $m_{i,j_i} \geq 1$ for $i = 1, \ldots, m$. Furthermore, as $m_{i,j} \geq 1$ for $j = 1, \ldots, m_i$ and $i = 1, \ldots, m$, the size of the ciphertexts resulting from our scheme is at least as short as the size of the ciphertexts produced by the scheme of [4].

Table 2.2: Performance of our $\mathsf{POLBE}_{\mathsf{PK}}$ scheme compared with the scheme of [4]

	Encryption	Decryption	Ciphertext Size
Our $\mathsf{POLBE}_{\mathsf{PK}}$ scheme	$\sum_{i=1}^{m} \sum_{j=1}^{m_i} m_{i,j}$	m	$l_1 + (\sum_{i=1}^{m} m_i).n + n$
The scheme of [4]	$\sum_{i=1}^{m} \sum_{j=1}^{m_i} m_{i,j}$	$\sum_{i=1}^{m} m_{i,j_i}$	$l_1 + (\sum_{i=1}^{m} \sum_{j=1}^{m_i} m_{i,j}).n + n$

2.4.3 Security

In the following, we study the security properties of our $\mathsf{POLBE}_{\mathsf{PK}}$ scheme. We first address the security of our scheme against Insiders' attacks (Theorem 2.1), then we consider its resistance against Outsiders' attacks (Theorem 2.2).

Remark 2.6 *As in the security analysis of our* POLBE *scheme (Section 1.4), we denote by $m_{\vee\wedge}, m_{\vee}$ and $m_{\wedge}$, respectively, the maximum values that the quantities m, m_i and $m_{i,j}$ can take respectively in the considered policies.*

Theorem 2.1 *Our POLBE_{PK} scheme is $\mathsf{IND\text{-}Pol\text{-}CCA}^{I}_{PK}$ secure in the random oracle model under the assumption that BDHP is hard.*

Proof As depicted in Figure 2.2, Theorem 2.1 follows from two reduction arguments:

1. Lemma 2.1 shows that an $\mathsf{IND\text{-}Pol\text{-}CCA}^{\mathsf{I}}_{\mathsf{PK}}$ attack on our $\mathsf{POLBE}_{\mathsf{PK}}$ scheme can be converted into an IND-CCA attack on the $\mathsf{BasicPub}^{hy_I}$ scheme (Definition 0.38).

2. Algorithm $\mathsf{BasicPub}^{hy_I}$ is shown to be IND-CCA secure in the random oracle model under the assumption that BDHP is hard (See Section 0.3.5 for more details).

□

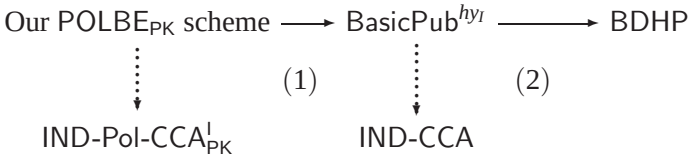


Figure 2.2: Reductionist Security for our POLBE$_{PK}$ Scheme (X=I)

Lemma 2.1 is given below. Note that the function Ψ is the one defined in Lemma 1.1 (Chapter 1).

Lemma 2.1 *Let $\mathcal{A}^\circ$ be an* IND-Pol-CCA$^{I}_{PK}$ *adversary with advantage $Adv_{\mathcal{A}^\circ} \geq \varepsilon$ when attacking our* POLBE$_{PK}$ *scheme. Assume that $\mathcal{A}^\circ$ has running time $t_{\mathcal{A}^\circ}$ and makes at most q_c queries to oracle* CredGen-O*, q_d queries to oracle* Decrypt-O *as well as q_0 queries to oracle H_0. Then, there exists an* IND-CCA *adversary $\mathcal{A}^\bullet$ the advantage of which, when attacking the* BasicPub^{hy_I} *scheme, is such that $Adv_{\mathcal{A}^\bullet} \geq \Psi(q_c, q_d, q_0, N, m_{\vee\wedge}, m_\vee, m_\wedge).\varepsilon$. Its running time is $t_{\mathcal{A}^\bullet} = O(t_{\mathcal{A}^\circ})$.*

Proof Let $\mathcal{A}^\circ$ be an IND-Pol-CCA$^{I}_{PK}$ adversary with advantage $\mathrm{Adv}_{\mathcal{A}^\circ} \geq \varepsilon$ when attacking our POLBE$_{PK}$ scheme. Assume that $\mathcal{A}^\circ$ has running time $t_{\mathcal{A}^\circ}$ and makes at most q_c queries to oracle CredGen-O, q_d queries to oracle Decrypt-O as well as q_0 queries to oracle H_0. In the following, we construct an IND-CCA adversary $\mathcal{A}^\bullet$ that uses adversary $\mathcal{A}^\circ$ to mount an attack against the BasicPub^{hy_I} scheme.

The IND-CCA game, played between the challenger and algorithm $\mathcal{A}^\bullet$, starts with the Setup$^\bullet$ stage described below.

- **Setup$^\bullet$**. Given the security parameter k, the challenger does the following:

  1. Run the Setup algorithm of the BasicPub^{hy_I} scheme, which generates the public parameters $\mathcal{P}^\star = (q, \mathbb{G}_1, \mathbb{G}_T, e, P, m^\star.n, H_1, H_2, H_3)$, for some $m^\star \in \mathbb{N}^*$
  2. Run the KeyGen algorithm of the BasicPub^{hy_I} scheme, which generates a private key $sk^\star \in \mathbb{Z}_q^*$ and the corresponding public key $pk^\star = (R^\star, Q^\star)$, where $R^\star = sk^\star \cdot P$
  3. Give the public parameters $\mathcal{P}^\star$ and the public key $pk^\star$ to adversary $\mathcal{A}^\bullet$, while keeping secret the private key $sk^\star$

  Before interacting with adversary $\mathcal{A}^\circ$, adversary $\mathcal{A}^\bullet$ does the following:

  1. Perform the same operations as the ones performed by adversary $\mathcal{A}^\bullet$ from item 1 to item 6 in Section 1.4.3.
  2. Choose two hash functions: $\bar{H}_2^\bullet : \{1, \ldots, m_{\vee\wedge}\} \to \{0,1\}^n$ and $\hat{H}_2^\bullet : \mathbb{G}_1 \to \{0,1\}^{m^\bullet.n}$
  3. Define the function $\Delta^\bullet : \{0,1\}^{m^\bullet.n} \times \{1 \ldots, m^\bullet\} \to \{0,1\}^n$ which on input of a tuple (X, i) returns the i^{th} block of length n of the binary string X i.e. the bits from $(i-1).n+1$ to $i.n$ of X.

Remark 2.7 *As in Section 1.4.3, we assume that adversaries $\mathcal{A}^\bullet$ and $\mathcal{A}^\circ$ are parameterized with the value $m^\star \in \mathbb{N}^*$. Besides, we assume that N, the number of available credential issuers, is such that $N \geq m_{\vee\wedge} m_\vee$. Our proof can be easily adapted to the case where $N \leq m_{\vee\wedge} m_\vee$.*

Remark 2.8 *Let $Pol_{cr} = \wedge_{i=1}^{m_i^\bullet} \vee_{j=1}^{m_i^\bullet} \wedge_{k=1}^{m_{i,j}^\bullet} \langle I_{\kappa_{i,j,k}^\bullet}, A_{l_{i,j,k}^\bullet} \rangle$. Policy Pol_{cr} is called the 'crucial' policy. Algorithm $\mathcal{A}^\bullet$ hopes that the 'target' policy $Pol_{ch}^{pk_{ch}}$, which will be chosen by adversary $\mathcal{A}^\circ$ in the* Challenge *stage of the* IND-Pol-CCA$^l_{PK}$ *game, is equal to policy Pol_{cr}.*

The interaction between algorithm $\mathcal{A}^\bullet$ (the challenger) and adversary $\mathcal{A}^\circ$ consists of five stages: Setup$^\circ$, Phase-1°, Challenge$^\circ$, Phase-2° and Guess$^\circ$, which we describe below.

- **Setup**$^\circ$. Algorithm $\mathcal{A}^\bullet$ does the following:

  1. Let $\mathcal{P}^\bullet = (q, \mathbb{G}_1, \mathbb{G}_T, e, P, n, H_0^\bullet, H_1, H_2^\bullet, H_3)$ be the public parameters, where $H_0^\bullet$ and $H_2^\bullet$ are controlled by algorithm $\mathcal{A}^\bullet$ and the tuple $(q, \mathbb{G}_1, \mathbb{G}_T, e, P, n, H_1, H_3)$ is taken from $\mathcal{P}^\star$. Algorithm $\mathcal{A}^\bullet$ controls $H_0^\bullet$ and $H_2^\bullet$ as follows:
     - For the random oracle $H_0^\bullet$, algorithm $\mathcal{A}^\bullet$ operates as in Section 1.4.3.
     - For the random oracle $H_2^\bullet$, on input of a tuple (K, G, i, j), algorithm $\mathcal{A}^\bullet$ returns the value $\Delta^\bullet(\hat{H}_2^\bullet(K) \oplus H_2(G^{\upsilon_{i,j}^{\bullet\,-1} \omega_i^{\bullet\,-1}}) \oplus \bar{H}_2^\bullet(j), i)$.
  2. Run algorithm KeyGen to obtain a public/private key pair $(pk_{\mathrm{ch}}, sk_{\mathrm{ch}})$
  3. Define the set of credential issuers $\mathcal{I} = \{I_1, \ldots, I_N\}$ as follows:
     - For $\kappa \in \{\kappa_{i,j,k}^\bullet\}$, the public key of I_κ is $R_\kappa = r_\kappa \cdot R^\star = (r_\kappa sk^\star) \cdot P$
     - For $\kappa \in \{1, \ldots, N\} \setminus \{\kappa_{i,j,k}^\bullet\}$, the public key of I_κ is $R_\kappa = s_\kappa \cdot P$ for some randomly chosen $s_\kappa \in \mathbb{Z}_q^*$.
  4. Give the public parameters $\mathcal{P}^\bullet$, the public/private key pair $(pk_{\mathrm{ch}}, sk_{\mathrm{ch}})$, and the credential issuers' public keys $\{R_\kappa\}_{\kappa=1}^N$ to adversary $\mathcal{A}^\circ$.

- **Phase-1**$^\circ$. Adversary $\mathcal{A}^\circ$ performs a polynomial number of oracle queries adaptively.

- **Challenge**$^\circ$. Once adversary $\mathcal{A}^\circ$ decides that Phase-1 is over, it outputs two equal length messages M_0 and M_1 as well as a policy $Pol_{\mathrm{ch}}^{pk_{\mathrm{ch}}}$ on which it wishes to be challenged. Algorithm $\mathcal{A}^\bullet$ responds as follows:

  1. If $Pol_{\mathrm{ch}}^{pk_{\mathrm{ch}}} \neq Pol_{\mathrm{cr}}$, then report failure and terminate (we refer to this event as $\mathcal{E}_{\mathrm{ch}}$)
  2. Otherwise, give the messages M_0, M_1 to the challenger who picks randomly $b \in \{0,1\}$ and returns a ciphertext $C^\star = (U, v^\star, W)$ representing the BasicPub^{hy} encryption of message M_b using the public key $pk^\star$. Upon receiving the challenger's response, compute the values $v_{i,j} = \Delta^\bullet(\hat{H}_2^\bullet(pk_{\mathrm{ch}}) \oplus v^\star \oplus \bar{H}_2^\bullet(j), i)$, then return the ciphertext $C_{\mathrm{ch}} = (U, [[v_{i,j}]_{j=1}^{m_i^\bullet}]_{i=1}^{m_i^\bullet}, W)$ to adversary $\mathcal{A}^\circ$.

- **Phase-2**$^\circ$. Adversary $\mathcal{A}^\circ$ performs a polynomial number of oracle queries adaptively.

- **Guess**$^{\circ}$. Algorithm $\mathcal{A}^{\circ}$ outputs a guess b' for b. Upon receiving b', algorithm $\mathcal{A}^{\bullet}$ outputs b' as its guess for b.

During Phase-1° and Phase-2°, adversary $\mathcal{A}^{\circ}$ can make the queries of its choice to two oracles, denoted by CredGen-O$^{\circ}$ and Decrypt-O$^{\circ}$, controlled by adversary $\mathcal{A}^{\bullet}$ as described below.

- **CredGen-O**$^{\circ}$. Assume that $\mathcal{A}^{\circ}$ makes a query on a tuple (I_{κ}, A^{pk_u}). Let $[A_{\iota}, H_{0,\iota}, \lambda_{\iota}]$ be the tuple from H_0^{list} such that $A_{\iota} = A^{pk_u}$, then algorithm $\mathcal{A}^{\bullet}$ responds as follows:
  1. If $\iota = l^{\bullet}_{i,j,1}$ and $\kappa \in \{\kappa^{\bullet}_{i,j,k}\}_{i,j,k}$, then report failure and terminate (event $\mathcal{E}_{\mathsf{cred}}$)
  2. If $\iota \neq l^{\bullet}_{i,j,1}$ and $\kappa \in \{\kappa^{\bullet}_{i,j,k}\}_{i,j,k}$, then return $(r_{\kappa}\lambda_{\iota}) \cdot R^{\star} = (r_{\kappa} s^{\star}) \cdot H_{0,\iota}$
  3. If $\kappa \in \{1, \ldots, N\} \setminus \{\kappa^{\bullet}_{i,j,k}\}_{i,j,k}$, then return $s_{\kappa} \cdot H_{0,\iota}$
- **Decrypt-O**$^{\circ}$. Assume that adversary $\mathcal{A}^{\circ}$ makes a query on a tuple $(C, Pol^{pk_{\mathrm{ch}}}, \{j_1, \ldots, j_m\})$. Then, algorithm $\mathcal{A}^{\bullet}$ responds as follows:
  1. If $Pol^{pk_{\mathrm{ch}}} \neq Pol^{pk_{\mathrm{ch}}}_{\mathrm{ch}}$ and $Pol^{pk_{\mathrm{ch}}}$ involves a condition $\langle I_{\kappa}, A^{pk_{\mathrm{ch}}} \rangle$ such that $\kappa \in \{\kappa^{\bullet}_{i,j,k}\}$ and $A^{pk_{\mathrm{ch}}} \in \{A_{l^{\bullet}_{i,j,1}}\}$, then report failure and terminate (event $\mathcal{E}_{\mathsf{dec}}$)
  2. If $Pol^{pk_{\mathrm{ch}}} \neq Pol^{pk_{\mathrm{ch}}}_{\mathrm{ch}}$ and $Pol^{pk_{\mathrm{ch}}}$ does not involve any condition $\langle I_{\kappa}, A \rangle$ such that $\kappa \in \{\kappa^{\bullet}_{i,j,k}\}$ and $A \in \{A_{l^{\bullet}_{i,j,1}}\}$, then do the following:
     (a) Run oracle CredGen-O$^{\circ}$ multiple times until obtaining the qualified set of credentials $\varsigma_{j_1,\ldots,j_m}(Pol^{pk_{\mathrm{ch}}})$
     (b) Run algorithm Decrypt on input the tuple $(C, sk_{\mathrm{ch}}, \varsigma_{j_1,\ldots,j_m}(Pol^{pk_{\mathrm{ch}}}))$ and return the resulting output back to adversary $\mathcal{A}^{\circ}$
  3. If $Pol^{pk_{\mathrm{ch}}} = Pol^{pk_{\mathrm{ch}}}_{\mathrm{ch}}$, then do the following: let $C = (U, [[v_{i,j}]_{j=1}^{m^{\bullet}_i}]_{i=1}^{m^{\bullet}}, W)$, then compute the values $v^{\bullet}_i = v_{i,j_i} \oplus \bar{H}^{\bullet}_2(j_i)$, and make a decryption query to the challenger on ciphertext $C^{\bullet} = (U, \hat{H}^{\bullet}_2(pk_{\mathrm{ch}}) \oplus (v^{\bullet}_1 \| \ldots \| v^{\bullet}_{m^{\bullet}}), W)$. Upon receiving the challenger's response, forward it to adversary $\mathcal{A}^{\circ}$

Remark 2.9 *Without loss of generality, we assume that adversary $\mathcal{A}^{\circ}$ always makes the appropriate query on assertion A^{pk_u} to the random oracle $H_0^{\bullet}$ before making any query involving A^{pk_u} to oracles* CredGen-O$^{\circ}$ *and* Decrypt-O$^{\circ}$.

We assume that adversary $\mathcal{A}^{\circ}$ respects the following restrictions when performing oracle queries during Phase-1° and Phase-2°:

1. It does not try to obtain a qualified set of credentials for policy $Pol^{pk_{\mathrm{ch}}}_{\mathrm{ch}}$
2. It does not make a query to oracle Decrypt-O$^{\circ}$ on a tuple $(C, Pol^{pk_{\mathrm{ch}}}, \{j_1, \ldots, j_m\})$ such that $\varphi_{j_1,\ldots,j_m}(C, pk_{\mathrm{ch}}, Pol^{pk_{\mathrm{ch}}}) = \varphi_{j_1,\ldots,j_m}(C_{\mathrm{ch}}, pk_{\mathrm{ch}}, Pol^{pk_{\mathrm{ch}}}_{\mathrm{ch}})$ i.e. if $C = (U, [[v_{i,j}]_{j=1}^{m_i}]_{i=1}^{m}, W)$ and $C_{\mathrm{ch}} = (U^{\mathrm{ch}}, [[v^{\mathrm{ch}}_{i,j}]_{j=1}^{m_i}]_{i=1}^{m}, W^{\mathrm{ch}})$, then we should not have

$$\begin{cases} U = U^{\mathrm{ch}}, W = W^{\mathrm{ch}} \\ \{(v_{i,j_i}, \wedge_{k=1}^{m_{i,j_i}} \langle I_{\kappa_{i,j_i,k}}, A_{i,j_i,k}^{pk_{\mathrm{ch}}} \rangle) = (v_{i,j_i}^{\mathrm{ch}}, \wedge_{k=1}^{m_{i,j_i}} \langle I_{\kappa_{i,j_i,k}}, A_{i,j_i,k}^{pk_{\mathrm{ch}}} \rangle)\}_{i=1}^{m} \end{cases}$$

In the following, we analyze the simulation described above:

If algorithm $\mathcal{A}^{\bullet}$ does not report failure during the simulation, then the view of algorithm $\mathcal{A}^{\circ}$ is identical to its view in the real attack. In fact, we observer the following:

1. The responses of algorithm $\mathcal{A}^{\bullet}$ to all queries of adversary $\mathcal{A}^{\circ}$ to oracle $H_0^{\bullet}$ are uniformly and independently distributed in group $\mathbb{G}_1$ as in the real $\mathsf{IND\text{-}Pol\text{-}CCA}^{\mathsf{I}}_{\mathsf{PK}}$ attack.
2. All the responses of algorithm $\mathcal{A}^{\bullet}$ to queries made by adversary $\mathcal{A}^{\circ}$ to oracles $\mathsf{CredGen\text{-}O}^{\circ}$ and $\mathsf{Decrypt\text{-}O}^{\circ}$ are consistent.
3. The ciphertext C_{ch} given to adversary $\mathcal{A}^{\circ}$ at the end of the $\mathsf{Challenge}^{\circ}$ stage corresponds to the encryption according to Pol_{ch} of M_b for some random $b \in \{0,1\}$, as shown in Remark 2.10.

Remark 2.10 *For adversary $\mathcal{A}^{\circ}$, the ciphertext C_{ch} represents a correct encryption of message M_b according to policy $Pol_{ch}^{pk_{ch}}$. In fact, the ciphertext $C^{\star}$ is such that $U = H_1(M_b \| t) \cdot P$, $W = M_b \oplus H_3(t)$ (for some randomly chosen $t \in \{0,1\}^{m^{\bullet}.n}$), and $v^{\star} = t \oplus H_2(g^r)$ where $g = e(R^{\star}, Q^{\star})$. Let $t_i = \Delta^{\bullet}(t,i)$, then the following holds*

$$\begin{aligned} v_{i,j} &= \Delta^{\bullet}(\hat{H}_2^{\bullet}(pk_{ch}) \oplus t \oplus H_2(e(R^{\star}, Q^{\star})^r) \oplus \bar{H}_2^{\bullet}(j), i) \\ &= \Delta^{\bullet}(t,i) \oplus \Delta^{\bullet}(\hat{H}_2^{\bullet}(pk_{ch}) \oplus H_2([e((r_{\beta_{i,j}^{\bullet}} r_{\alpha_i^{\bullet}}) \cdot R^{\star}, (r_{\beta_{i,j}^{\bullet}}^{-1} \upsilon_{i,j}^{\bullet} r_{\alpha_i^{\bullet}}^{-1} \omega_i^{\bullet}) \cdot Q^{\star})^r]^{{\upsilon_{i,j}^{\bullet}}^{-1} {\omega_i^{\bullet}}^{-1}}) \oplus \bar{H}_2^{\bullet}(j), i) \\ &= t_i \oplus H_2^{\bullet}(e((r_{\beta_{i,j}^{\bullet}} r_{\alpha_i^{\bullet}}) \cdot R^{\star}, \sum_{k=1}^{m_{i,j}^{\bullet}} r_{\gamma_{i,j,k}^{\bullet}} H_{0,l_{i,j,k}^{\bullet}})^r, i, j) = t_i \oplus H_2^{\bullet}([\prod_{k=1}^{m_{i,j}^{\bullet}} e(R_{\kappa_{i,j,k}^{\bullet}}, H_{0,l_{i,j,k}^{\bullet}})]^r, i, j) \end{aligned}$$

The remaining of the analysis is similar to the one made for our POLBE scheme in Section 1.4.3. □

Theorem 2.2 *Our POLBE_{PK} scheme is $\mathsf{IND\text{-}Pol\text{-}CCA}^{O}_{PK}$ secure in the random oracle model under the assumption that CDHP is hard.*

Proof As depicted in Figure 2.3, Theorem 2.1 follows from two reduction arguments:

1. Lemma 2.2 shows that an $\mathsf{IND\text{-}Pol\text{-}CCA}^{\mathsf{O}}_{\mathsf{PK}}$ attack on our $\mathsf{POLBE}_{\mathsf{PK}}$ scheme can be converted into an IND-CCA attack on the $\mathsf{ElG\text{-}HybridPub}^{hy_I}$ scheme (Definition 0.40).
2. Algorithm $\mathsf{ElG\text{-}BasicPub}^{hy_I}$ is shown to be IND-CCA secure in the random oracle model under the assumption that CDHP is hard (See Section 0.3.5 for more details).

□

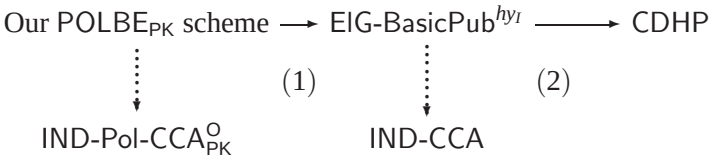


Figure 2.3: Reductionist Security for our POLBE$_{PK}$ Scheme (X=O)

Lemma 2.2 *Let $\mathcal{A}^\circ$ be an* IND-Pol-CCA$^{O}_{PK}$ *adversary with advantage $Adv_{\mathcal{A}^\circ} \geq \varepsilon$ when attacking our* POLBE$_{PK}$ *scheme. Then, there exists an* IND-CCA *adversary $\mathcal{A}^\bullet$ the advantage of which, when attacking the* EIG-BasicPub^{hy_I} *scheme, is such that $Adv_{\mathcal{A}^\bullet} \geq \varepsilon$. Its running time is $t_{\mathcal{A}^\bullet} = O(t_{\mathcal{A}^\circ})$.*

Proof Let $\mathcal{A}^\circ$ be an IND-Pol-CCA$^{O}_{PK}$ adversary with advantage $\mathrm{Adv}_{\mathcal{A}^\circ} \geq \varepsilon$ when attacking our POLBE$_{PK}$ scheme. In the following, we construct an IND-CCA adversary $\mathcal{A}^\bullet$ that uses adversary $\mathcal{A}^\circ$ to mount an attack against the EIG-BasicPub^{hy_I} scheme.

The IND-CCA game, played between the challenger and algorithm $\mathcal{A}^\bullet$, starts with the Setup$^\bullet$ stage described below.

- **Setup$^\bullet$**. Given the security parameter k, the challenger does the following:

  1. Run the Setup algorithm of the EIG-BasicPub^{hy_I} scheme, which generates the public parameters $\mathcal{P}^\star = (q, \mathbb{G}_1, \mathbb{G}_T, e, P, m^\star.n, H_1, H_2, H_3)$, for some $m^\star \in \mathbb{N}^*$
  2. Run the KeyGen algorithm of the EIG-BasicPub^{hy_I} scheme, which generates a private key $sk^\star \in \mathbb{Z}_q^*$ and the corresponding public key $pk^\star = R^\star = sk^\star \cdot P$
  3. Give the public parameters $\mathcal{P}^\star$ and the public key $pk^\star$ to adversary $\mathcal{A}^\bullet$, while keeping secret the private key $sk^\star$.

  Before interacting with adversary $\mathcal{A}^\circ$, adversary $\mathcal{A}^\bullet$ does the following:

  1. Choose a hash function $\tilde{H}_2^\bullet : \{0,1\}^* \to \{0,1\}^n$
  2. Define the function $\Delta^\bullet : \{0,1\}^{m^\bullet.n} \times \{1 \ldots, m^\bullet\} \to \{0,1\}^n$ which on input of a tuple (X,i) returns the i^{th} block of length n of the binary string X i.e. the bits from $(i-1).n+1$ to $i.n$ of X.

Remark 2.11 *As in Section 1.4.3, we assume that adversaries $\mathcal{A}^\bullet$ and $\mathcal{A}^\circ$ are parameterized with the value $m^\star$. Besides, we assume that N, the number of available credential issuers, is such that $N \geq m_{\vee\wedge} m_\vee$. Our proof can be easily adapted to the case where $N \leq m_{\vee\wedge} m_\vee$.*

The interaction between algorithm $\mathcal{A}^\bullet$ (the challenger) and adversary $\mathcal{A}^\circ$ consists of five stages: Setup$^\circ$, Phase-1°, Challenge$^\circ$, Phase-2° and Guess$^\circ$, which we describe below.

- **Setup**$^\circ$. Algorithm $\mathcal{A}^\bullet$ does the following:

  1. Let $\mathcal{P}^\bullet = (q, \mathbb{G}_1, \mathbb{G}_T, e, P, n, H_0, H_1, H_2^\bullet, H_3)$ be the public parameters, where the function $H_0 : \{0,1\}^* \to \mathbb{G}_1$ is a randomly chosen hash function, the oracle $H_2^\bullet$ is controlled by algorithm $\mathcal{A}^\bullet$, and the tuple $(q, \mathbb{G}_1, \mathbb{G}_T, e, P, n, H_1, H_3)$ is taken from $\mathcal{P}^\star$. Algorithm $\mathcal{A}^\bullet$ controls the random oracle $H_2^\bullet$ as follows: on input of (K, G, i, j), return the value $\Delta^\bullet(\hat{H}_2(K), i) \oplus \tilde{H}_2(G \| i \| j)$.
  2. Define the set of credential issuers $\mathcal{I} = \{I_1, \ldots, I_N\}$ as follows:
     - For $\kappa \in \{\kappa^\bullet_{i,j,k}\}$, the public key of I_κ is $R_\kappa = r_\kappa \cdot R^\star = (r_\kappa sk^\star) \cdot P$
     - For $\kappa \in \{1, \ldots, N\} \setminus \{\kappa^\bullet_{i,j,k}\}$, the public key of I_κ is $R_\kappa = s_\kappa \cdot P$ for some randomly chosen $s_\kappa \in \mathbb{Z}_q^*$.
  3. Give the public parameters $\mathcal{P}^\bullet$, the public key $pk_{\text{ch}} = pk^\star$ and the credential issuers' public and master keys $\{(R_\kappa, s_\kappa)\}_{\kappa=1}^N$ to adversary $\mathcal{A}^\circ$.

- **Phase-1**$^\circ$. Adversary $\mathcal{A}^\circ$ performs a polynomial number of oracle queries adaptively.

- **Challenge**$^\circ$. Once adversary $\mathcal{A}^\circ$ decides that Phase-1 is over, it outputs two equal length messages M_0 and M_1 as well as a policy $Pol_{\text{ch}}^{pk_{\text{ch}}}$ on which it wishes to be challenged. Then, the following is performed:

  1. Adversary $\mathcal{A}^\bullet$ gives the messages M_0, M_1 to the challenger who picks randomly $b \in \{0,1\}$ and returns a ciphertext $C^\star = (U, v^\star, W)$ representing the encryption of message M_b with public key $pk^\star$ using the ElG-BasicPub$^{hy_\mathcal{I}}$ scheme.
  2. Upon receiving the challenger's response, adversary $\mathcal{A}^\bullet$ computes the values $v_{i,j} = \Delta^\bullet(v^\star, i) \oplus \tilde{H}_2(\prod_{k=1}^{m_{i,j}} e(U, s_{\kappa_{i,j,k}} \cdot H_0(A_{i,j,k})) \| i \| j)$, then it forwards the ciphertext $C_{\text{ch}} = (U, [[v_{i,j}]_{j=1}^{m_i^\bullet}]_{i=1}^{m^\bullet}, W)$ to adversary $\mathcal{A}^\circ$.

- **Phase-2**$^\circ$. Adversary $\mathcal{A}^\circ$ performs a polynomial number of oracle queries adaptively.

- **Guess**$^\circ$. Algorithm $\mathcal{A}^\circ$ outputs a guess b' for b. Upon receiving b', algorithm $\mathcal{A}^\bullet$ outputs b' as its guess for b.

During Phase-1° and Phase-2°, adversary $\mathcal{A}^\circ$ can make the queries of its choice to two oracles, denoted by CredGen-O$^\circ$ and Decrypt-O$^\circ$, controlled by adversary $\mathcal{A}^\bullet$ as described below.

- **CredGen-O**$^\circ$. Since adversary $\mathcal{A}^\circ$ has access to the oracle H_0 and the master keys of the different credential issuers, it does not need to make queries to this oracle.

- **PolDec-O**$^\circ$. Assume that adversary $\mathcal{A}^\circ$ makes a query on a tuple $(C, Pol^{pk_{\text{ch}}}, \{j_1, \ldots, j_m\})$. Let $C = (U, [[v_{i,j}]_{j=1}^{m_i^\bullet}]_{i=1}^{m^\bullet}, W)$, then algorithm $\mathcal{A}^\bullet$ responds as follows:

  1. Compute the values $v_i^\bullet = v_{i,j_i} \oplus \tilde{H}_2(\prod_{k=1}^{m_{i,j_i}} e(U, s_{\kappa_{i,j_i,k}} \cdot H_0(A_{i,j_i,k}^{pk_{\text{ch}}})) \| i \| j_i)$

2. Make a decryption query to the challenger on ciphertext $C^\bullet = (U, v_1^\bullet \| \ldots \| v_{m^\bullet}^\bullet, W)$. Upon receiving the challenger's response, forward it to adversary $\mathcal{A}^\circ$.

We assume that adversary $\mathcal{A}^\circ$ respects the following restriction when performing oracle queries during Phase-1° and Phase-2°:

1. It does not make a query to oracle Decrypt-O$^\circ$ on a tuple $(C, Pol^{pk_{\text{ch}}}, \{j_1, \ldots, j_m\})$ such that $\varphi_{j_1,\ldots,j_m}(C, pk_{\text{ch}}, Pol^{pk_{\text{ch}}}) = \varphi_{j_1,\ldots,j_m}(C_{\text{ch}}, pk_{\text{ch}}, Pol_{\text{ch}}^{pk_{\text{ch}}})$ i.e. if $C = (U, [[v_{i,j}]_{j=1}^{m_i}]_{i=1}^{m}, W)$ and $C_{\text{ch}} = (U^{\text{ch}}, [[v_{i,j}^{\text{ch}}]_{j=1}^{m_i}]_{i=1}^{m}, W^{\text{ch}})$, then we should not have

$$\begin{cases} U = U^{\text{ch}}, W = W^{\text{ch}} \\ \{(v_{i,j_i}, \wedge_{k=1}^{m_{i,j_i}} \langle I_{\kappa_{i,j_i,k}}, A_{i,j_i,k}^{pk_{\text{ch}}} \rangle) = (v_{i,j_i}^{\text{ch}}, \wedge_{k=1}^{m_{i,j_i}} \langle I_{\kappa_{i,j_i,k}}, A_{i,j_i,k}^{pk_{\text{ch}}} \rangle)\}_{i=1}^{m} \end{cases}$$

In the following, we analyze the simulation described above:

In the simulation described above, the view of algorithm $\mathcal{A}^\circ$ is identical to its view in the real attack, which implies $\text{Adv}_{\mathcal{A}^\bullet} \geq \varepsilon$. In fact, we observe the following:

1. The responses of algorithm $\mathcal{A}^\bullet$ to all queries of adversary $\mathcal{A}^\circ$ to oracle $H_0^\bullet$ are uniformly and independently distributed in group $\mathbb{G}_1$ as in the real IND-Pol-CCA$_{\text{PK}}^{\text{O}}$ attack.
2. All the responses of algorithm $\mathcal{A}^\bullet$ to queries made by adversary $\mathcal{A}^\circ$ to oracles CredGen-O$^\circ$ and Decrypt-O$^\circ$ are consistent.
3. The ciphertext C_{ch} given to adversary $\mathcal{A}^\circ$ at the end of the Challenge$^\circ$ stage corresponds to the encryption according to $Pol_{\text{ch}}^{pk_{\text{ch}}}$ of M_b for some random $b \in \{0,1\}$, as shown in Remark 2.12.

Remark 2.12 *For adversary $\mathcal{A}^\circ$, the ciphertext C_{ch} represents a correct encryption of message M_b according to policy $Pol_{ch}^{pk_{ch}}$. In fact, the ciphertext $C^\star$ is such that $U = r \cdot P$ (where $r = H_1(M_b \| t)$), $W = M_b \oplus H_3(t)$ (for some randomly chosen $t \in \{0,1\}^{m^\bullet . n}$), and $v^\star = t \oplus H_2(r \cdot pk^\star)$. Let $t_i = \Delta^\bullet(t, i)$, then the following holds*

$$\begin{aligned} v_{i,j} &= \Delta^\bullet(t \oplus H_2(r \cdot pk^\star), i) \oplus \tilde{H}_2(\prod_{k=1}^{m_{i,j}} e(r \cdot P, s_{\kappa_{i,j,k}} \cdot H_0(A_{i,j,k}^{pk_{ch}})) \| i \| j) \\ &= t_i \oplus \Delta^\bullet(H_2(r \cdot pk^\star), i) \oplus \tilde{H}_2(\prod_{k=1}^{m_{i,j}} e(s_{\kappa_{i,j,k}} \cdot P, H_0(A_{i,j,k}^{pk_{ch}}))^r \| i \| j) = t_i \oplus H_2^\bullet(r \cdot pk^\star \| \pi_{i,j}^r \| i \| j) \end{aligned}$$

The rest of the analysis of the simulation described above is similar to the one given for the simulation described in Section 1.4.3.

□

Now that we have formalized the concept of policy-based public-key encryption and proposed a provably secure policy-based public-key encryption scheme using bilinear pairings over elliptic curves, we present in the following section an application of this primitive in the context of automated trust negotiation.

2.5 Automated Trust Negotiation

The concept of automated trust negotiation was first introduced by Winsborough et al. in [146]. It allows to regulate the exchange of sensitive information between entities that do not have prior knowledge of each other. It is based on the idea that, in large-scale open environments like the Internet, sensitive resources can be guarded by well-defined credential-based policies, which are policies fulfilled by digital credentials issued by trusted credential issuers. That is, a sensitive resource must not be disclosed by its owner before having a proof of the compliance of the recipient with the associated credential-based policy. In this context, even credentials are treated as potential sensitive resources, access to which is controlled through policies fulfilled by other credentials.

The traditional approach for trust negotiation consists in a bilateral exchange of digital credentials. That is, trust between communicating entities is incrementally built by iteratively disclosing digital credentials according to the associated disclosure policies [34]. Typically, a trust negotiation protocol involves two entities: on one hand, a *requester*, or the entity that initiates the interaction by requesting access to a certain resource, and, on the other hand, a *provider*, or the entity owning (or, more generally, managing access to) the requested resource. For the sake of simplicity, we omit the differentiation between requesters and providers, and consider a trust negotiation process as a peer-to-peer process, where both entities (peers) can possess sensitive resources that need to be carefully protected according to pre-defined credential-based policies.

The general setting for a trust negotiation system consists of two entities, say Alice and Bob, and a set of trusted credential issuers $\mathcal{I} = \{I_1, \ldots, I_N\}$. It is naturally assumed that a trustworthy value of the public key of each issuer $I_\kappa \in \mathcal{I}$ is known by both Alice and Bob. Note that, in addition to well-established trusted authorities, any entity (end user) that is trusted by either Alice or Bob can play the role of a credential issuer. Alice and Bob can identify each other either through pseudonyms or through public keys. In contrast with the standard approach, where pseudonyms (including names/identities) and public keys (which are bound to users' identities through public-key certification) play a central role in trust establishment, they are almost irrelevant in the context of a basic trust negotiation process that occurs between two entities without prior interactions with each other. However, they can be used in subsequent interactions between the same entities to avoid the re-verification of pre-established trust relationships. The considered identifiers (pseudonyms or public keys) are automatically included in the assertions corresponding to the credentials issued by the different credential issuers.

The rest of this section is organized as follows: in Section 2.5.1, the describe the basic negotia-

tion protocol and discuss its shortcomings. In Section 2.5.2, we describe a negotiation protocol using the policy-based public-key encryption primitive. Finally, in Section 2.5.3, we discuss the problem of concealing sensitive policies.

2.5.1 Basic Negotiation Protocol

Consider the protocol sketched by Figure 2.4.

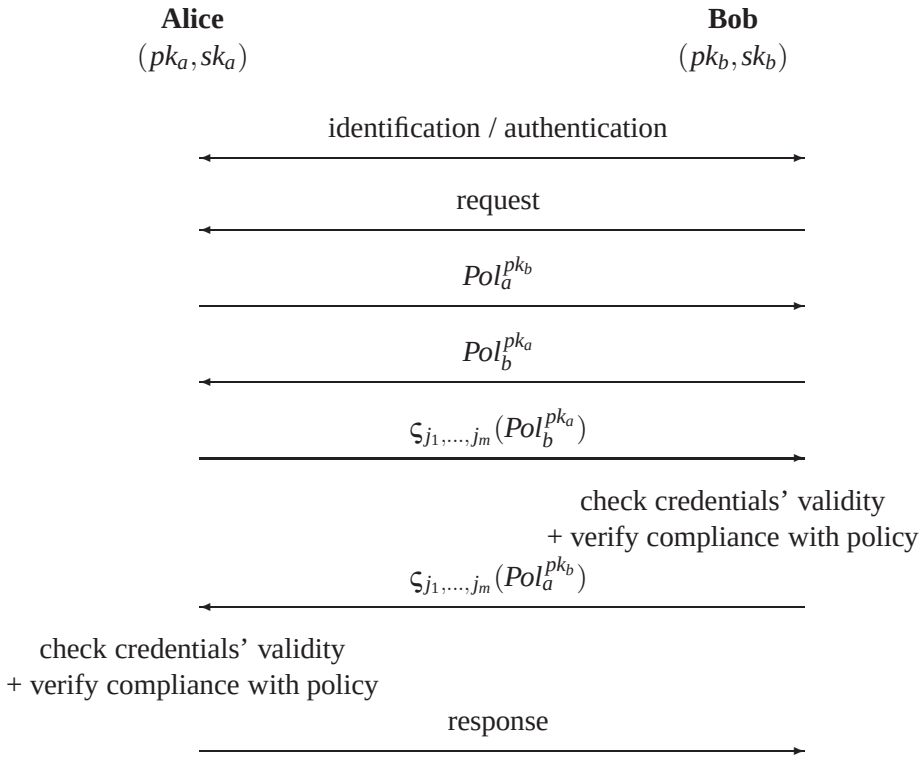


Figure 2.4: Basic negotiation protocol

The entities Alice and Bob identify each other through their public keys pk_a and pk_b respectively. The negotiation protocol is described below. Note that we use the policy model and the notational conventions defined in Section 2.3.

1. Alice and Bob start their interaction by running a mutual identification/authentication protocol. Concretely, Alice and Bob first exchange their public keys pk_a and pk_b. Then, they run a challenge-response protocol the goal of which is to ensure that each of them has access to the private key corresponding to the claimed public key i.e. Alice has access to the private key sk_a and Bob has access to the private key sk_b.

2. Bob sends his request for a sensitive resource to Alice.

3. Upon receiving Bob's request, Alice sends back a request for a qualified set of credentials for the policy $Pol_a^{pk_b}$ that is associated to the requested resource.

4. Upon receiving Alice's policy $Pol_a^{pk_b}$, Bob selects, from his credential wallet, a qualified set of credentials for $Pol_a^{pk_b}$, denoted by $\varsigma_{j_1,\ldots,j_m}(Pol_a^{pk_b})$. Then, he defines a disclosure policy $Pol_b^{pk_a}$ which he sends back to Alice. Policy $Pol_b^{pk_a}$ defines the conditions under which Bob is willing to disclose his credentials.

5. Upon receiving Bob's policy $Pol_b^{pk_a}$, Alice selects from her credential wallet a qualified set of credentials for $Pol_b^{pk_a}$, denoted by $\varsigma_{j_1,\ldots,j_m}(Pol_b^{pk_a})$. If the selected set of credentials can be disclosed without any constraint, then Alice sends it back to Bob. Otherwise, Alice sends her credential's disclosure policy back to Bob as in the stage 3 described above. In this case, Bob acts as in stage 4.

6. Upon receiving Alice's credentials, Bob first checks the validity of each of the credentials using the issuers' public key, then he checks the compliance of the set of credentials with his policy. If the different verifications are valid, Bob sends his set of credential to Alice.

7. Upon receiving Bob's credentials, Alice first checks the validity of each of the credentials using the issuers' public key, then she checks the compliance of the set of credentials with her policy. If the different verifications are valid, Alice sends (finally!) the requested resource to Bob.

As described above, the traditional approach for trust negotiation is based on the exchange of credentials such as X.509 attribute certificates [93] and SPKI certificates [67]. This approach suffers from at least two shortcomings:

- **cyclic policy interdependency**: suppose that Alice has a credential $\varsigma(R_1, A_1^{pk_a})$ that she is willing to disclose if and only if she gets a proof that Bob has credential $\varsigma(R_2, A_2^{pk_b})$, and at the same time Bob has credential $\varsigma(R_2, A_2^{pk_b})$ but is willing to disclose it if and only if he obtains a proof that Alice has credential $\varsigma(R_1, A_1^{pk_a})$. Using the traditional negotiation protocol described above, the negotiation would fail because neither $\varsigma(R_1, A_1^{pk_a})$ and $\varsigma(R_2, A_2^{pk_b})$ can be disclosed before the other. This happens even though allowing Alice and Bob to exchange both $\varsigma(R_1, A_1^{pk_a})$ and $\varsigma(R_2, A_2^{pk_b})$ would not violate their respective disclosure policies. This scenario corresponds to the so called cyclic policy interdependency problem, which clearly cannot be solved as long as the negotiation protocol is based on the exchange of digital credentials.

- **data minimization**: trust negotiation often occurs in environments where the privacy of communicating entities is critical. The privacy principle of data minimization, also referred to as the *data quality principle* [72], states that only strictly necessary information should be collected for a given purpose. From this perspective, the standard approach for trust negotiation is not optimal. In particular, consider the following scenario: assume that the policy associated to the resource requested by Bob is fulfilled either by

credential $\varsigma(R_3, A_3^{pk_b})$ or by credential $\varsigma(R_4, A_4^{pk_b})$, and assume that that Bob has access to $\varsigma(R_3, A_3^{pk_b})$. According to the data minimization principle, the trust negotiation protocol should allow Alice to know that Bob is compliant with her policy without knowing the specific credential he possesses. Obviously, this cannot be achieved as long as the negotiation protocol requires the disclosure of credentials.

A number of advanced credential systems and associated protocols can be used to overcome the limitations of the standard approach in trust negotiation. Oblivious signature-based envelopes [110, 111], hidden credentials [92, 42], and secret handshakes [20, 49] allow to address the cyclic policy interdependency problem. Furthermore, together with oblivious attribute certificates [107], private credentials [44], anonymous credentials [46, 47, 51, 115, 9] and zero-knowledge proof protocols, they can be used to address various requirements of the data minimization principle. In [109], Li et al. show that these credential systems and the associated protocols allow to address some limitations in traditional trust negotiation. However, they can be used only as fragments of a trust negotiation process. For instance, a protocol that can be used to handle cyclic policy interdependencies should be invoked only when such cycles occur during a trust negotiation process. They introduce a trust negotiation framework is which the diverse credential systems and the associated protocols can be combined, integrated, and used as needed.

In [92], Holt et al. formalize the concept of hidden credentials and propose a trust negotiation protocol that allows to overcome the two shortcomings outlined above. In the following section, we propose a further improvement of their protocol using our policy-based public-key encryption primitive.

2.5.2 Cryptography-Based Negotiation Protocol

Consider the protocol sketched by Figure 2.5. As in the basic negotiation protocol, the entities Alice and Bob identify each other through their public keys pk_a and pk_b, respectively. Their interactions are described below:

1. Bob initiates the negotiation by sending the public key pk_b corresponding to his private key sk_b. If Alice is willing to interact with Bob, she sends back the public key pk_a corresponding to her private key sk_a. The public key pk_a will be used by Bob to encrypt the request he wishes to send to Alice.

2. Bob encrypts his request with respect to policy $Pol_b^{pk_a}$ and public key pk_a using a policy-based public-key encryption algorithm. Then, he sends the resulting ciphertext C_b to Alice. Here, policy $Pol_b^{pk_a}$ specifies the conditions under which Alice is authorized to have access to the content of Bob's request.

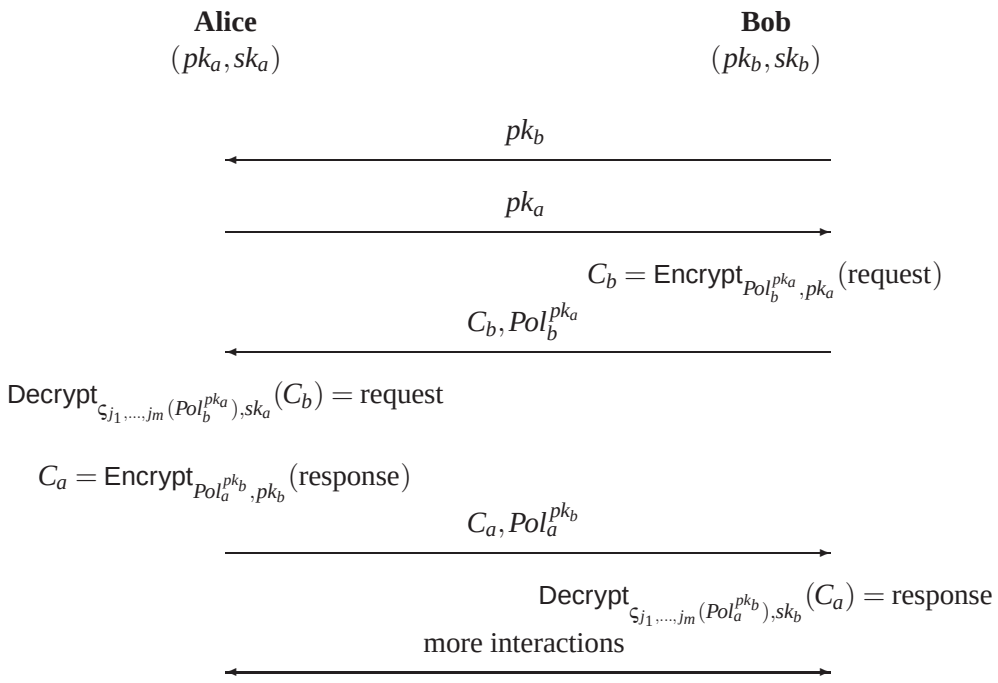


Figure 2.5: Negotiation protocol using polic-based public-key encryption

3. Upon receiving the ciphertext C_b, Alice uses her private key sk_a and a qualified set of credentials for policy $Pol_b^{pk_a}$ to decrypt it and get Bob's request in cleartext.

4. If Alice is willing to respond to Bob's request, she encrypts her response to Bob's request with respect to policy $Pol_a^{pk_b}$ and public key pk_b using a policy-based public-key encryption algorithm. Then, she sends the resulting ciphertext C_a back to Bob. Here, policy $Pol_a^{pk_b}$ specifies the conditions under which Bob is authorized to have access to the content of the response as well as the conditions under which Bob is authorized to know that Alice is compliant with policy $Pol_b^{pk_a}$.

5. Upon receiving the ciphertext C_a, Bob uses his private key sk_b and a qualified set of credentials for policy $Pol_a^{pk_b}$ to decrypt it and get Alice's response in cleartext.

6. Alice and Bob may have further interactions between each other.

The protocol described above is almost similar to the one proposed in [92]. The difference is that instead of using pseudonyms for identification purposes, we use the public keys of the communicating entities. This allows us to rely on a collusion-free policy-based encryption scheme, as compared with the encryption scheme used in [92] which suffers from potential collusion attacks.

An illustration of our negotiation protocol is given below.

Example 2.1 *Consider a set of credential issuers $\mathcal{I} = \{I_1, I_2, I_3, I_4, I_5\}$ and assume that Alice has been issued the credential $\varsigma(R_1, A_1^{pk_a})$, while Bob has been issued credentials $\varsigma(R_2, A_2^{pk_b})$, $\varsigma(R_3, A_3^{pk_b})$ and $\varsigma(R_4, A_4^{pk_b})$. Assume, for instance, that Bob wishes to have access to a sensitive resource that is under the control of Alice. Finally, assume that the trust negotiation protocol between Alice and Bob is subject to the following constraints:*

- *In order for Bob to be authorized to have access to the requested resource, he must have access to credential $\varsigma(R_3, A_3^{pk_b})$ together with either $\varsigma(R_4, A_4^{pk_b})$ or $\varsigma(R_5, A_5^{pk_b})$.*
- *In order for Alice to be able to know the request of Bob, she must have access to either credential $\varsigma(R_1, A_1^{pk_a})$ or credential $\varsigma(R_2, A_2^{pk_a})$.*
- *In order for Bob to be authorized to know the fact that Alice has access to credential $\varsigma(R_1, A_1^{pk_a})$, he must have access to credential $\varsigma(R_2, A_2^{pk_b})$.*
- *In order for Alice to be authorized to know the fact that Bob has access to credential $\varsigma(R_2, A_2^{pk_b})$, she must have access to credential $\varsigma(R_1, A_1^{pk_a})$.*

As explained above, by using the standard approach, which is based on the mutual exchange of credentials, Alice and Bob do not manage to successfully finish the negotiation protocol. Moreover, the exchange of credentials allows Alice to know the fact that Bob is compliant with the policy associated to the requested resource and the specific credentials she has access to. On the contrary, our protocol, depicted in Figure 2.6, allows to successfully achieve the negotiation while respecting the privacy principle of data minimization. Using the proposed protocol, the interactions between Alice and Bob are as follows:

1. *Bob initiates the negotiation by sending his public key pk_b. If Alice is willing to interact with Bob, she sends back her public key pk_a.*
2. *Bob encrypts his request with respect to policy $Pol_b^{pk_a} = \langle R_1, A_1^{pk_a} \rangle \vee \langle R_2, A_2^{pk_a} \rangle$ and public key pk_a using a policy-based public-key encryption algorithm. Then, he sends the resulting ciphertext C_b to Alice.*
3. *Upon receiving the ciphertext C_b, Alice uses her credential $\varsigma(R_1, A_1^{pk_a})$ and her private key sk_a to decrypt it and get Bob's request in cleartext.*
4. *If Alice is willing to respond to Bob's request, she encrypts her response to Bob's request with respect to policy $Pol_a^{pk_b} = \langle R_2, A_2^{pk_b} \rangle \wedge \langle R_3, A_3^{pk_b} \rangle \wedge (\langle R_4, A_4^{pk_b} \rangle \vee \langle R_5, A_5^{pk_b} \rangle)$ and public key pk_b using a policy-based public-key encryption algorithm. Then, she sends the resulting ciphertext C_a back to Bob.*
5. *Upon receiving the ciphertext C_a, Bob uses his private key sk_b together with the credentials $\varsigma(R_2, A_2^{pk_b})$, $\varsigma(R_3, A_3^{pk_b})$ and $\varsigma(R_4, A_4^{pk_b})$ to decrypt it and get Alice's response in cleartext.*

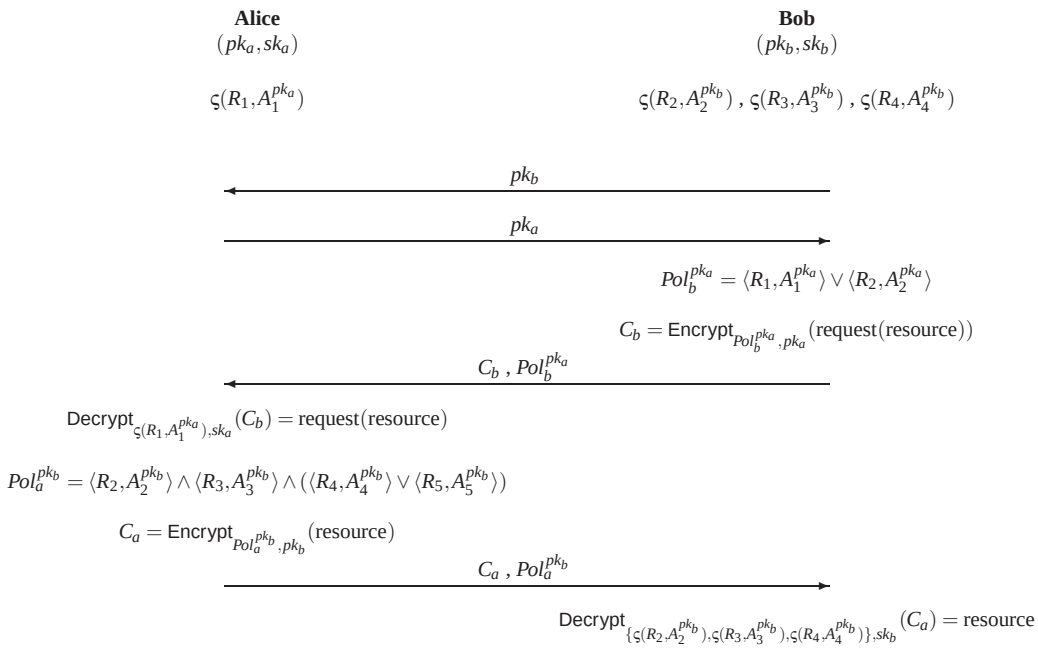


Figure 2.6: An illustration of our negotiation protocol

2.5.3 Concealing Sensitive Policies

In the negotiation protocol proposed in the previous section, Alice and Bob exchange the policies according to which their sensitive resources are encrypted. In some scenarios, the exchanged policies can be considered as sensitive resources as well. Indeed, a policy specifying a sensitive credential can be seen as a 'red flag' to attackers meaning that the resource it protects is valuable. The form of the policy or the number of conditions it contains can also leak information. As for standard resources and sensitive credentials, such policies need to be carefully protected through other policies. In some cases, the policies need even to be partially or fully hidden from other users.

In [92], Holt et al. suggest to simply hide the policy according to which a message is encrypted. Consequently, the recipient has to try the decryption with all the possible combinations of the credentials he has access to. The encryption scheme proposed by Holt et al. suffers from at least two shortcomings: 1) the size of the ciphertext allows the recipient to guess the form of the policy and the number of conditions it specifies, 2) the recipient is provided a way to recognize correct decryption of elements in the ciphertext. In [42], Bradshaw et al. describe an encryption scheme that, in addition to improving the performance of the scheme of [92], overcomes these two shortcomings.

The solution proposed in [42] still suffers from an additional shortcoming. Indeed, the resulting ciphertext allows to know the exact number of conditions specified by the policy according to which a message was encrypted. In order to overcome this shortcoming, the authors pro-

pose to add bogus elements to the ciphertext. In addition to overcoming the collusion attacks and improving the performance of the encryption and decryption algorithms, our policy-based public-key encryption scheme is such that the ciphertext does not allow to know neither the number of conditions specified by the policy nor the form of the policy.

2.6 Conclusion

In this chapter, we addressed the collusion attacks from which may suffer the policy-based encryption primitive presented in Chapter 1. We formally defined the policy-based public-key encryption primitive and the related security model. We proposed a provably secure implementation of the proposed primitive using bilinear pairings over elliptic curves. Finally, we showed how the policy-based public-key encryption primitive can be used in the context of automated trust negotiation to overcome the cyclic policy interdependency problem while adhering with the privacy principle of data minimization.

After having studied the policy-based encryption in Chapter 1 and its collusion-free version in Chapter 2, we address in the following chapter the policy-based signature primitive.

CHAPTER 3

Policy-Based Signature

3.1 Introduction

One of the most significant contributions of the concept of public-key cryptography is the digital signature primitive, whose original goal is to reproduce the handwritten signature in the digital world. Once we addressed the policy-based encryption primitive in Chapter 1 and its collusion-free version in Chapter 2, we focus in this chapter on the policy-based signature primitive. While the trustworthiness of a signature is based on the identity of the signer in standard and identity-based signature schemes, it is based on its compliance with a policy fulfilled by digital credentials in policy-based signature. The shift from the identity-oriented approach to our policy-based approach is discussed below.

A standard digital signature scheme allows an entity to generate a signature on a message in a way such that the signature is valid with respect to a public key if and only if it was generated using the corresponding private key. In other words, a valid signature proves that the message has originated from an entity having access to the private key associated to the public key according to which the validity of the signature is considered. Traditionally, the verification of the validity of a signature is often combined with the verification of a public-key certificate that guarantees the relationship between the public key according to which the signature is verified and the identity of the signing entity. Digital signature schemes together with public-key certification provide (identity-based) authentication, integrity and non-repudiation.

The concept of identity-based cryptography, first formulated by Shamir in 1984 [137], represents an elegant alternative to public-key certification. An identity-based signature scheme allows to generate a signature on a message so such that it is only valid with respect to the identity of the entity that generated it. In fact, an identity-based signature scheme is simply a signature scheme where the public key of an entity is directly derived from its identity. The corresponding private key is generated by a central authority, called private key generator. In contrast with identity-based encryption, efficient solutions for identity-based signature schemes were quickly found by the research community [71, 69]. Recently, Bellare et al. demonstrated that identity-based signature schemes can be constructed from any conventional signature scheme [30]. As for the standard digital signature primitive, an identity-based signature scheme provides both integrity and (identity-based) authentication. However, it does not provide non-repudiation because the private key of an entity is also known by the private key generator.

The identity of an entity is generally not sufficient to determining the trustworthiness of the signatures it generates. This is especially true in the context of large-scale open environments like the Internet, as in such environments, interactions often occur between entities from different security domains without pre-existing knowledge of each other. As for authorization, an increasingly popular approach consists in expressing trust requirements through policies fulfilled by digital credentials. Consequently, both standard and identity-based signature schemes need to be used in combination with a mechanism that proves the compliance of the signer with a trust establishment policy defined by the verifier of the signature. The standard approach is that the verifier of the signature first receives a set of credentials from the signer. Then, it verifies the validity of each of the received credentials, and finally checks that the received set of credentials is effectively a qualified set of credentials for the policy. According to this approach, at least two separate mechanisms are required to provide trust establishment together with integrity and non-repudiation.

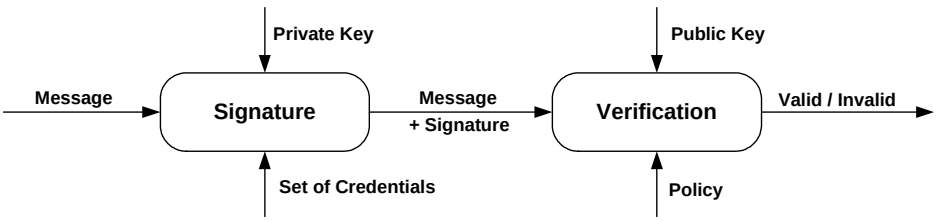


Figure 3.1: Policy-Based Signature

The goal of our policy-based signature primitive is to achieve integrity, non-repudiation and trust establishment in a logically single step. A policy-based signature scheme allows an entity to generate a signature on a message with respect to a public key and a policy so that the signature is valid if and only if the entity has access to a qualified set of credentials for the policy as well as to the private key corresponding to the used public key. The validity of the signature provides thus a proof of the compliance of the signer with the policy according to which the signature was generated. An illustration of our policy-based signature primitive is shown in Figure 3.1.

The policy-based signature primitive has to fulfill the following requirements:

- **proof of compliance**: the validity of a signature generated using a policy-based signature scheme represents of proof of compliance of the signer with the policy according to which the signature was generated. In other words, the signature is valid if and only if it was generated using a qualified set of credentials for the policy.
- **non-repudiation**: the validity of a signature generated using a policy-based signature scheme is also defined with respect to a public key. A valid signature can be generated only by an entity having access to the corresponding private key. This requirement aims at preventing an entity from denying previous commitments or actions agreed upon through the signature procedure.
- **integrity**: as for conventional digital signature, this requirement prevents a signed message from an accidental or malicious alteration during its transmission.
- **credential ambiguity**: a valid signature provides a proof that the signer is compliant with the policy according to which it was generated. In the case where the policy consists of disjunctions of conditions, the verifying entity should not be able to know which specific credentials were used to generate the signature.
- **performance**: in contrast with the policy-based encryption that can be implemented using a basic identity-based encryption scheme, a concrete implementation of the policy-based signature primitive cannot be achieved using an identity-based signature scheme. Indeed, while conjunctions can be achieved using multiple signatures, disjunctions cannot be realized. Intuitively, the disjunction structure in a policy is similar to the ring structure in ring signatures [131]. An elegant method should be found to efficiently adapt the ring structure to the context of complex policies formalized as monotone Boolean expressions written in standard normal forms.
- **provable security**: a policy-based signature scheme has to be provably secure under a well-defined strong security model that takes into consideration the specific features of policy-based signature. Here, we consider existential unforgeability against chosen message attacks in the random oracle model.

In this chapter, we formalize the concept of policy-based signature, we propose a provably secure policy-based signature scheme and prove its usefulness through the description of an original application scenario. The rest of the chapter is organized as follows: in Section 3.2 we discuss related work. In Section 3.3, we formally define the policy-based signature primitive and present the related security models, namely unforgeability against chosen message attacks and credential ambiguity. In Section 3.4, we first describe our pairing-based policy-based signature scheme. Then, we discuss its consistency and efficiency before proving its security in the random oracle model under the assumption of the hardness of the CDHP problem. Finally, in Section 3.5, we present an original form of certificates, called proof-carrying proxy certificates, which can be realized using the policy-based signature primitive.

3.2 Related Work

The policy-based signature primitive allows to sign a message with respect to a policy representing an access structure defining the conditions under which the signer of the message is trusted. A couple of advanced signature primitives dealing with different forms of access structures can be found in the literature. In this section, we discuss the relevant ones and compare them with our policy-based signature primitive.

Certificateless and Self-Certified Signatures

In [5], Al-Riyami et al. describe a certificateless signature scheme based on the identity-based signature scheme of [90], without formally proving its security. A certificateless signature scheme allows to merge the functionality of the standard digital signature primitive with the one of certification verification. In other words, it allows to perform signature verification and certification verification in a logically single step. Informally, this primitive is for the certificateless public-key encryption primitive what our policy-based signature primitive is for our policy-based public-key encryption presented in Chapter 2.

The concept of self-certified signatures, first presented in [106], shares with the policy-based signature and the certificateless signature primitive the idea of combining the functionalities of the verification of a 'standard' digital signature with the verification of certification information. Indeed, in self-certified signatures, the signer first generates a temporary signing key using his long-term signing key and his public-key certification information together. Then, it signs a message and certification information using this temporary signing key. In the verification stage, both the signature on the message and certification are checked together.

Self-certified signatures are extended to multi-certification signature whereby multiple certificates are verified together with the signature. The multi-certification signature scheme described in [106] can be seen as a policy-based signature scheme where policies are restricted to conjunctions of credentials. However, multi-certification signatures cannot support disjunctions of credentials while respecting the credential ambiguity property. In other words, the policy-based signature primitive can be seen as a generalization of self-certified signatures that supports both disjunctive and conjunctive authorization structures.

Ring Signatures and Identity-Based Ring Signatures

The concept of ring signatures was first introduced by Rivest et al. in [131]. A ring signature allows a member of an ad-hoc collection of N users, denoted by $\mathcal{U} = \{u_1, \ldots, u_N\}$, to prove that a message is signed by a user u_κ, for some $\kappa \in \{1, \ldots, N\}$. In other words, a valid ring signature represents a proof that it was either generated by u_1, or by u_2, ..., or by u_N. Furthermore, the

ring signature ensures that the verifier has no better way to guess which specific member of the set $\mathcal{U}$ is the actual author of the signature. The latter property is called *signer ambiguity* and is intuitively similar to the credential ambiguity property, which has to be fulfilled by policy-based signature schemes.

The concept of identity-based ring signatures, first addressed by Zhang in [150] and further studied by Lin et al. in [113, 89, 56], combines the concepts of identity-based cryptography [137] and ring signatures in that the public keys of the ring members are directly derived (generally through a hash function) from their respective identifiers, while the corresponding private keys are issued by a central private key generator. We refer the reader to [55] for a comprehensive summary of the so far existing work on identity-based ring signatures. As the identifiers can be replaced by any type of assertions, an identity-based ring signature might be used to construct a policy-based signature scheme where policies are restricted to disjunctions of atomic conditions fulfilled by credentials delivered by a centralized credential issuer. Some of the existing identity-based ring signature can even easily be extended to support multiple credential issuers. However, as long as the credentials (signing keys) are shared by credential issuers and legitimate holders, the non-repudiation property required by the policy-based signature primitive cannot be achieved. More precisely, a 'basic' identity-based ring signature that is used to achieve the functionality of policy-based signature (for policies limited to disjunctions of conditions) suffers from the collusion attacks as discussed in Chapter 2.

Our work on policy-based signature owes much to the research work on identity-based ring signatures in general and in particular to the solutions presented in [150, 113]. While, identity-based ring signatures and policy-based signatures are conceptually different, they somehow are functionally similar. Indeed, a policy-based signature scheme can be seen as a collusion-free version of a basic identity-based ring signature scheme that is extended to support general-form Boolean expressions written in standard normal forms. In contrast with the schemes presented in [150, 113], our policy-based signature scheme is supported by formal security models and proofs. The latter owes much to the technique presented in [88] to prove the security of the identity-based ring signature of [150]. Our reductionist proof can be easily adapted to prove the security of the signature scheme of [113].

Threshold Signatures and Identity-Based Threshold Signatures

As for threshold decryption schemes, discussed in Section 1.2, the motivation behind threshold signature is *"to share the power of a cryptosystem"*, typically in applications where a group of mutually 'suspicious' entities with potentially 'conflicting' interests must cooperate to achieve a common goal [62]. More precisely, a threshold signature scheme allows a pre-defined set of users $\mathcal{U} = \{u_1, \ldots, u_n\}$ to receive 'shares' of a private signature key in such a way that, for some parameter k such that $1 \leq k \leq n$, any subset of k users can collaborate to create a valid signature on a message, whereas any collection of $k-1$ or fewer users cannot. Threshold signatures have widely been addressed in the literature, and a number of schemes have been proposed; we refer

the reader for example to [40] for a discussion on the value of threshold signatures.

While a policy-based signature primitive allows to generate a signature on a message with respect to an access structure defined through a credential-based policy formalized as a monotone Boolean expression written in a standard normal form, a threshold signature scheme allows to generate a signature on a message according to a structure defined through a (k,n)-secret sharing scheme. A (k,n)-threshold structure can be easily expressed as a monotone Boolean expression written in a standard normal form as well. A number of identity-based threshold signature schemes have recently been proposed in the literature, such as the one described in [12]. By analogy with identity-based ring signature schemes discussed above, one might think that an identity-based threshold signature scheme can be seen as a policy-based signature scheme whereby policies are restricted to monotone Boolean expressions that are equivalent to threshold structures. This is not true because the intuition behind identity-based signature schemes is to define the group of users $\mathcal{U}$ through an identity from which will be derived the public key used to verify the validity of the generated threshold signatures, while the corresponding private key (which might be compared to a credential) is splitted according to an adequate secret sharing scheme and distributed among the users in $\mathcal{U}$. Actually, what is needed is a threshold signature scheme for which the signature is performed with respect to the identities of the members of $\mathcal{U}$ so that these identities can be replaced by the assertions specified by a policy.

Now that we have discussed related work, we formally define the policy-based signature primitive in the following section.

3.3 Formal Definitions

3.3.1 Policy Model

Our policy model for policy-based signature is exactly similar to the policy model used for the policy-based public-key encryption primitive in Section 2.3.

3.3.2 Policy-Based Signature

A formal definition of policy-based signature is given below:

Definition 3.1 *A **policy-based signature scheme**, denoted by* POLBS*, is specified by six algorithms:* Setup, Issuer-Setup, KeyGen, CredGen, Sign *and* Verify.

- **Setup**. *On input of a security parameter k, this algorithm generates the system public parameters $\mathcal{P}$ including the different spaces, groups and public functions that will be*

referenced by subsequent algorithms. Furthermore, it specifies a public key space $\mathcal{K}$, a message space $\mathcal{M}$ and a signature space $\mathcal{S}$.

- **Issuer-Setup**. *This algorithm generates a random master key s_κ and the corresponding public key R_κ for credential issuer $I_\kappa \in \mathcal{I}$.*
- **KeyGen**. *This algorithm is run by an end user to generate at random a private key sk_u and the corresponding public key pk_u.*
- **CredGen**. *On input of the public key R_κ of a credential issuer $I_\kappa \in \mathcal{I}$ and an assertion $A^{pk_u} \in \{0,1\}^*$, this algorithm returns the credential $\varsigma(R_\kappa, A^{pk_u})$.*
- **Sign**. *On input of a message M, a private key sk and a set of credentials ρ, this algorithm returns a signature σ.*
- **Verify**. *On input of a message M, a signature σ, a public key pk_u and a policy Pol^{pk_u}, this algorithm returns either $\top$ (for* valid*) or $\bot$ (for* invalid*).*

Remark 3.1 *In the* Sign *algorithm defined above, whenever $sk \neq sk_u$ or the set of credentials ρ is such that $\rho \not\models Pol^{pk_u}$, the output of the* Verify *algorithm is $\bot$. To avoid this trivial case, we consider, from now on, that $sk = sk_u$ and the set of credentials ρ is such that $\rho \models Pol^{pk_u}$. In other words, algorithm* Sign *takes as input the private key sk_u and a qualified set of credentials $\varsigma_{j_1,\ldots,j_m}(Pol^{pk_u})$, for some set of indices $\{j_i \in \{1,\ldots,m_i\}\}_{i=1}^m$.*

A POLBS scheme has to satisfy the standard consistency constraint i.e.

$$\sigma = \mathsf{Sign}_{\varsigma_{j_1,\ldots,j_m}(Pol^{pk_u}),sk_u}(M) \;\Rightarrow\; \mathsf{Verify}_{Pol^{pk_u}),pk_u}(M,\sigma) = \top, \text{ for some } \{j_i \in \{1,\ldots,m_i\}\}_{i=1}^m$$

In the following, we describe our security models for POLBS schemes.

3.3.3 Security Model

As for standard digital signature schemes, a POLBS scheme has to fulfill the security requirement of unforgeability. Besides, in order to be compliant with the privacy principle of data minimization, a POLBS scheme has to fulfill the credential ambiguity property. In the following, we formally address the two requirements respectively.

The standard acceptable notion of security for standard signature schemes is existential unforgeability against chosen message attacks [86]. Therefore, we require the same security notion for POLBS schemes. The definition of existential unforgeability should naturally be adapted to the particular setting of policy-based signature. A POLBS scheme is such that a user, be it an end user or a credential issuer, must not be able to generate a valid signature on a message if one of

the two following cases occurs: i) he does not fulfill the policy according to which the signature has to be generated, ii) he does not have access to the private key corresponding to the public key that will be used to verify the validity of the signature. Assume, for instance, that a user Alice, receives a message, from user Bob whose public key is pk_b, that is signed with respect to a policy Pol^{pk_b} using a POLBS scheme. Two attack scenarios should be considered:

1. A third-party, say Charlie, who has 'somehow' access to a qualified set of credentials for Pol^{pk_b} tries to generate a valid signature on the message on behalf of Bob. For example, Charlie may represent a collusion of the different credential issuers specified by Pol^{pk_b}. As Charlie has not access to Bob's private key sk_b, he must not be able to successfully generate a valid signature. Because Charlie is not the legitimate signer of the message he will be called *Outsider*.

2. Bob does not have access to a qualified set of credentials for policy Pol^{pk_b} and tries to generate a valid signature on the message according to Pol^{pk_b}. As Bob does not fulfill Pol^{pk_b}, he must not be able to successfully generate a valid signature, although he has access to the private key sk_b. As opposed to the Outsider adversary, Bob will be called *Insider*.

Thus, we define existential unforgeability for POLBS schemes in terms of an interactive game, denoted EUF-Pol-CMA$^{\mathsf{X}}$ (where $\mathsf{X} = \mathsf{I}$ for Insider adversaries and $\mathsf{X} = \mathsf{O}$ for Outsider adversaries), played between a challenger and an adversary. A formal definition of the EUF-Pol-CMA$^{\mathsf{X}}$ game is given below.

Definition 3.2 *The* **EUF-Pol-CMA^{X}** *game consists of three stages:* Setup*,* Queries *and* Forge*, which we describe below:*

- **Setup**. *On input of a security parameter k, the challenger does the following*
  1. *Run algorithm* Setup *to obtain the system public parameters* $\mathcal{P}$
  2. *Run algorithm* Issuer-Setup *N times to obtain a set of credential issuers* $\mathcal{I} = \{I_1, \ldots, I_N\}$
  3. *Run algorithm* KeyGen *to obtain a public/private key pair* (pk_f, sk_f)
  4. *Give to the adversary the parameters* $\mathcal{P}$*, the public key* pk_f *and the public keys of the different credential issuers included in* $\mathcal{I}$.

- **Queries**. *The adversary performs adaptively a polynomial number of oracle queries which we define below. By "adaptively", we mean that each query may depend on the challenger's replies to the previously performed queries.*

- **Forge**. *Once the adversary decides that the* Queries *stage is over, it outputs a message* M_f*, a policy* $Pol_f^{pk_f}$ *and a signature* σ_f*, and wins the game if* $\mathsf{Verify}_{Pol_f^{pk_f}, pk_f}(M_f, \sigma_f) = \top$.

During the Queries *stage, the adversary may perform queries to two oracles controlled by the challenger. On one hand, a credential generation oracle denoted by* **CredGen-O**. *On the other hand, a signature oracle denoted by* **Sign-O**. *While the oracles are executed by the challenger, their input is specified by the adversary. The two oracles are defined as follows:*

- **CredGen-O**. *On input of the public key R_κ of a credential issuer $I_\kappa \in \mathcal{I}$ and an assertion A^{pk_u}, run algorithm* CredGen *on input of the tuple (R_κ, A^{pk_u}) and return the resulting credential $\varsigma(R_\kappa, A^{pk_u})$.*
- **Sign-O**. *On input of a message M and a policy Pol^{pk_f} and a set of indices $\{j_1, \ldots, j_m\}$, first run algorithm* CredGen *once or multiple times to obtain the qualified set of credentials $\varsigma_{j_1,\ldots,j_m}(Pol^{pk_f})$, then run algorithm* Sign *on input of the tuple $(M, sk_f, \varsigma_{j_1,\ldots,j_m}(Pol^{pk_f}))$ and return the resulting output to the adversary.*

The oracle queries made by the adversary during the Queries *stage are subject to some restrictions depending on the type of adversary. In fact, we distinguish two types of attackers:*

- *for $X = I$, the adversary is given, in addition to the parameters provided by the challenger during* Setup*, the private key sk_f. An adversary of this type is not allowed to obtain (through queries to oracle* CredGen-O*) a qualified set of credentials for the policy $Pol_f^{pk_f}$.*
- *for $X = O$, the adversary is given, in addition to the parameters provided by the challenger during* Setup*, the master keys of the different credential issuers included in $\mathcal{I}$. An adversary of this type does not have access to the private key sk_f and do not need to perform queries to* CredGen-O.

Obviously, an adversary, be it insider or outsider, is not allowed to perform a query to oracle Sign-O *on the tuple $(M_f, Pol_f^{pk_f})$.*

In the following, we provide a formal definition for EUF-Pol-CMA^X secure POLBS schemes.

Definition 3.3 *The advantage of an adversary $\mathcal{A}^X$ in the* EUF-Pol-CMA^X *game is defined to be the quantity $Adv_{\mathcal{A}^X} = Pr[\mathcal{A}^X \text{ wins}]$. A* POLBS *scheme is* EUF-Pol-CMA^X *secure if no probabilistic polynomial time adversary has a non-negligible advantage in the* EUF-Pol-CMA^X *game.*

Intuitively, the credential ambiguity property that should be fulfilled by POLBS schemes is equivalent to the signer-ambiguity property supported by ring signature schemes [131]. Formally, we define credential ambiguity against chosen message attacks for POLBS schemes in terms of an interactive game (denoted by CrA-Pol-CMA), played between a challenger and an adversary, which we define below.

Definition 3.4 *The* CrA-Pol-CMA *game consists of three stages:* Setup, Challenge *and* Guess, *which we describe below.*

- **Setup**. *On input of a security parameter k, the challenger does the following:*
  1. *Run algorithm* Setup *to obtain the system public parameters* $\mathcal{P}$
  2. *Run algorithm* Issuer-Setup N *times to obtain a set of credential issuers* $\mathcal{I} = \{I_1, \ldots, I_N\}$
  3. *Give to the adversary the parameters* $\mathcal{P}$ *as well as the public and master keys of the different credential issuers included in* $\mathcal{I}$.
- **Challenge**. *The adversary chooses a message* M_{ch}*, a pair of keys* (pk_{ch}, sk_{ch}) *and a policy* $Pol_{ch}^{pk_{ch}}$ *on which he wishes to be challenged. The challenger does the following:*
  1. *For* $i = 1, \ldots, m$*, pick at random* $j_i^{ch} \in \{1, \ldots, m_i\}$,
  2. *Run algorithm* CredGen m *times to obtain the set of credentials* $\varsigma_{j_1^{ch},\ldots,j_m^{ch}}(Pol_{ch}^{pk_{ch}})$,
  3. *Run algorithm* Sign *on input the tuple* $(M_{ch}, pk_{ch}, sk_{ch}, Pol_{ch}, \varsigma_{j_1^{ch},\ldots,j_m^{ch}}(Pol_{ch}^{pk_{ch}}))$ *and return the resulting output to the adversary.*
- **Guess**. *The adversary outputs a tuple* $(j_1, \ldots, j_m)$*, and wins the game if* $(j_1^{ch}, \ldots, j_m^{ch}) = (j_1, \ldots, j_m)$.

In the following, we provide a formal definition for CrA-Pol-CMA secure POLBS schemes.

Definition 3.5 *The advantage of an adversary* $\mathcal{A}$ *in the* CrA-Pol-CMA *game is defined to be the quantity* $Adv_{\mathcal{A}} = Max_i\{|Pr[j_i = j_i^{ch}] - \frac{1}{m_i}|\}$*, where the parameters* m_i *are those defined by the challenge policy* $Pol_{ch}^{pk_{ch}}$*. A* POLBS *scheme is* CrA-Pol-CMA *secure if no probabilistic polynomial time adversary has a non-negligible advantage in the* CrA-Pol-CMA *game.*

Now that we have formally defined policy-based signature and the related security model, we propose in the following an elegant and relatively efficient policy-based signature scheme the security of which is proved in the random oracle model.

3.4 A Pairing-Based Implementation

3.4.1 Description

Our POLBS scheme consists of the algorithms described below.

- **Setup**. On input of a security parameter k, do the following:
  1. Run algorithm BDH-Setup on input k to generate output $(q, \mathbb{G}_1, \mathbb{G}_T, e, P)$
  2. Define three hash functions: $H_0 : \{0,1\}^* \to \mathbb{G}_1$, $H_1 : \{0,1\}^* \to \mathbb{Z}_q^*$ and $H_2 : \mathbb{G}_1 \to \mathbb{Z}_q^*$
  3. Let $\mathcal{P} = (q, \mathbb{G}_1, \mathbb{G}_T, e, P, n, H_0, H_1, H_2)$.
- **Issuer-Setup**. Let $\mathcal{I} = \{I_1, \ldots, I_N\}$ be a set of credential issuers. Each credential issuer $I_\kappa \in \mathcal{I}$ picks at random a secret master key $s_\kappa \in \mathbb{Z}_q^*$ and publishes the corresponding public key $R_\kappa = s_\kappa \cdot P$.
- **KeyGen**. This algorithm picks at random a private key $sk_u \in \mathbb{Z}_q^*$ and computes the corresponding public key $pk_u = sk_u \cdot P$.
- **CredGen**. On input of the public key R_κ of issuer $I_\kappa \in \mathcal{I}$ and assertion $A^{pk_u} \in \{0,1\}^*$, this algorithm outputs $\varsigma(R_\kappa, A^{pk_u}) = s_\kappa \cdot H_0(A^{pk_u})$.
- **Sign**. On input of a message M, a private key sk_u and a qualified set of credentials $\varsigma_{j_1,\ldots,j_m}(Pol^{pk_u})$ for policy Pol^{pk_u}, do the following:
  1. For $i = 1, \ldots, m$, do the following:
     (a) Pick at random $Y_i \in \mathbb{G}_1$, then compute $x_{i,j_i+1} = e(P, Y_i)$
     (b) For $l = j_i + 1, \ldots, m_i, 1, \ldots, j_i - 1 \bmod (m_i + 1)$, do the following:
        i. Compute $\tau_{i,l} = \prod_{k=1}^{m_{i,l}} e(R_{\kappa_{i,l,k}}, H_0(A_{i,l,k}^{pk_u}))$
        ii. Pick at random $Y_{i,l} \in \mathbb{G}_1$, then compute $x_{i,l+1} = e(P, Y_{i,l}) * \tau_{i,l}^{H_1(M \| x_{i,l} \| m \| i \| l)}$
     (c) Compute $Y_{i,j_i} = Y_i - H_1(M \| x_{i,j_i} \| m \| i \| j_i) \cdot (\sum_{k=1}^{m_{i,j_i}} \varsigma(R_{\kappa_{i,j_i,k}}, A_{i,j_i,k}^{pk_u}))$
  2. Compute $Y = \sum_{i=1}^{m} \sum_{j=1}^{m_i} Y_{i,j}$, then compute $Z = (sk_u + H_2(Y))^{-1} \cdot P$
  3. Return $\sigma = ([[x_{i,j}]_{j=1}^{m_i}]_{i=1}^{m}, Y, Z)$
- **Verify**. Let $\sigma = ([[x_{i,j}]_{j=1}^{m_i}]_{i=1}^{m}, Y, Z)$ be a signature on message M according to policy Pol^{pk_u} and public key pk_u. To check the validity of σ, do the following:
  1. Compute $\tau_{i,j} = \prod_{k=1}^{m_{i,j}} e(R_{\kappa_{i,j,k}}, H_0(A_{i,j,k}^{pk_u}))$ (for $j = 1, \ldots, m_i$ and $i = 1, \ldots, m$)
  2. Compute $\alpha_0 = e(pk_u + H_2(Y) \cdot P, Z)$
  3. Compute $\alpha_1 = \prod_{i=1}^{m} [\prod_{j=1}^{m_i} x_{i,j}]$ and $\alpha_2 = e(P, Y) * \prod_{i=1}^{m} \prod_{j=1}^{m_i} \tau_{i,j}^{H_1(M \| x_{i,j} \| m \| j \| i)}$
  4. If $\alpha_0 = e(P, P)$ and $\alpha_1 = \alpha_2$, then return $\top$, otherwise return $\bot$

Remark 3.2 *The intuition behind our* Sign *and* Verify *algorithms is as follows:*

1. *Each conjunction of conditions* $\wedge_{k=1}^{m_{i,j}} \langle I_{\kappa_{i,j,k}}, A_{i,j,k}^{pk_u} \rangle$ *is associated to a tag* $\tau_{i,j}$.
2. *For each indice i, the set of tags* $\{\tau_{i,j}\}_{j=1}^{m_i}$ *is equivalent to a set of ring members. The signature key of the ring member corresponding to the tag* $\tau_{i,j}$ *consists of the credentials* $\{\varsigma(R_{\kappa_{i,j,k}}, A_{i,j,k}^{pk_u})\}_{k=1}^{m_{i,j}}$.

3. *The generated signature corresponds to a set of ring signatures which validity can be checked using the global 'glue' value Y. The latter can be computed only by a user having access to a qualified set of credentials for policy Pol^{pk_u}.*

4. *The element Z represents the [151] short signature on the value Y using the private key sk_u. Therefore, σ proves that the entity whose public key is pk_u is compliant with Pol^{pk_u}.*

3.4.2 Consistency and Efficiency

Our POLBS scheme satisfies the consistency constraint thanks to the following statements:

$$\alpha_0 = e(pk_u + H_2(Y)\cdot P, Z) = e((sk_u + H_2(Y))\cdot P, (sk_u + H_2(Y))^{-1}\cdot P) = e(P,P) \tag{3.1}$$

$$\tau_{i,j}^{H_1(M\|x_{i,j}\|m\|i\|j)} = x_{i,j+1} * e(P, Y_{i,j})^{-1} \qquad (\text{where} x_{i,m_i+1} = x_{i,1}) \tag{3.2}$$

$$\begin{aligned}
\alpha_2 &= \lambda * \prod_{i=1}^{m}[\prod_{j=1}^{m_i} \tau_{i,j}^{H_1(M\|x_{i,j}\|m\|i\|j)}] \qquad (\text{where } \lambda = e(P,Y)) \\
&= \lambda * \prod_{i=1}^{m}[\prod_{j=1}^{m_i-1} x_{i,j+1} * e(P,Y_{i,j})^{-1} * x_{i,1} * e(P,Y_{i,m_i})^{-1}] \\
&= \lambda * \prod_{i=1}^{m}[\prod_{j=1}^{m_i} x_{i,j} * \prod_{j=1}^{m_i} e(P,Y_{i,j})^{-1}] \\
&= \lambda * [\prod_{i=1}^{m}\prod_{j=1}^{m_i} x_{i,j}] * [e(P, \sum_{i=1}^{m}\sum_{j=1}^{m_i} Y_{i,j})]^{-1} = \lambda * \alpha_1 * \lambda^{-1}
\end{aligned} \tag{3.3}$$

Table 3.1: Computational costs of our POLBS scheme

Sign	$(\sum_{i=1}^{m}[1+\sum_{j=1}^{m_i-1}(1+m_{i,j})]).\mathbf{pa} + (\sum_{i=1}^{m}(m_i-1+\max(m_{i,j}))).\mathbf{ad}_1$ $+(m+1).\mathbf{mu}_1 + (\sum_{i=1}^{m}\sum_{j=1}^{m_i-1} m_{i,j}).\mathbf{mu}_T + (\sum_{i=1}^{m}(m_i-1)).\mathbf{exp}_T$
Verify	$(3+\sum_{i=1}^{m}\sum_{j=1}^{m_i} m_{i,j}).\mathbf{pa} + (\sum_{i=1}^{m} m_i).\mathbf{exp}_T + \mathbf{mu}_1$ $+(2m-1+\sum_{i=1}^{m}(2(m_i-1)+\sum_{j=1}^{m_i} m_{i,j})).\mathbf{mu}_T$

In Table 3.1, we summarize the computational costs (in the worst case) of our POLBS scheme. We consider the computational costs of our algorithms in terms of the following notations: **pa**

the pairing, $\mathbf{ad}_1$ the addition in the group $\mathbb{G}_1$, $\mathbf{mu}_1$ the scalar multiplication in the group $\mathbb{G}_1$, $\mathbf{mu}_T$ the multiplication in the group $\mathbb{G}_T$, $\mathbf{exp}_T$ the exponentiation in the group $\mathbb{G}_T$. As for POLBE and $\mathsf{POLBE}_{\mathsf{PK}}$ schemes, we ignore here the costs of hash computations.

The essential operation in pairing-based cryptography is pairing computations:

- our Sign algorithm requires a total of $\sum_{i=1}^{m} m_i + \sum_{i=1}^{m} \sum_{j \neq j_i} m_{i,j}$ pairing computations. Note that the values $\tau_{i,l}$ does not depend on the signed message M. Thus, they can be pre-computed by the end user, cached and used in subsequent signatures involving the corresponding credential-based conditions i.e. $\langle R_{\kappa_{i,l,k}}, A_{i,l,k}^{pk_u} \rangle$.
- our Verify algorithm requires a total of $3 + \sum_{i=1}^{m} \sum_{j=1}^{m_i} m_{i,j}$ pairing computations. Two of them are used for the verification of the glue value Y, of which the pairing $e(P,P)$ can be pre-computed.

Let l_i denote the bit-length of the bilinear representation of an element of group $\mathbb{G}_i$ $(i = 1,2)$. The bit-length of a signature produced by our POLBS scheme is equal to $(\sum_{i=1}^{m} m_i).l_T + 2.l_1$. Observe that thanks to the adopted ring structure, the length of the signature does not depend on the values $m_{i,j}$ i.e. the number of conjunctions in a DNF-form policy. This property cannot be achieved using, for instance, the ring structure of the identity-based ring signature scheme proposed in [150].

3.4.3 Security

In the following, we study the security properties of our POLBS scheme. We first address the unforgeability our scheme, respectively, against Insiders' attacks (Theorem 3.1) and against Outsider' attacks (Theorem 3.2), then we discuss the credential ambiguity property (Theorem 3.3).

Remark 3.3 *As in the security analysis of our* POLBE *scheme (Section 1.4), we denote by $m_{\vee\wedge}, m_{\vee}$ and $m_{\wedge}$, respectively, the maximum values that the quantities m, m_i and $m_{i,j}$ can take respectively in the considered policies.*

Theorem 3.1 *Our* POLBS *scheme is* EUF-Pol-CMA^I *secure in the random oracle model under the assumption that CDHP is hard.*

Proof Theorem 3.1 is a consequence of Lemma 3.1. □

Lemma 3.1 *Let $\mathcal{A}^\circ$ be an* EUF-Pol-CMA^I *adversary with advantage $Adv_{\mathcal{A}^\circ} \geq \varepsilon$ when attacking our* POLBS *scheme. Assume that adversary $\mathcal{A}^\circ$ has running time $t_{\mathcal{A}^\circ}$ and makes at most q_c queries to oracle* CredGen-O*, q_s queries to oracle* Sign-O*, q_0 queries to oracle H_0 and q_1 queries to oracle H_1. Then, there exists an adversary $\mathcal{A}^\bullet$ the advantage of which, when attacking* CDHP*, is such that $Adv_{\mathcal{A}^\bullet} \geq 9/(100 q_0^{m_{\vee\wedge} m_{\vee}} \sum_{l=1}^{m_{\vee}} l! \binom{m_{\vee}}{l})$.*

Assuming that $q \geq Max\{2m_{\vee\wedge}m_{\vee}, 2m_{\vee\wedge}q_s q_1\}$ *and* $\varepsilon \leq 32(q_1+1-m_{\vee\wedge}m_{\vee})/q$*, the running time of adversary* $\mathcal{A}^{\bullet}$ *is such that* $t_{\mathcal{A}^{\bullet}} \leq (32q_1+4)t_{\mathcal{A}^{\circ}}/\varepsilon$.

Proof Our proof of Lemma 3.1 is inspired by the method described in [88], which in turn is based on the oracle replay technique [127]. Informally, by a polynomial replay of the attack with different random oracles, we allow the attacker to forge two signatures that are related so that the attacker is able to solve the underlying hard problem (CDHP in our case).

Let $\mathcal{A}^{\circ}$ be an EUF-Pol-CMA$^{\mathsf{I}}$ adversary with advantage $\mathrm{Adv}_{\mathcal{A}^{\circ}} \geq \varepsilon$ when attacking our POLBS scheme. Assume that adversary $\mathcal{A}^{\circ}$ has running time $t_{\mathcal{A}^{\circ}}$ and makes at most q_c queries to oracle CredGen-O, q_s queries to oracle Sign-O, q_0 queries to oracle H_0 and q_1 queries to oracle H_1. In the following, we construct an algorithm $\mathcal{A}^{\bullet}$ that uses $\mathcal{A}^{\circ}$ to mount an attack against CDHP.

The EUF-Pol-CMA$^{\mathsf{I}}$ game between the challenger and algorithm $\mathcal{A}^{\bullet}$ starts with the Setup$^{\bullet}$ stage which we describe below.

- **Setup**$^{\bullet}$. The challenger gives to adversary $\mathcal{A}^{\bullet}$ the BDH parameters $(q, \mathbb{G}_1, \mathbb{G}_T, e, P)$ as well as a CDHP-instance $(P, a\cdot P, b\cdot P) = (P, P_1, P_2)$ for these parameters.

  Before interacting with adversary $\mathcal{A}^{\circ}$, algorithm $\mathcal{A}^{\bullet}$ does the following:

  1. Choose the values $i^{\bullet} \in \{1,\ldots,m_{\vee\wedge}\}$, $j^{\bullet} \in \{1,\ldots,m_{\vee}\}$ and $m^{\bullet}_{i^{\bullet},j^{\bullet}} \in \{1,\ldots,m_{\wedge}\}$
  2. Pick at random the values $\kappa^{\bullet}_{i^{\bullet},j^{\bullet},k} \in \{1,\ldots,N\}$ and $l_{i^{\bullet},j^{\bullet},k} \in \{1,\ldots,q_0\}$, for $k = 1,\ldots,m^{\bullet}_{i^{\bullet},j^{\bullet}}$
  3. Pick at random $\theta^{\bullet}_{i^{\bullet},j^{\bullet},k} \in \mathbb{Z}^*_q$, for $k = 2,\ldots,m^{\bullet}_{i^{\bullet},j^{\bullet}}$, then compute $\theta^{\bullet}_{i^{\bullet},j^{\bullet},1} = \sum_{k=2}^{m^{\bullet}_{i^{\bullet},j^{\bullet}}} \theta^{\bullet}_{i^{\bullet},j^{\bullet},k}$

The interaction between algorithm $\mathcal{A}^{\bullet}$ and adversary $\mathcal{A}^{\circ}$ consists of three stages: Setup$^{\circ}$, Queries$^{\circ}$ and Forge$^{\circ}$, which we describe below.

- **Setup**$^{\circ}$. Algorithm $\mathcal{A}^{\bullet}$ does the following:

  1. Let $\mathcal{P}^{\bullet} = (q, \mathbb{G}_1, \mathbb{G}_T, e, P, n, H^{\bullet}_0, H^{\bullet}_1, H_2)$ be the system public parameters, where the oracles $H^{\bullet}_0$ and $H^{\bullet}_1$ are controlled by algorithm $\mathcal{A}^{\bullet}$, $H_2 : \mathbb{G}_1 \rightarrow \mathbb{Z}^*_q$ is a public hash function, the tuple $(q, \mathbb{G}_1, \mathbb{G}_T, e, P)$ is given to algorithm $\mathcal{A}^{\bullet}$ in the Setup$^{\bullet}$ stage, and the value $n \in \mathbb{N}^*$ is chosen by algorithm $\mathcal{A}^{\bullet}$. Algorithm $\mathcal{A}^{\bullet}$ controls $H^{\bullet}_0$ and $H^{\bullet}_1$ as follows:
     - For the random oracle $H^{\bullet}_0$, algorithm $\mathcal{A}^{\bullet}$ maintains a list of tuples $[A_\iota, H_{0,\iota}, \lambda_\iota]$ which we denote H^{list}_0. The list is initially empty. Assume that adversary $\mathcal{A}^{\circ}$ makes a query on assertion $A^{pk_u} \in \{0,1\}^*$, then adversary $\mathcal{A}^{\bullet}$ responds as follows:
       (a) If A^{pk_u} appears on the list H^{list}_0 in a tuple $[A_\iota, H_{0,\iota}, \lambda_\iota]$, then return $H_{0,\iota}$

(b) If $pk_u = pk_{\mathrm{f}}$ and A^{pk_u} does not appear on H_0^{list} and A^{pk_u} is the $l^{\bullet}_{i^{\bullet},j^{\bullet},1}$-th distinct query to oracle $H_0^{\bullet}$, then compute $H_{0,l^{\bullet}_{i,j,1}} = r^{-1}_{\kappa^{\bullet}_{i^{\bullet},j^{\bullet},1}} \cdot (P_2 - \theta^{\bullet}_{i^{\bullet},j^{\bullet},1} \cdot P)$, return $H_{0,l^{\bullet}_{i^{\bullet},j^{\bullet},1}}$, and add the entry $[A^{pk_u}, H_{0,l^{\bullet}_{i^{\bullet},j^{\bullet},1}}, \mathsf{null}]$ to H_0^{list}

(c) If $pk_u = pk_{\mathrm{f}}$ and A^{pk_u} does not appear on H_0^{list} and A^{pk_u} is the $l^{\bullet}_{i^{\bullet},j^{\bullet},k}$-th distinct query to oracle $H_0^{\bullet}$ (for $k > 1$), then compute $H_{0,l^{\bullet}_{i^{\bullet},j^{\bullet},k}} = (r^{-1}_{\kappa^{\bullet}_{i^{\bullet},j^{\bullet},k}} \theta^{\bullet}_{i^{\bullet},j^{\bullet},k}) \cdot P$, return $H_{0,l^{\bullet}_{i^{\bullet},j^{\bullet},k}}$, and add $[A^{pk_u}, H_{0,l^{\bullet}_{i^{\bullet},j^{\bullet},k}}, r^{-1}_{\kappa^{\bullet}_{i^{\bullet},j^{\bullet},k}} \theta^{\bullet}_{i^{\bullet},j^{\bullet},k}]$ to H_0^{list}

(d) Otherwise, pick at random $\lambda \in \mathbb{Z}_q^*$, return $\lambda \cdot P$ and add $[A^{pk_u}, \lambda \cdot P, \lambda]$ to H_0^{list}

Note that $H_0^{\bullet}$ is such that $\tau^{\bullet}_{i^{\bullet},j^{\bullet}} = \prod_{k=1}^{m^{\bullet}_{i^{\bullet},j^{\bullet}}} e(R_{\kappa_{i^{\bullet},j^{\bullet},k}}, H_0(A^{pk_{\mathrm{f}}}_{i^{\bullet},j^{\bullet},k})) = e(P, ab \cdot P)$.

– For the random oracle $H_1^{\bullet}$, $\mathcal{A}^{\bullet}$ maintains a list of tuples $[(M_\iota, x_\iota, m_\iota, i_\iota, j_\iota), H_{4,\iota}]$ which we denote H_1^{list}. The list is initially empty. Assume that adversary $\mathcal{A}^{\circ}$ makes a query to the random oracle $H_1^{\bullet}$ on input (M, x, m, i, j), then algorithm $\mathcal{A}^{\bullet}$ responds as follows:

(a) If (M, x, m, i, j) already appears on H_1^{list} in a tuple $[(M_\iota, x_\iota, m_\iota, i_\iota, j_\iota), H_{1,\iota}]$, then output $H_{1,\iota}$

(b) Otherwise, pick at random $H \in \mathbb{Z}_q^*$, output H and add $[(M, x, m, i, j), H]$ to H_1^{list}

2. Define the set of credential issuers $\mathcal{I} = \{I_1, \ldots, I_N\}$ as follows:

– For $\kappa \in \{\kappa^{\bullet}_{i^{\bullet},j^{\bullet},k}\}$, the public key of I_κ is $R_\kappa = r_\kappa \cdot P_1$ for some randomly chosen $r_\kappa \in \mathbb{Z}_q^*$. In this case, the master key of I_κ is $s_\kappa = r_\kappa a \in \mathbb{Z}_q^*$

– For $\kappa \in \{1, \ldots, N\} \setminus \{\kappa^{\bullet}_{i^{\bullet},j^{\bullet},k}\}$, the public key of I_κ is $R_\kappa = s_\kappa \cdot P$ for some randomly chosen master key $s_\kappa \in \mathbb{Z}_q^*$

3. Run algorithm KeyGen to obtain a public/private key pair $(pk_{\mathrm{f}}, sk_{\mathrm{f}})$

4. Give the parameters $\mathcal{P}^{\bullet}$ and the credential issuers' public keys $\{R_\kappa\}_{\kappa=1}^{N}$ to $\mathcal{A}^{\circ}$

- **Queries**$^{\circ}$. Adversary $\mathcal{A}^{\circ}$ performs a polynomial number of oracle queries adaptively.

- **Forge**$^{\circ}$. Algorithm $\mathcal{A}^{\circ}$ outputs a message M_{f}, a policy $Pol_{\mathrm{f}}^{pk_{\mathrm{f}}}$ and a signature σ_{f}. The adversary wins the game if $\mathsf{Verify}_{Pol_{\mathrm{f}}^{pk_{\mathrm{f}}}, pk_{\mathrm{f}}}(M_{\mathrm{f}}, \sigma_{\mathrm{f}}) = \top$.

During the Queries$^{\circ}$ stage, adversary $\mathcal{A}^{\circ}$ can make the queries of its choice to two oracles, denoted by CredGen-O$^{\circ}$ and Sign-O$^{\circ}$, controlled by adversary $\mathcal{A}^{\bullet}$ as described below.

- **CredGen-O**$^{\circ}$. Assume that adversary $\mathcal{A}^{\circ}$ makes a query on a tuple (I_κ, A). Let $[A_\iota, H_{0,\iota}, \lambda_\iota]$ be the tuple from H_0^{list} such that $A_\iota = A^{pk_u}$, then algorithm $\mathcal{A}^{\bullet}$ responds as follows:

1. If $\iota = l^{\bullet}_{i^{\bullet},j^{\bullet},1}$ and $\kappa \in \{\kappa^{\bullet}_{i^{\bullet},j^{\bullet},k}\}$, then report failure and terminate (event $\mathcal{E}_{\mathsf{cred}}$)

2. If $\iota \neq l^{\bullet}_{i^{\bullet},j^{\bullet},1}$ and $\kappa \in \{\kappa^{\bullet}_{i^{\bullet},j^{\bullet},k}\}$, then return $(r_\kappa \lambda_\iota) \cdot P_1 = (r_\kappa a) \cdot H_{0,\iota} = s_\kappa \cdot H_{0,\iota}$

3. If $\kappa \in \{1, \ldots, N\} \setminus \{\kappa^{\bullet}_{i^{\bullet},j^{\bullet},k}\}$, then return $s_\kappa \cdot H_{0,\iota}$

- **Sign-O$^\circ$**. Assume that adversary $\mathcal{A}^\circ$ makes a query on a tuple (M, Pol^{pk_f}). Algorithm $\mathcal{A}^\bullet$ responds as follows:

  1. Pick at random $h_{i,1} \in \mathbb{Z}_q^*$, and $Y_{i,j} \in \mathbb{G}_1$ ($j = 1,\ldots,m_i$ and $i = 1,\ldots,m$)
  2. Compute $\tau_{i,j} = \prod_{k=1}^{m_{i,j}} e(R_{\kappa_{i,j,k}}, H_0^\bullet(A_{i,j,k}^{pk_f}))$ ($j = 1,\ldots,m_i$ and $i = 1,\ldots,m$)
  3. For $j = 1,\ldots,m_i - 1$ and $i = 1,\ldots,m$, compute the value $x_{i,j+1} = e(P, Y_{i,j}) * \tau_{i,j}^{h_{i,j}}$, then compute $h_{i,j+1} = H_1(M\|x_{i,j+1}\|m\|i\|j+1)$. In order to compute the value $h_{i,j}$, algorithm $\mathcal{A}^\bullet$ refers to the list H_1^{list}. If $(M, x_{i,j}, m, i, j)$ appears on H_1^{list} in a tuple $[(M_l, x_l, m_l, i_l, j_l), H_{1,l}]$, then algorithm $\mathcal{A}^\bullet$ sets $h_{i,j} = H_{1,l}$. Otherwise, it picks at random $H \in \mathbb{Z}_q^*$, sets $h_{i,j} = H$ and adds the tuple $[(M, x_{i,j}, m, i, j), H]$ to H_1^{list}.
  4. Let $x_{i,1} = e(P, Y_{i,m_i}) * \tau_{i,m_i}^{h_{i,m_i}}$ and $h_{i,1} = H_1(M\|x_{i,1}\|m\|i\|1)$, then
     (a) If $(M, x_{i,1}, m, i, 1)$ already appears on the list H_1^{list} in a tuple $[(M_l, x_l, m_l, i_l, 1), H_{1,l}]$ such that $H_{1,l} \neq h_{i,1}$, then report failure and terminate (event referred to as $\mathcal{E}_{\mathsf{sig}}$).
     (b) Otherwise, add the tuple $[(M, x_{i,1}, m, i, 1), h_{i,1}]$ to H_1^{list}.
  5. Compute $Y = \sum_{i=1}^{m} \sum_{j=1}^{m_i} Y_{i,j}$ and $Z = (sk_u + H_2(Y))^{-1} \cdot P$, then return the signature $\sigma = ([x_{i,j}]_{j=1}^{m_i}]_{i=1}^{m}, Y, Z)$ to adversary $\mathcal{A}^\circ$.

Remark 3.4 *Without loss of generality, we assume that adversary $\mathcal{A}^\circ$ always makes the appropriate query to the random oracle $H_0^\bullet$ on assertion A^{pk_u} before making any query involving A^{pk_u} to oracles* CredGen-O$^\circ$ *and* Sign-O$^\circ$.

In the following, we analyze the simulation described above.

Let w be the whole set of random tapes that take part in an attack by adversary $\mathcal{A}^\circ$, with the environment simulated by algorithm $\mathcal{A}^\bullet$, but excluding the randomness related to the oracle $H_1^\bullet$. The success probability of adversary $\mathcal{A}^\circ$ in forging a valid ring signature scheme is then taken over the space $(w, H_1^\bullet)$. Let $\mathcal{S}$ be the set of successful executions of adversary $\mathcal{A}^\circ$, then the following holds

$$\mathrm{Adv}_{\mathcal{A}^\circ} = Pr[(w, H_1^\bullet) \in \mathcal{S}] \geq \varepsilon \tag{3.4}$$

We define the events $\mathcal{E}_0$ and $\mathcal{E}_0'$ as follows:

- $\mathcal{E}_0$ is the event that adversary $\mathcal{A}^\circ$ succeeds in forging the signature $\sigma_f = ([x_{i,j}^f]_{j=1}^{m_i}]_{i=1}^{m}, Y^f, Z^f)$ without making a query to the random oracle $H_1^\bullet$ on at least one of the tuples $(M_f, x_{i,j}^f, m, i, j)$
- $\mathcal{E}_0'$ is defined to be the event that event $\mathcal{E}_{\mathsf{sig}}$ occurs at one of the queries made by adversary $\mathcal{A}^\circ$ to the oracle Sign-O$^\circ$.

Then, the following statements hold:

$$Pr[\mathcal{E}_0] \leq \frac{m_{\vee\wedge} m_{\vee}}{q} \quad , \quad Pr[\mathcal{E}_0'] \leq \frac{m_{\vee\wedge} q_s q_1}{q} \tag{3.5}$$

Let $\mathcal{S}'$ be the set of successful executions of adversary $\mathcal{A}^{\circ}$ for which it has made queries to the random oracle $H_1^{\bullet}$ on the all the tuples $(M_{\mathrm{f}}, x_{i,j}^{\mathrm{f}}, m, i, j)$, then the following holds

$$\begin{aligned} Pr[(w, H_1^{\bullet}) \in \mathcal{S}'] &= Pr[\neg\mathcal{E}_0].Pr[\neg\mathcal{E}_0'].Pr[(w, H_1^{\bullet}) \in \mathcal{S}] \\ &\geq (1 - \frac{m_{\vee\wedge} m_{\vee}}{q}).(1 - \frac{m_{\vee\wedge} q_s q_1}{q}).\varepsilon \end{aligned} \tag{3.6}$$

Let $Q_1, \ldots, Q_{q_1}$ denote the different queries made by adversary $\mathcal{A}^{\circ}$ to the random oracle $H_1^{\bullet}$. We denote by $Q_{\beta_{i,j}}$ (for $\beta_{i,j} \in \{1, \ldots, q_1\}$) the query made by adversary $\mathcal{A}^{\circ}$ to the random oracle $H_1^{\bullet}$ on the tuple $(M_{\mathrm{f}}, x_{i,j}^{\mathrm{f}}, m, i, j)$. Let i^{lq} and j^{lq} be the indexes such that for all $(i, j) \neq (i^{lq}, j^{lq})$, $\beta_{i,j} < \beta_{i^{lq}, j^{lq}}$. The value $\beta_{i^{lq}, j^{lq}}$ is called the *last-query index*.

We define $\mathcal{S}'_{\beta_{i^{lq}, j^{lq}}}$ to be the set of executions from $\mathcal{S}'$ whose last-query index is $\beta_{i^{lq}, j^{lq}}$. Since $\beta_{i^{lq}, j^{lq}}$ may range between $\beta \leq \beta = m_{\vee\wedge} m_{\vee}$ and q_1, this gives us a partition of $\mathcal{S}'$ in at least $q_1 + 1 - \beta$ classes.

We define $\mathcal{E}_1$ to be the event that algorithm $\mathcal{A}^{\bullet}$ obtains a successful execution $(w^1, H_1^{\bullet 1}) \in \mathcal{S}'_{\beta^1_{i^{lq}, j^{lq}}}$, for some last-query index $\beta^1_{i^{lq}, j^{lq}}$, after invoking t_1 times adversary $\mathcal{A}^{\circ}$ with randomly chosen tuples $(w, H_1^{\bullet})$.

In the particular case where $t_1 = (Pr[(w, H_1^{\bullet}) \in \mathcal{S}'])^{-1}$, and since $(1 - \frac{1}{X})^X \leq e^{-1}$ (for $X > 1$), the following statement holds

$$Pr[\mathcal{E}_1] = 1 - (1 - Pr[(w, H_1^{\bullet}) \in \mathcal{S}'])^{t_1} \geq 1 - e^{-1} > \frac{3}{5} \tag{3.7}$$

We define the set $\mathcal{J}$ of last-query indexes which are more likely to appear as follows:

$$\mathcal{J} = \{\beta_{i^{lq}, j^{lq}} \; s.t. \; Pr[(w, H_1^{\bullet}) \in \mathcal{S}'_{\beta_{i^{lq}, j^{lq}}} | (w, H_1^{\bullet}) \in \mathcal{S}'] \geq \gamma\}, \text{ where } \gamma = \tfrac{1}{2(q_1 + 1 - \beta)}$$

Let $\mathcal{S}'_{\mathcal{J}} = \{(w, H_1^{\bullet}) \in \mathcal{S}'_{\beta_{i^{lq},j^{lq}}} \; s.t. \; \beta_{i^{lq},j^{lq}} \in \mathcal{J}\}$ be the subset of successful executions corresponding to the set $\mathcal{J}$. Since the subsets $\mathcal{S}'_{\beta_{i^{lq},j^{lq}}}$ are pairwise disjoint, the following holds

$$\begin{aligned} Pr[(w, H_1^{\bullet}) \in \mathcal{S}'_{\mathcal{J}} | (w, H_1^{\bullet}) \in \mathcal{S}'] &= \sum_{\beta_{i^{lq},j^{lq}} \in \mathcal{J}} Pr[(w, H_1^{\bullet}) \in \mathcal{S}'_{\beta_{i^{lq},j^{lq}}} | (w, H_1^{\bullet}) \in \mathcal{S}'] \\ &= 1 - \sum_{\beta_{i^{lq},j^{lq}} \notin \mathcal{J}} Pr[(w, H_1^{\bullet}) \in \mathcal{S}'_{\beta_{i^{lq},j^{lq}}} | (w, H_1^{\bullet}) \in \mathcal{S}'] \\ &\geq 1 - (\frac{1}{2\gamma} - |\mathcal{J}|).\gamma \geq \frac{1}{2} \end{aligned} \tag{3.8}$$

Let $\alpha = (1 - \frac{m_{\vee\wedge} m_{\vee}}{q}).(1 - \frac{m_{\vee\wedge} q_s q_1}{q}).\varepsilon.\gamma$, then equation (3.6) leads to the following statement

$$\begin{aligned} Pr[(w, H_1^{\bullet}) \in \mathcal{S}'_{\beta_{i^{lq},j^{lq}}}] &= Pr[(w, H_1^{\bullet}) \in \mathcal{S}'].Pr[(w, H_1^{\bullet}) \in \mathcal{S}'_{\beta_{i^{lq},j^{lq}}} | (w, H_1^{\bullet}) \in \mathcal{S}'] \\ &\geq \alpha \end{aligned} \tag{3.9}$$

Given that the oracle $H_1^{\bullet}$ can be written as a pair $(\tilde{H}_1, h_{i^{lq},j^{lq}})$, where $\tilde{H}_1$ corresponds to the answers for all the queries to oracle $H_1^{\bullet}$ except the query $Q_{\beta_{i^{lq},j^{lq}}}$ whose answer is denoted as $h_{i^{lq},j^{lq}}$, we define the set $\Omega_{\beta_{i^{lq},j^{lq}}}$ as follows:

$$\Omega_{\beta_{i^{lq},j^{lq}}} = \{(w, (\tilde{H}_1, h_{i^{lq},j^{lq}})) \in \mathcal{S}' \; s.t. \; Pr_{h_{i^{lq},j^{lq}}}[(w, (\tilde{H}_1, h_{i^{lq},j^{lq}})) \in \mathcal{S}'_{\beta_{i^{lq},j^{lq}}}] \geq \delta - \alpha\}$$

According to the splitting lemma (Lemma 0.1), for $\delta = 2\alpha$ the following statements hold

$$\forall \, (w, (\tilde{H}_1, h_{i^{lq},j^{lq}})) \in \Omega_{\beta_{i^{lq},j^{lq}}}, \; Pr_{h_{i^{lq},j^{lq}}}[(w, (\tilde{H}_1, h_{i^{lq},j^{lq}})) \in \mathcal{S}'_{\beta_{i^{lq},j^{lq}}}] \geq \alpha \tag{3.10}$$

$$Pr[(w, (\tilde{H}_1, h_{i^{lq},j^{lq}})) \in \Omega_{\beta_{i^{lq},j^{lq}}} | (w, (\tilde{H}_1, h_{i^{lq},j^{lq}})) \in \mathcal{S}'_{\beta_{i^{lq},j^{lq}}}] \geq \frac{\alpha}{\delta} = \frac{1}{2} \tag{3.11}$$

Now, assume that event $\mathcal{E}_1$ occurs and that the successful execution $(w^1, (\tilde{H}_1^1, h^1_{i^{lq},j^{lq}}))$ is in $\mathcal{S}'_{\mathcal{J}}$. Let $\mathcal{E}_2$ be the event that algorithm $\mathcal{A}^{\bullet}$ obtains, for some last-query index $\beta^1_{i^{lq},j^{lq}}$, a successful execution $(w^1, (\tilde{H}_1^1, h^2_{i^{lq},j^{lq}})) \in \Omega_{\beta^1_{i^{lq},j^{lq}}}$ such that $h^2_{i^{lq},j^{lq}} \neq h^1_{i^{lq},j^{lq}}$, after invoking t_2 times adversary $\mathcal{A}^{\circ}$, with fixed $(w^1, \tilde{H}_1^1)$ and randomly chosen $h_{i^{lq},j^{lq}}$. In the particular case where $t_2 = (\alpha - \frac{1}{q})^{-1}$, the following holds

$$Pr[\mathcal{E}_2] = 1 - (1 - (\alpha - \frac{1}{q}))^{t_2} \geq 1 - e^{-1} > \frac{3}{5} \tag{3.12}$$

Consider $(w^1, (\tilde{H}_1^1, h^1_{i^{lq},j^{lq}}))$ and $(w^1, (\tilde{H}_1^1, h^2_{i^{lq},j^{lq}}))$, the two successful executions of the attack obtained by algorithm $\mathcal{A}^{\bullet}$ if events $\mathcal{E}_1$ and $\mathcal{E}_2$ occur. For the two considered executions, the

random tapes w are identical, whereas the answers of the random oracle $H_1^{\bullet}$ to the queries of adversary $\mathcal{A}^{\circ}$ are identical only until the query $Q_{\beta^1_{i^{lq},j^{lq}}}$.

Let $\sigma_f^1 = ([x_{i,j}^1]_{j=1}^{m_i^1}]_{i=1}^{m^1}, Y^1, Z^1)$ and $\sigma_f^2 = ([x_{i,j}^2]_{j=1}^{m_i^2}]_{i=1}^{m^2}, Y^2, Z^2)$ be the signatures forged by adversary $\mathcal{A}^{\circ}$ through the two considered successful executions respectively. With probability greater than $\frac{1}{\sum_{l=1}^{m_{\vee}} l!\binom{m_{\vee}}{l}}$, we have $m^1 = m^2 = m$ and $m_i^1 = m_i^2 = m_i$ $(i = 1, \ldots, m)$. In this case, the following statements hold

1. $x_{i,j}^1 = x_{i,j}^2$ $(j = 1, \ldots, m_i$ and $i = 1, \ldots, m)$
2. $h_{i,j}^1 = h_{i,j}^2$ (for $j \neq j^{lq}$ and $i \neq i^{lq}$) and $h_{i^{lq},j^{lq}}^1 \neq h_{i^{lq},j^{lq}}^2$

The fact that σ_f^1 and σ_f^2 are valid policy-based signatures leads to the equality $e(P, Y^2 - Y^1) = \tau_{i^{lq},j^{lq}}^{h_{i^{lq},j^{lq}}^1 - h_{i^{lq},j^{lq}}^2}$. With probability greater than $1/q_0^{m_{\vee\wedge}m_{\vee}}$, we have $\tau_{i^{\bullet},j^{\bullet}}^{\bullet} = \tau_{i^{lq},j^{lq}}$. In this case, note that adversary $\mathcal{A}^{\circ}$ does not make a query to oracle $\mathsf{CredGen}^{\circ}$ on assertion $A_{l_{i^{\bullet},j^{\bullet},1}}^{\bullet}$ (event $\mathcal{E}_{\mathsf{cred}}$ does not occur). This case leads to the equality $\tau_{i^{lq},j^{lq}} = e(P, ab \cdot P)$, and so $Y^2 - Y^1 = (h_{i^{lq},j^{lq}}^1 - h_{i^{lq},j^{lq}}^2) \cdot (ab \cdot P)$.

To summarize, with probability $Pr[\mathcal{A}^{\bullet}$ wins$]$, algorithm $\mathcal{A}^{\bullet}$ succeeds to obtain $ab \cdot P$ by computing the quantity $(h_{i^{lq},j^{lq}}^1 - h_{i^{lq},j^{lq}}^2)^{-1} \cdot (Y^2 - Y^1)$. From statements (3.7), (3.8), (3.11) and (3.12), we have

$$\mathrm{Adv}_{\mathcal{A}^{\bullet}} = Pr[\mathcal{A}^{\bullet} \text{ wins}] \geq \frac{9}{100 q_0^{m_{\vee\wedge}m_{\vee}}} \cdot \frac{1}{\sum_{l=1}^{m_{\vee}} l!\binom{m_{\vee}}{l}}.$$

For $q \geq Max\{2m_{\vee\wedge}m_{\vee}, 2m_{\vee\wedge}q_s q_1\}$ and $\varepsilon \leq 32(q_1 + 1 - m_{\vee\wedge}m_{\vee})/q$, the running time of adversary $\mathcal{A}^{\bullet}$ is such that the following holds

$$t_{\mathcal{A}^{\bullet}} = (t_1 + t_2).t_{\mathcal{A}^{\circ}} = (\frac{\gamma}{\alpha} + \frac{1}{\alpha - 1/q}).t_{\mathcal{A}^{\circ}} \leq (\frac{4}{\varepsilon} + \frac{32(q_1 + 1 - m_{\vee\wedge}m_{\vee})}{\varepsilon}).t_{\mathcal{A}^{\circ}} \leq \frac{32q_1 + 4}{\varepsilon}.t_{\mathcal{A}^{\circ}}$$

Remark 3.5 *Our security reduction does not depend on the parameter $m_{\wedge}$. On the other hand, it depends exponentially on the parameters $m_{\vee\wedge}$ and $m_{\vee}$ which needs further improvement. Finally, note that the ID-based ring signature presented in [151] is not supported by any security arguments. Our proof could be easily adapted to realize the missing proofs. In fact, the ID-based ring signature of [151] is almost similar to our signature algorithm when used in the particular case where policies are such that $m_{\vee\wedge} = m_{\wedge} = 1$.*

□

Theorem 3.2 *Our* POLBS *scheme is* EUF-Pol-CMA^{O} *secure in the random oracle model under the assumption that* $(k+1)$*-EP is hard.*

Proof The security of our scheme POLBS in the EUF-Pol-CMA^{O} game is equivalent to the security of the short signature scheme presented in [151]. In fact, the outsider adversary succeeds in forging a POLBS scheme if and only if it succeeds in generating a valid Z corresponding to a valid $([[x_{i,j}]_{j=1}^{m_i}]_{i=1}^{m}, Y)$ associated to the pair of keys $(pk_{\mathrm{f}}, sk_{\mathrm{f}})$. As the adversary has access to the master keys of the different credential issuers, its is able to generate a valid tuple $([[x_{i,j}]_{j=1}^{m_i}]_{i=1}^{m}, Y)$ corresponding to any policy associated to pk_{f}. Therefore, the adversary needs to be able to generate a [151] short signature on Y using the protected private key sk_{f}. The short signature of [151] is proved to be secure in the random oracle model under the assumption that the $(k+1)$-EP problem is hard. □

Theorem 3.3 *Our* POLBS *scheme is* CrA-Pol-CMA *secure in the random oracle model.*

Proof Let M_{ch} be the message and $\sigma_{\mathrm{ch}} = ([x_{i,j}^{\mathrm{ch}}]_{j=1}^{m_i}]_{i=1}^{m}, Y^{\mathrm{ch}}, Z^{\mathrm{ch}})$ be the signature which the adversary is challenged on in the CrA-Pol-CMA game. Our POLBS scheme is such that the following holds

1. $x_{i,j}^{\mathrm{ch}} = e(P, Y_{i,j-1}) * \tau_{i,j-1}^{H_1(M_{\mathrm{ch}} \| x_{i,j-1}^{\mathrm{ch}} \| m \| i \| j-1)}$ for $j \neq j_i^{\mathrm{ch}} + 1$ and $x_{i,j_i^{\mathrm{ch}}+1}^{\mathrm{ch}} = e(P, Y_i)$
2. $Y^{\mathrm{ch}} = \sum_{i=1}^{m} [\sum_{j \neq j^{\mathrm{ch}}} Y_{i,j} + Y_i - H_1(M_{\mathrm{ch}} \| x_{i,j_i^{\mathrm{ch}}}^{\mathrm{ch}} \| m \| i \| j_i^{\mathrm{ch}}) \cdot (\sum_{k=1}^{m_{i,j_i^{\mathrm{ch}}}} \varsigma(R_{\kappa_{i,j_i^{\mathrm{ch}},k}}, A_{i,j_i^{\mathrm{ch}},k}))]$

Since Y_i and $Y_{i,j-1}$ are chosen at random from $\mathbb{G}_1$, and H_1 is assumed to be a random oracle, we have that $x_{i,j}^{\mathrm{ch}}$ and Y^{ch} are uniformly distributed in $\mathbb{G}_T$ and $\mathbb{G}_1$ respectively. If $(j_1, \ldots, j_m)$ is the tuple output by the adversary in the CrA-Pol-CMA game, then we have $Pr[j_i = j_i^{\mathrm{ch}}]$, for $i = 1, \ldots, m$.

□

Now that we have formalized the concept of policy-based signature and proposed a provably secure policy-based signature scheme using bilinear pairings over elliptic curves, we present in the following section an application of this primitive in the context of proxy certification.

3.5 Proof-Carrying Proxy Certificates

The concept of proxy certificates, first formalized in [122], allows an end user to delegate some responsibility to another user, called agent, so that the latter can perform certain actions on behalf of the former. A proxy certificate is a certificate that, in contrast with the public-key certificates issued by trusted certification authorities, is generated by an end user. It represents

the signature of the end user on a message that typically contains the identity of the end user himself, the public key of the agent and a set of statements defining the terms of the delegation. It allows the agent to authenticate with other users as if it was the end user when performing the delegated actions. Proxy certification has been suggested for use in a number of applications particularly in distributed computing environments where delegation of rights is quite common. Examples include grid computing [23], mobile agents for e-commerce [57], and mobile communication [54]. More recently, an X.509 certificate profile for proxy certificates was proposed in [140]. In this section, we introduce an advanced form of proxy certificates which are generated using the policy-based signature primitive.

Whenever an agent wants to perform an action on behalf of an end user, it must prove that it is authorized by the end user to perform the action on its behalf. This is achieved by providing a valid proxy certificate and by proving the possession of the private key corresponding to the agent's public key specified by the certificate. Furthermore, the agent has to prove that the end user is compliant with the authorization policy associated to the action it wants to perform. As already mentioned in this thesis, an increasingly popular approach consists in using authorization policies fulfilled by digital credentials. The traditional approach consists in that the end user gives the agent a qualified set of credentials for the policy. The agent provides this set of credentials together with its proxy certificates. The entity that is in charge of making the authorization decision is called the verifier. The latter not only has to check the validity of each of the received credentials, but also it has to check that the received set of credentials fulfills the authorization policy associated to the requested action.

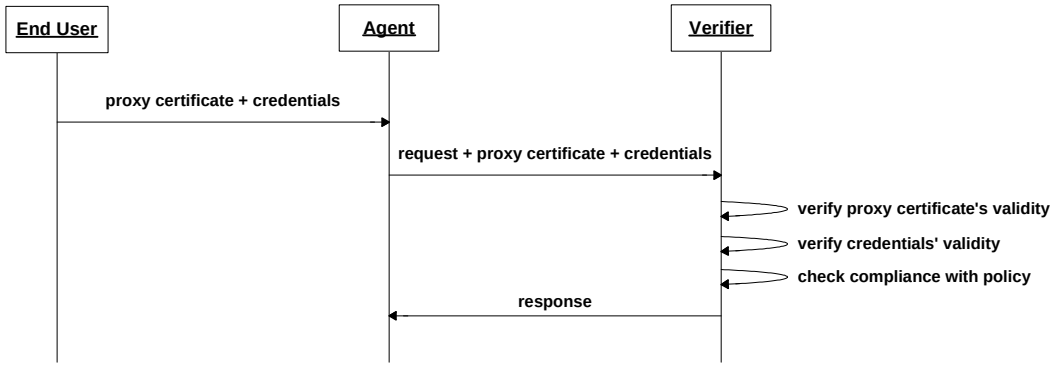


Figure 3.2: Proxy certificates: standard approach

The standard approach, depicted in Figure 3.2, suffers from three shortcomings:

- verifying the validity of the proxy certificate and the validity of the different credentials separately is a burden for the verifier.
- we believe that managing the end user's credentials and proving its compliance with an authorization policy should not be the role of the agent.

- proving the compliance of a user with a credential-based policy through the disclosure of a qualified set of credentials is not optimal from a privacy point of view. More precisely, it is not compliant with the privacy principle of data minimization according to which only strictly necessary information should be collected for a given purpose. For instance, assume that the authorization policy requires the possession of at least one credential belonging to a set of multiple credentials. Then, according to the data minimization principle, the verifier should not know more than the fact that the end user is compliant with the policy. In other words, the verifier should not know which specific credential fulfilling the authorization policy is held by the end user.

In this section, we present an advanced form of proxy certificates called *proof-carrying proxy certificates*. In contrast with the standard proxy certificates that are generated using standard (public-key) signature schemes, the proposed certificates are generated using a policy-based signature scheme i.e. a signature scheme for which the validity of a generated signature proves the compliance of the signer with a credential-based policy. Using this special form of proxy certificates, the end user does not disclose any of its credentials. It uses them to generate a proof of compliance with the verifier's authorization policy. Besides, the agent does not have to deal with the end user's credentials. It just provides its proof-carrying proxy certificate (in addition to proving the possession of the private key corresponding to the agent's public key specified by the certificate). Finally, the verifier just needs to verify the validity of the received proxy certificate with respect to its policy i.e. the verification of the validity of the proxy certificate and the authorization decision making are performed in a logically single step.

The rest of this section is organized as follows: we describe the general setting of our proof-carrying proxy certification mechanism in Section 3.5.1. In Section 3.5.2, an illustration of our approach is given through the description of an application scenario. In Section 3.5.3, we discuss some approaches related to our concept of proof-carrying proxy certification.

3.5.1 General Setting

The setting for proof-carrying proxy certification is similar to the one defined for the policy-based signature primitive. Four types of players are considered: end users, credential issuers, agents and verifiers (service providers). We consider a public key infrastructure where each end user holds a pair of keys (pk_u, sk_u). An end user is identified by its public key pk_u. We consider a set of credential issuers $\mathcal{I} = \{I_1, \ldots, I_N\}$, where the public key of I_κ, for $\kappa \in \{1, \ldots, N\}$, is denoted R_κ while the corresponding master key is denoted s_κ. We assume that a trustworthy value of the public key of each of the credential issuers is known by the end users. Any credential issuer $I_\kappa \in \mathcal{I}$ may be asked by an end user to issue a credential corresponding to a set of statements. The requested credential is basically the digital signature of the credential issuer on an assertion denoted A^{pk_u}. Upon receiving a request for generating a credential on assertion A^{pk_u}, a credential issuer I_κ first checks the validity of the assertion. If it is valid, then I_κ returns the credential $\varsigma(R_\kappa, A^{pk_u})$. Otherwise, I_κ returns an error message.

Each service provider defines an authorization policy for each action on a sensitive resource it controls. When an end user wants to interact with the service provider (verifier) through an agent, it first generates a pair of keys (pk_a, sk_a) for the agent. Then, it specifies the content of the proxy certificate - a message, denoted M, containing the end user's public key pk_u, the public key of the agent pk_a and the delegation constraints. Finally, the end user generates a signature on the content of the proxy certificate using a policy-based signature algorithm.

When the agent decides to interact with the verifier, it provides its proof-carrying proxy certificate along with a proof of possession of the private key sk_a corresponding to the public key pk_u contained in the proxy certificate. The verifier first checks the delegation constraints specified by the proxy certificate to be sure that the agent is allowed by the end user to perform the requested action on its behalf. Then, it checks the validity of the signature on the content of the proxy certificate using the verification algorithm of the used policy-based signature scheme. This algorithm takes as input the proof-carrying proxy certificate, the end user's public key pk_u, and the authorization policy Pol^{pk_u}. At the end, the verifier obtains a proof that the agent whose public key is pk_a is allowed by an end user whose public key is pk_u to perform the action on its behalf and that the end user is compliant with the authorization policy specified by the verifier. The interactions between the end user, the agent and the verifier are depicted in Figure 3.3.

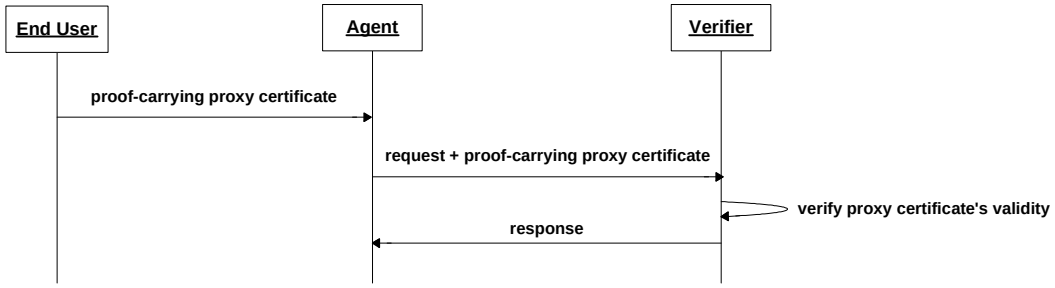


Figure 3.3: Proof-carrying proxy certificates

The policy-based signature scheme used for the generation of proof-carrying proxy certificates allows to meet the required security/privacy properties:

- **unforgeability**: the signature on a proof-carrying proxy certificate must not be valid with respect to policy Pol^{pk_u} if the signer does not use the private key sk_u or a qualified set of credentials for policy Pol^{pk_u}. In other words, on one hand, the agent cannot obtain a valid proof-carrying proxy certificate with respect to policy Pol^{pk_u} from a user that does not have access to the private key sk_u, and, on the other hand, the end user cannot generate a valid proof-carrying proxy certificate with respect to policy Pol^{pk_u} if it does not have access to a qualified set of credentials for the policy.
- **credential ambiguity**: in the case where there exists multiple qualified sets of credentials for policy Pol^{pk_u}, a valid proxy-carrying proxy certificate must not reveal which specific set of credentials has been used to generate the certificate.

In the following section, we describe a sample scenario illustrating the concept of proof-carrying proxy certificates.

3.5.2 An Application Scenario

Consider the following scenario: a researcher (end user) wants to perform some operations on various hosts on a scientific computation oriented grid environment. The operations can be executed independently, can depend on each other, or can be executed only at specific periods of time. From its laptop the researcher wants to submit a number of requests to the destination hosts and have the operations executed while he is doing other things including being offline. For each request, an authenticated connection needs to be established with the corresponding destination host. An authorization policy is associated to the operations and the researcher has to prove his compliance with the policy in order for the operations to be permitted. The researcher delegates the management of the different operations to one or more agents.

Currently, authorization in grid environments is identity-based. The researcher whose public/private key pair is denoted (pk_u, sk_u) holds an X.509 certificate binding his global identity to his public key. In order to make the agent act on his behalf, he generates for the agent a random pair of keys denoted (pk_a, sk_a). Then, he issues an X.509 proxy certificate [140] associated to the generated key pair. The certificate contains in addition to the agent's public key pk_a, a set of statements indicating the valid operations that the agent is allowed to perform on behalf of the researcher, as well as a restricted validity period. The authentication of the agent is therefore based on its key pair, the proxy certificate generated by the researcher and the public-key certificate of the researcher. Authorization to perform a specific task is based on the identity of the researcher (taken from his X.509 certificate) as well as on the statements within the proxy certificate.

As explained in [23], an identity-based approach to authorization and authentication for large grids "will not provide the scalability, flexibility, and ease of management that a large grid needs to control access to its sensitive resources", while a property-based approach where properties are carried by digital credentials is more appropriate. In scientific grids for instance, properties may include whether the requesting agent is acting on behalf of a professor, a student or an administrator, whether the agent is acting on behalf of a member of a particular research project whose membership list is not maintained locally, whether the agent is acting on behalf of a researcher from academy or industry, *etc.*

In the credential-based approach, the agent needs to prove that its owner (the researcher) is compliant with a specific credential-based authorization policy in order for the operations to be executed. Using standard credential systems such as X.509 attribute certificates, the agent needs to have access to the credentials of its owner to provide the necessary authorization arguments. For example, assume that a policy requires the researcher to be either a research staff member of company X or company Y. Suppose that the researcher is employed by company X,

he consequently has been issued a credential of the form $cred_X^u = \varsigma(I_X, pk_u : employee)$. In addition to the proxy certificate, the researcher gives to the agent the credential $cred_X^u$. During the authentication and authorization phase, the agent submits in addition to its proxy certificate, the researcher's credential $cred_X^u$. The remote host where the operation needs to be executed does the following: 1) check the validity of the proxy certificate using the public key pk_u, 2) check the validity of $cred_X^u$ using the public key of the 'trusted' credential issuer, 3) check whether the provided credential fulfills the authorization policy for the requested operations. If all the validity checks are successful, the task is executed. Otherwise, an error message is returned.

Using proof-carrying proxy certificates allows to combine the verification of the validity of the proxy certificate and the authorization decision making in a way that improves the privacy of the researcher. In fact, instead of using a standard signature scheme, the researcher generates the agent's proxy certificate by running a policy-based signature algorithm on input of his private key sk_u, his credential $cred_X^u$ and the policy '$\langle I_X, pk_u : employee\rangle \vee \langle I_Y, pk_u : employee\rangle$'. The new proxy certificate carries in addition to delegation rights, the authorization arguments necessary for the execution of the operations. Hence, instead of performing three validity checks, the remote host needs just to verify the validity of the proxy certificate with respect to the policy '$cred_X^u$ or $cred_Y^u$' using the researcher's public key pk_u. Furthermore, thanks to the credential ambiguity property, the remote host will not know whether the agent is acting on behalf of a company X or company Y.

3.5.3 Related Approaches

The intuition behind the concept of proof-carrying proxy certificates comes originally from proof-carrying codes [121]. The latter is a technique that can be used for safe execution of untrusted code. In a typical scenario, a code receiver establishes a set of safety rules that guarantee safe behavior of programs, and the code producer creates a formal safety proof that proves, for the untrusted code, adherence to the safety rules. Then, the receiver is able to use a proof validator to check that the proof is valid and hence the untrusted code is safe to execute.

By analogy with proof-carrying codes, a proof-carrying authentication mechanism based on higher-order logic was presented in [7]: the client desiring access must construct a proof using his attribute certificates, and the server will simply check the validity of the proof. The logic-based approach leads to a simple and efficient solution that integrates different authentication frameworks including X.509 and SPKI/SDSI. However, this approach cannot be used in the context of proof-carrying proxy certification because it does not provide a signature scheme fulfilling the required properties.

Providing a privacy preserving proof of compliance with a credential-based policy is a problem that has been studied in recent literature. In [9], Backes et al. exploit cryptographic zero-knowledge proofs to allow requesting users to prove their adherence with a credential-based policy. The proposed solution provides better privacy guarantees than our concrete implementation

of proof-carrying proxy certificates as the users may prove their compliance while preserving their anonymity. However, as the described protocol requires interaction between the credentials holder (end user) and the verifier, it can not be directly used to implement proof-carrying proxy certificates. An interesting line for future research would be to exploit the Fiat-Shamir heuristic [70] to transform their interactive protocols into a signature scheme that could be used to implement proof-carrying proxy certificates.

3.6 Conclusion

In this chapter, we formally defined our policy-based signature and the related security models. We proposed a provably secure policy-based signature scheme using bilinear pairings over elliptic curves. We finally proposed a novel form of proxy certificates, called proof-carrying proxy certificates, based on our policy-based signature primitive. In the following chapter, we summarize the research work presented in this thesis and discuss future research directions.

Conclusion

The last chapter of this manuscript summarizes the main contributions and discusses future research directions. It consists of two parts: while in the first part we discuss the theoretical aspects of our research work, we discuss in the second part the proposed applications of policy-based cryptography.

Theory

The theoretical part of this thesis consists of the three areas discussed below:

- **formal definitions**: we formalize the concept of policy-based cryptography by formally defining three policy-based cryptographic primitives: policy-based encryption, policy-based public-key encryption and policy-based signature. For each of these primitives, we define the related security notions. While the security models associated to our encryption primitives consider indistinguishability against chosen ciphertext attacks, the security model associated to our signature primitive considers existential unforgeability against chosen message attacks. By extending well-established security models for asymmetric cryptographic schemes, we came up with new models that are suited to the particular setting of policy-based cryptography.

  An interesting line of future research would be to define advanced policy-based cryptographic primitives with more sophisticated features. In particular, a possible extension of our primitives would be to consider hierarchies of credential issuers. From this perspective, the extension of the concept of hierarchical identity-based cryptography to the policy-based setting seems to be a promising direction for future research.

- **implementations**: for each of the defined policy-based cryptographic primitives, we proposed an implementation based on bilinear pairings over elliptic curves. Bilinear pairings recently proved to be extremely useful in cryptography thanks to a set of properties that cannot be achieved using standard cryptographic primitives. In addition to being a new approach in cryptography, our policy-based cryptographic primitives can be seen as an other illustration of the usefulness of bilinear pairings. Moreover, as discussed in the different chapters of this thesis, our schemes are at least as efficient as the related schemes found so far in the literature.

  The essential operation in pairing-based cryptography is pairing computation. The design of policy-based cryptographic schemes presented in this thesis ensures that the total number of pairing computations depends linearly on the number of credentials specified by a policy according to which a message is encrypted or signed. Although such operation can be mathematically and practically optimized, it still has to be minimized in future designs of policy-based encryption and signature schemes. In addition to computational costs, bandwidth consumption need to be taken into account during the design of cryptographic schemes. The properties of bilinear pairings and the adopted policy model whereby policies are written in standard normal forms allowed us to considerably reduce the size of the ciphertexts and signatures produced by our policy-based encryption and policy-based signature schemes respectively. Ideally, policy-based encryption and policy-based signature schemes should generate constant-size ciphertexts and signatures. Two starting points on this line of research would be, on one hand, the research work on constant-size hierarchical identity-based encryption schemes [37] and, on the other hand, the constant-size ring signature scheme derived from the anonymous identification scheme proposed in [64].

- **security proofs**: we prove the security of our policy-based cryptographic schemes under the associated security models in the random oracle model. The reductionist security proofs of our policy-based encryption scheme and policy-based public-key encryption scheme rely on the Fujisaki-Okamoto transformations, whereas the reductionist security proof of our policy-based signature scheme follows the oracle replay technique.

  Although sufficient in the context of this thesis, our reductionist security proofs remain theoretical. A further improvement of our security analysis would be to consider the concept of practice-oriented provable security, first introduced by Bellare in [25]. The idea is to explicitly capture the quantitative aspects of security, such as the number of cycles of adversary computation the scheme can withstand or the size of the used security parameter. Along the same lines, the reductionist proofs to be developed need to preserve as much as possible the strength of the mathematical problems the hardness of which guarantees the security of the proposed policy-based cryptographic schemes. In other words, one needs to develop tighter reductions, since this directly affects the practical efficiency of the proposed schemes with respect to the claimed security level.

  Finally, the security proofs proposed in this thesis are performed in the random oracle model. Another interesting direction for research would focus on evaluating the security of policy-based cryptographic schemes with respect to ‘standard’ models without random oracles.

Now that we summarized our theoretical contributions and discussed some related future directions, we turn to possible applications of policy-based cryptography as presented in this thesis.

Applications

In this thesis, we described the following applications of policy-based cryptography:

- **controlling access to XML documents**: the most intuitive application of policy-based encryption is as an enforcement mechanism of access control policies. In this context, we presented a framework for controlling access to released XML documents using our policy-based encryption primitive. In contrast with the approaches found in the literature, ours allows an entity to delegate all or part of its policy enforcement process to trusted third parties, while enhancing the privacy of data consumers. For the sake of simplicity, the framework presented in this thesis deals with the most basic representation of XML documents. As future work, one can extend the framework to support advanced access control features such as policies associated to specific attributes of an XML document, policies associated to DTDs, and policies with different propagation options.
- **sticky privacy policies**: we showed how our policy-based encryption primitive can be used to implement the sticky policy paradigm, which can be considered as one of the fundamental principles of privacy protection over the Internet. Naturally, the policy-based encryption primitive is to be used in combination with other tools such as refinement checking algorithms, auditing and monitoring techniques, trusted platforms, *etc.*
- **establishment of ad-hoc communities**: we presented an application scenario where the policy-based encryption primitive is used to achieve privacy-enhanced establishment of ad-hoc communities. More precisely, we showed that the particular features of policy-based encryption allow to meet the privacy requirement of data minimization according to which only strictly necessary information should be collected for a given purpose. The presented approach can be extended to support any kind of ad-hoc communities such as the ones formed in peer-to-peer networks.
- **automated trust negotiation**: our policy-based public-key encryption allows to overcome the collusion attacks that are inherent to our original policy-based encryption primitive. We showed how the proposed scheme can be used to solve the cyclic policy interdependency problem in the context of automated trust negotiation, while improving the solutions found in literature.
- **proof-carrying proxy certification**: we presented a novel form of proxy certificates, called proof-carrying proxy certificates, which is based on our policy-based signature primitive. As for standard proxy certificates, a proof-carrying proxy certificate allows an entity to delegate responsibility to another entity so that it can perform certain actions on its behalf. Additionally, the proof-carrying proxy certificate proves the compliance of the first entity with a specific trust establishment or authorization policy.

Our concept of policy-based cryptography can be viewed to some extend as a generalization of the concept of identity-based cryptography. Indeed, an identity-based cryptographic scheme is a policy-based cryptographic scheme whereby policies are restricted to single conditions fulfilled by identity-based credentials. As mentioned by Shamir in [137], identity-based cryptography is "ideal for closed groups of users such as the executives of a multinational company or the branches of a large bank, since the headquarters of the corporation can serve as a key generation center". In large-scale open environments like the Internet, the identity attribute either is insufficient or irrelevant for authorization and trust establishment and should be replaced by policies fulfilled by assertion-based credentials. From this perspective, policy-based cryptography can be viewed as an alternative for identity-based cryptography that better copes with the specific requirements of such environments. Finally, both identity-based and policy-based cryptography should be considered as a good complement of traditional symmetric and asymmetric cryptography.

Bibliography

[1] XML Encryption Syntax and Processing, December 2002.

[2] C. Adams and S. Lloyd. *Understanding Public-Key Infrastructure: Concepts, Standards, and Deployment Considerations*. Macmillan Technical Publishing, Indianapolis, USA, 1999.

[3] S. Al-Riyami. Cryptographic schemes based on elliptic curve pairings. Ph.D. Thesis, Royal Holloway, University of London, 2004.

[4] S. Al-Riyami, J. Malone-Lee, and N. Smart. Escrow-free encryption supporting cryptographic workflow. Cryptology ePrint Archive, Report 2004/258, 2004. `http://eprint.iacr.org/`.

[5] S. Al-Riyami and K. Paterson. Certificateless public key cryptography. In *ASIACRYPT*, pages 452–473. Springer-Verlag, 2003.

[6] A. Anderson. Predicates for boolean web service policy languages. In *WWW 2005 Workshop on Policy Management for the Web*, 2005.

[7] A. Appel and E. Felten. Proof-carrying authentication. In *ACM Conference on Computer and Communications Security*, pages 52–62, 1999.

[8] P. Ashley, S. Hada, G. Karjoth, C. Powers, and M. Schunter. Enterprise Privacy Authorization Language (EPAL 1.0), March 2003.

[9] M. Backes, J. Camenisch, and D. Sommer. Anonymous yet accountable access control. In *WPES '05: Proceedings of the 2005 ACM workshop on Privacy in the electronic society*, pages 40–46, New York, NY, USA, 2005. ACM Press.

[10] J. Baek, R. Safavi-Naini, and W. Susilo. Efficient multi-receiver identity-based encryption and its application to broadcast encryption. In *Public Key Cryptography*, pages 380–397, 2005.

[11] J. Baek and Y. Zheng. Identity-based threshold decryption, 2004.

[12] J. Baek and Y. Zheng. Identity-based threshold signature scheme from the bilinear pairings. In *ITCC '04: Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'04) Volume 2*, page 124, Washington, DC, USA, 2004. IEEE Computer Society.

[13] W. Bagga, M. Backes, G. Karjoth, and Matthias Schunter. Efficient comparison of enterprise privacy policies. In *SAC '04: Proceedings of the 2004 ACM symposium on Applied computing*, pages 375–382, New York, NY, USA, 2004. ACM Press.

[14] W. Bagga and L. Bussard. Distance-bounding proof of knowledge to avoid real-time attacks. In *SEC*, pages 223–238, 2005.

[15] W. Bagga, S. Crosta, P. Michiardi, and R. Molva. Establishment of ad-hoc communities through policy-based cryptography, 2006. To appear in Proceedings of Workshop on Cryptography for Ad hoc Networks (WCAN'06).

[16] W. Bagga, S. Crosta, and R. Molva. Proof-carrying proxy certificates. In *Proceedings of Conference on Security and Cryptography for Networks (SCN'06)*, volume 4116 of *LNCS*, pages 321–335. Springer-Verlag, 2006.

[17] W. Bagga and R. Molva. Policy-based cryptography and applications. In *Proceedings of Financial Cryptography and Data Security (FC'05)*, volume 3570 of *LNCS*, pages 72–87. Springer-Verlag, 2005.

[18] W. Bagga and R. Molva. Collusion-free policy-based encryption, 2006. To appear in Proceedings of Information Security Conference (ISC'06).

[19] W. Bagga, R. Molva, and S. Crosta. Policy-based encryption schemes from bilinear pairings. In *ASIACCS '06: Proceedings of the 2006 ACM Symposium on Information, computer and communications security*, pages 368–368, New York, NY, USA, 2006. ACM Press.

[20] D. Balfanz, G. Durfee, N. Shankar, D. Smetters, J. Staddon, and H. Wong. Secret handshakes from pairing-based key agreements. In *SP '03: Proceedings of the 2003 IEEE Symposium on Security and Privacy*, page 180, Washington, DC, USA, 2003. IEEE Computer Society.

[21] P. Barreto, H. Kim, B. Lynn, and M. Scott. Efficient algorithms for pairing-based cryptosystems. In *Proceedings of the 22nd Annual International Cryptology Conference on Advances in Cryptology*, pages 354–368. Springer-Verlag, 2002.

[22] P. Barretto. The pairing-based crypto lounge. `http://paginas.terra.com.br/informatica/paulobarreto/pblounge.html`.

[23] J. Basney, W. Nejdl, D. Olmedilla, V. Welch, and M. Winslett. Negotiating trust on the grid. In *2nd WWW Workshop on Semantics in P2P and Grid Computing*, New York, USA, May 2004.

[24] O. Baudron, D. Pointcheval, and J. Stern. Extended notions of security for multicast public key cryptosystems. In *ICALP '00: Proceedings of the 27th International Colloquium on Automata, Languages and Programming*, pages 499–511, London, UK, 2000. Springer-Verlag.

[25] M. Bellare. Practice-oriented provable-security. In *ISW '97: Proceedings of the First International Workshop on Information Security*, pages 221–231. Springer-Verlag, 1998.

[26] M. Bellare, A. Boldyreva, and S. Micali. Public-key encryption in a multi-user setting: Security proofs and improvements. In *EUROCRYPT*, pages 259–274, 2000.

[27] M. Bellare, A. Boldyreva, and J. Staddon. Randomness re-use in multi-recipient encryption schemeas. In *PKC '03: Proceedings of the 6th International Workshop on Theory and Practice in Public Key Cryptography*, pages 85–99, London, UK, 2003. Springer-Verlag.

[28] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A concrete security treatment of symmetric encryption. In *FOCS '97: Proceedings of the 38th Annual Symposium on Foundations of Computer Science (FOCS '97)*, page 394, Washington, DC, USA, 1997. IEEE Computer Society.

[29] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among notions of security for public-key encryption schemes. In *CRYPTO '98: Proceedings of the 18th Annual International Cryptology Conference on Advances in Cryptology*, pages 26–45. Springer-Verlag, 1998.

[30] M. Bellare, C. Namprempre, and G. Neven. Security proofs for identity-based identification and signature schemes. In Cachin and Camenisch [45], pages 268–286.

[31] M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *Proceedings of the 1st ACM conference on Computer and communications security*, pages 62–73. ACM Press, 1993.

[32] J. Benaloh and J. Leichter. Generalized secret sharing and monotone functions. In *CRYPTO '88: Proceedings on Advances in cryptology*, pages 27–35, New York, NY, USA, 1990. Springer-Verlag New York, Inc.

[33] E. Bertino and E. Ferrari. Secure and selective dissemination of xml documents. *ACM Trans. Inf. Syst. Secur.*, 5(3):290–331, 2002.

[34] E. Bertino, E. Ferrari, and A. Squicciarini. Trust negotiations: Concepts, systems, and languages. *Computing in Science and Engg.*, 6(4):27–34, 2004.

[35] I. Blake, G. Seroussi, and N. Smart. *Elliptic curves in cryptography*. Cambridge University Press, New York, NY, USA, 1999.

[36] G. Blakley. Safeguarding cryptographic keys. In *1979 National Computer Conference: June 4–7, 1979, New York, New York*, pages 313–317, 1979.

[37] D. Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In Ronald Cramer, editor, *EUROCRYPT*, volume 3494 of *Lecture Notes in Computer Science*, pages 440–456. Springer, 2005.

[38] D. Boneh and M. Franklin. Identity-based encryption from the weil pairing. In *Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology*, pages 213–229. Springer-Verlag, 2001.

[39] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the weil pairing. In *Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security*, pages 514–532. Springer-Verlag, 2001.

[40] N. Borselius, C. Mitchell, and A. Wilson. On the value of threshold signatures. *SIGOPS Oper. Syst. Rev.*, 36(4):30–35, 2002.

[41] L. Bouganim, F. Dang Ngoc, and P. Pucheral. Client-based access control management for xml documents. In Mario A. Nascimento, M. Tamer Özsu, Donald Kossmann, Renée J. Miller, José A. Blakeley, and K. Bernhard Schiefer, editors, *VLDB*, pages 84–95. Morgan Kaufmann, 2004.

[42] R. Bradshaw, J. Holt, and K. Seamons. Concealing complex policies with hidden credentials. Cryptology ePrint Archive, Report 2004/109, 2004. `http://eprint.iacr.org/`.

[43] S. Brands. *Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy*. MIT Press, 2000.

[44] S. Brands. *Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy*. MIT Press, Cambridge, MA, USA, 2000.

[45] C. Cachin and J. Camenisch, editors. *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, volume 3027 of *Lecture Notes in Computer Science*. Springer, 2004.

[46] J. Camenisch and E. Van Herreweghen. Design and implementation of the idemix anonymous credential system. In *CCS '02: Proceedings of the 9th ACM conference on Computer and communications security*, pages 21–30, New York, NY, USA, 2002. ACM Press.

[47] J. Camenisch and A. Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *EUROCRYPT*, volume 2045 of *Lecture Notes in Computer Science*, pages 93+, 2001.

[48] R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited. *J. ACM*, 51(4):557–594, 2004.

[49] C. Castelluccia, S. Jarecki, and G. Tsudik. Secret handshakes from ca-oblivious encryption, 2004.

[50] Z. Chai, Z. Cao, and Y. Zhou. Efficient id-based broadcast threshold decryption in ad hoc network. In *IMSCCS*, volume 2, pages 148–154, 2006.

[51] D. Chaum. Security without identification: transaction systems to make big brother obsolete. *Commun. ACM*, 28(10):1030–1044, 1985.

[52] L. Chen, K. Harrison, D. Soldera, and N. Smart. Applications of multiple trust authorities in pairing based cryptosystems. In *Proceedings of the International Conference on Infrastructure Security*, pages 260–275. Springer-Verlag, 2002.

[53] R. Chinnici, G. Daniels, A. Karmarkar, A. Lewis, and U. Yalcinalp. Proposal for adding compositors to wsdl 2.0, 2004. `http://lists.w3.org/Archives/Public/www-ws-desc/2004Jan/0153.html`.

[54] J. Choi, K. Sakurai, and J. Park. Proxy certificates-based digital fingerprinting scheme for mobile communication. In *IEEE 37th Annual 2003 International Carnahan Conference on Security*, pages 587 – 594. IEEE Computer Society, 2003.

[55] S. Chow, R. Lui, L. Chi Kwong Hui, and S. Yiu. Identity based ring signature: Why, how and what next. In *EuroPKI*, pages 144–161, 2005.

[56] S. Chow, S., and L. Chi Kwong Hui. Efficient identity based ring signature. In *ACNS*, pages 499–512, 2005.

[57] J. Claessens, B. Preneel, and J. Vandewalle. (how) can mobile agents do secure electronic transactions on untrusted hosts? a survey of the security issues and the current solutions. *ACM Trans. Inter. Tech.*, 3(1):28–48, 2003.

[58] A. Cohen. The symbolic construction of community, 1985. London: Tavistock.

[59] L. Cranor, M. Langheinrich, M. Marchiori, M. Presler-Marshall, and J. Reagle. The Platform for Privacy Preferences 1.0 (P3P1.0) Specification, April 2002.

[60] E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, and P. Samarati. A fine-grained access control system for xml documents. *ACM Trans. Inf. Syst. Secur.*, 5(2):169–202, 2002.

[61] A. Dent. A survey of certificateless encryption schemes and security models. Cryptology ePrint Archive, Report 2006/211, 2006. `http://eprint.iacr.org/`.

[62] Y. Desmedt. Threshold cryptography. *European Transactions on Telecommunications*, 5(4):449–457, July 1994.

[63] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.

[64] Y. Dodis, A. Kiayias, A. Nicolosi, and V. Shoup. Anonymous identification in ad hoc groups. In Cachin and Camenisch [45], pages 609–626.

[65] Y. Dodis and M. Yung. Exposure-resilience for free: The hierarchical id-based encryption case. In *IEEE Security in Storage Workshop*, pages 45–52, 2002.

[66] K. Eisenträger, K. Lauter, and P. Montgomery. Fast elliptic curve arithmetic and improved weil pairing evaluation. In Marc Joye, editor, *CT-RSA*, volume 2612 of *Lecture Notes in Computer Science*, pages 343–354. Springer, 2003.

[67] C. Ellison. SPKI certificate documentation, 1998. `http://www.std.com/~cme/html/spki.html`.

[68] A. Fiat and M. Naor. Broadcast encryption. In *CRYPTO '93: Proceedings of the 13th annual international cryptology conference on Advances in cryptology*, pages 480–491, New York, NY, USA, 1994. Springer-Verlag New York, Inc.

[69] A. Fiat and A. Shamir. How to prove yourself: practical solutions to identification and signature problems. In *Proceedings on Advances in cryptology—CRYPTO '86*, pages 186–194. Springer-Verlag, 1987.

[70] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology — Crypto '86*, pages 186–194, New York, 1987. Springer-Verlag.

[71] U. Fiege, A. Fiat, and A. Shamir. Zero knowledge proofs of identity. In *Proceedings of the 19th annual ACM conference on Theory of computing*, pages 210–217. ACM Press, 1987.

[72] Organization for Economic Cooperation and Development (OECD). Recommendation of the council concerning guidelines governing the protection of privacy and transborder flows of personal data, 1980. `http://www.oecd.org/home/`.

[73] Organization for the Advancement of Structured Information Standards (OASIS). Security assertion markup language, 2002. `http://www.oasis-open.org/committees/security/`.

[74] Organization for the Advancement of Structured Information Standards (OASIS). Xacml profile for web services-working draft, 2003. `http://www.oasis-open.org/committees/download.php/3661/draft-xacml-wspl-04.pdf`.

[75] Organization for the Advancement of Structured Information Standards (OASIS). Assertions and protocols for the oasis security assertion markup language (saml) v2.0, 2005. `http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf`.

[76] Y. Frankel and Y. Desmedt. Parallel reliable threshold multisignature. Technical Report, University of Wisconsin-Milwaukee, TR-92-04-02, April, 1992.

[77] G. Frey, M. Muller, and H. Ruck. The tate pairing and the discrete logarithm applied to elliptic curve cryptosystems, 1999.

[78] E. Fujisaki and T. Okamoto. How to enhance the security of public-key encryption at minimum cost. In *Public Key Cryptography (PKC)*, pages 53–68, 1999.

[79] E. Fujisaki and T. Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In *Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, pages 537–554. Springer-Verlag, 1999.

[80] I. Fundulaki and M. Marx. Specifying access control policies for xml documents with xpath. In *SACMAT '04: Proceedings of the ninth ACM symposium on Access control models and technologies*, pages 61–69, New York, NY, USA, 2004. ACM Press.

[81] D. Galindo. Boneh-franklin identity based encryption revisited. To appear in Proceedings of 32nd International Colloquium on Automata, Languages and Programming (ICALP 2005).

[82] T. El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Proceedings of CRYPTO 84 on Advances in cryptology*, pages 10–18, New York, NY, USA, 1985. Springer-Verlag New York, Inc.

[83] T. Garefalakis. The generalized weil pairing and the discrete logarithm problem on elliptic curves. *Theor. Comput. Sci.*, 321(1):59–72, 2004.

[84] M. Girault. Self-certified public keys. In *EUROCRYPT*, pages 490–497, 1991.

[85] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[86] S. Goldwasser, S. Micali, and R. L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17(2):281–308, 1988.

[87] D. Hankerson, A. Menezes, and S. Vanstone. *Guide to Elliptic Curve Cryptography*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2003.

[88] J. Herranz. A formal proof of security of zhang and kim's id-based ring signature scheme. In *WOSIS'04*, pages 63–72. INSTICC Press, 2004. ISBN 972-8865-07-4.

[89] J. Herranz and G. Saez. New identity-based ring signature schemes. In *ICICS*, pages 27–39, 2004.

[90] F. Hess. Pseudonym systems. In *SAC '03: Proceedings of the 9th Annual International Workshop on Selected Areas in Cryptography*, volume 2595 of *LNCS*, pages 310–324. Springer-Verlag, 2003.

[91] J. Holt. Reconciling ca-oblivious encryption, hidden credentials, osbe and secret handshakes. Cryptology ePrint Archive, Report 2005/215, 2005.

[92] J. Holt, R. Bradshaw, K. Seamons, and H. Orman. Hidden credentials. In *Proc. of the 2003 ACM Workshop on Privacy in the Electronic Society*. ACM Press, 2003.

[93] ITU Telecommunication Standardization Sector (ITU-T). Itu-t recommendation x.509: The directory – authentication framework, 1997.

[94] J. Jonsson and B. Kaliski. Public-key cryptography standards (pkcs) #1: Rsa cryptography specifications version 2.1. *RFC 3447*, February 2003.

[95] A. Joux. A one round protocol for tripartite diffie-hellman. In *ANTS-IV: Proceedings of the 4th International Symposium on Algorithmic Number Theory*, pages 385–394, London, UK, 2000. Springer-Verlag.

[96] A. Joux. The weil and tate pairings as building blocks for public key cryptosystems. In *Proceedings of the 5th International Symposium on Algorithmic Number Theory*, pages 20–32. Springer-Verlag, 2002.

[97] J. Kahn. Entropy, independent sets and antichains: a new approach to dedekind's problem. In *Proc. Amer. Math. Soc. 130*, pages 371–378, 2002.

[98] G. Karjoth, M. Schunter, , and M. Waidner. The platform for enterprise privacy practices–privacy-enabled management of customer data. In *2nd Workshop on Privacy Enhancing Technologies (PET 2002)*, volume 2482 of *LNCS*, pages 69–84. Springer-Verlag, April 2002.

[99] S. Keoh, E. Lupu, and M. Sloman. Peace: A policy-based establishment of ad-hoc communities. In *Proceedings of the 20th Annual Computer Security Applications Conference (ACSAC'04)*, pages 386–395. IEEE Computer Society, 2004.

[100] D. Kleitman. On dedekind's problem: the number of monotone boolean functions. In *Proc. Amer. Math. Soc. 21*, pages 677–682, 1969.

[101] N. Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48(177):203–209, 1987.

[102] L. Kohnfelder. *Toward a Practical Public-Key Cryptosystem*. MIT Press, Cambridge, MA, USA, 1978.

[103] M. Kudo and S. Hada. Xml document security based on provisional authorization. In *CCS '00: Proceedings of the 7th ACM conference on Computer and communications security*, pages 87–96, New York, NY, USA, 2000. ACM Press.

[104] K. Kurosawa. Multi-recepient public-key encryption with shortened ciphertext. In *Public Key Cryptography (PKC)*, 2002.

[105] C. Laih and L. Harn. Generalized threshold cryptosystems. In *ASIACRYPT '91: Proceedings of the International Conference on the Theory and Applications of Cryptology*, pages 159–166, London, UK, 1993. Springer-Verlag.

[106] B. Lee and K. Kim. Self-certified signatures. In *INDOCRYPT '02: Proceedings of the Third International Conference on Cryptology*, pages 199–214, London, UK, 2002. Springer-Verlag.

[107] J. Li and N. Li. Oacerts: Oblivious attribute certificates. Technical Report TR-2005-26.

[108] J. Li and N. Li. Policy-hiding access control in open environment. In *PODC '05: Proceedings of the twenty-fourth annual ACM SIGACT-SIGOPS symposium on Principles of distributed computing*, pages 29–38, New York, NY, USA, 2005. ACM Press.

[109] J. Li, N. Li, and W. Winsborough. Automated trust negotiation using cryptographic credentials. In *CCS '05: Proceedings of the 12th ACM conference on Computer and communications security*, pages 46–57, New York, NY, USA, 2005. ACM Press.

[110] N. Li, W. Du, and D. Boneh. Oblivious signature-based envelope. In *Proceedings of the 22nd annual symposium on Principles of distributed computing*, pages 182–189. ACM Press, 2003.

[111] N. Li, W. Du, and D. Boneh. Oblivious signature-based envelope. *Distrib. Comput.*, 17(4):293–302, 2005.

[112] B. Libert and J. Quisquater. Efficient revocation and threshold pairing based cryptosystems. In *PODC '03: Proceedings of the twenty-second annual symposium on Principles of distributed computing*, pages 163–171, 2003.

[113] C. Lin and T. Wu. An identity-based ring signature scheme from bilinear pairings. Cryptology ePrint Archive, Report 2003/117, 2003. `http://eprint.iacr.org/`.

[114] A. Lysyanskaya, R. Rivest, A. Sahai, and S. Wolf. Pseudonym systems. In *SAC '99: Proceedings of the 6th Annual International Workshop on Selected Areas in Cryptography*, pages 184–199, London, UK, 2000. Springer-Verlag.

[115] A. Lysyanskaya, R. Rivest, A. Sahai, and S. Wolf. Pseudonym systems. In *SAC '99: Proceedings of the 6th Annual International Workshop on Selected Areas in Cryptography*, pages 184–199, London, UK, 2000. Springer-Verlag.

[116] A. Menezes, P. Van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*. CRC, October 1996.

[117] A. Menezes, S. Vanstone, and T. Okamoto. Reducing elliptic curve logarithms to logarithms in a finite field. In *STOC '91: Proceedings of the twenty-third annual ACM symposium on Theory of computing*, pages 80–89, New York, NY, USA, 1991. ACM Press.

[118] G. Miklau and D. Suciu. Cryptographically enforced conditional access for XML. Fifth International Workshop on the Web and Databases (WebDB 2002), June 2002.

[119] G. Miklau and D. Suciu. Controlling access to published data using cryptography. In *International Conference on Very Large Data Bases*, pages 898–909, September 2003.

[120] V. Miller. Use of elliptic curves in cryptography. In *Lecture notes in computer sciences; 218 on Advances in cryptology—CRYPTO 85*, pages 417–426, New York, NY, USA, 1986. Springer-Verlag New York, Inc.

[121] G. Necula. Proof-carrying code. In *POPL '97: Proceedings of the 24th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 106–119, New York, NY, USA, 1997. ACM Press.

[122] B. Clifford Neuman. Proxy-based authorization and accounting for distributed systems. In *International Conference on Distributed Computing Systems*, pages 283–291, 1993.

[123] National Institute of Standards and Technology (NIST) FIPS. Data encryption standard, 1976.

[124] IONA Technologies Position Paper. Position paper, 2004. `http://www.w3.org/2004/08/ws-cc/rbcc-20040902`.

[125] H. Petersen and P. Horster. Self-certified keys: Concepts and applications, 1997.

[126] D. Pointcheval and J. Stern. Security proofs for signature schemes. *Lecture Notes in Computer Science*, 1070:387, 1996.

[127] D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology: the journal of the International Association for Cryptologic Research*, 13(3):361–396, 2000.

[128] K. Ohgishi R. Sakai and M. Kasahara. Cryptosystems based on pairings. In *Proceedings of the symposium on cryptography and information security (SCIS'00)*, 2000.

[129] E. Ray and C. Maden. *Learning XML*. O'Reilly & Associates, Inc., Sebastopol, CA, USA, 2001.

[130] R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.

[131] R. Rivest, A. Shamir, and Y. Tauman. How to leak a secret. In *Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security*, pages 552–565. Springer-Verlag, 2001.

[132] P. Rogaway and T. Shrimpton. Cryptographic hash-function basics: Definitions, implications and separations for preimage resistance, second-preimage resistance, and collision resistance. Cryptology ePrint Archive, Report 2004/035, 2004. `http://eprint.iacr.org/`.

[133] S. Saeednia. Identity-based and self-certified key-exchange protocols. In *ACISP '97: Proceedings of the Second Australasian Conference on Information Security and Privacy*, pages 303–313, London, UK, 1997. Springer-Verlag.

[134] S. Saeednia. A note on girault's self-certified model. *Inf. Process. Lett.*, 86(6):323–327, 2003.

[135] C. Schnorr. Efficient identification and signatures for smart cards. In *EUROCRYPT '89: Proceedings of the workshop on the theory and application of cryptographic techniques on Advances in cryptology*, pages 688–689, New York, NY, USA, 1990. Springer-Verlag New York, Inc.

[136] A. Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

[137] A. Shamir. Identity-based cryptosystems and signature schemes. In *Proceedings of CRYPTO 84 on Advances in cryptology*, pages 47–53. Springer-Verlag New York, Inc., 1985.

[138] N. Smart. Access control using pairing based cryptography. In *Proceedings CT-RSA 2003*, pages 111–121. Springer-Verlag LNCS 2612, April 2003.

[139] D. Stinson and R. Wei. Bibliography on secret sharing schemes. `http://www.cacr.math.uwaterloo.ca/~dstinson/ssbib.html`.

[140] S. Tuecke, V. Welch, D. Engert, L. Pearlman, and M. Thompson. Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile. *RFC 3820*, June 2004.

[141] E. Verheul. Evidence that xtr is more secure than supersingular elliptic curve cryptosystems. In *EUROCRYPT '01: Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques*, pages 195–210, London, UK, 2001. Springer-Verlag.

[142] World Wide Web Consortium (W3C). Xml path language (xpath) 1.0, 1999. `http://web5.w3.org/TR/xpath`.

[143] World Wide Web Consortium (W3C). Web services description language (wsdl) 1.1, 2001. `http://www.w3.org/TR/wsdl`.

[144] World Wide Web Consortium (W3C). Web services policy 1.2 - framework (ws-policy), 2002. `http://www.w3.org/Submission/WS-Policy/`.

[145] World Wide Web Consortium (W3C). Extensible markup language (xml), 2003. `www.w3.org/XML/`.

[146] W. Winsborough, K. Seamons, and V. Jones. Automated trust negotiation. Technical Report TR-2000-05, Raleigh, NC, USA, 24 2000.

[147] ANSI X9.62 and FIPS 186-2. Elliptic curve digital signature algorithm, 1998.

[148] Y. Yacobi. A note on the bilinear diffie-hellman assumption. Cryptology ePrint Archive, Report 2002/113, 2002. `http://eprint.iacr.org/`.

[149] M. Yague. Survey on xml-based policy languages for open environments. *Information Assurance and Security*, 1(1):11–20, 2006.

[150] F. Zhang and K. Kim. ID-based blind signature and ring signature from pairings. In *ASIACRYPT*, pages 533–547. Springer-Verlag LNCS 2501, 2002.

[151] F. Zhang, R. Safavi-Naini, and W. Susilo. An efficient signature scheme from bilinear pairings and its applications. In *Public Key Cryptography*, pages 277–290, 2004.