TELECOM
PARIS

EURECOM
SOPHIA ANTIPOLIS

Protocoles d'Etablissement de Confiance pour Objets Communicants

(Trust Establishment Protocols for Communicating Devices)

THÈSE

présentée pour obtenir le grade de docteur

de l'Ecole Nationale Supérieure des Télécommunications
Spécialité : Informatique et réseaux

par

Laurent Bussard

Soutenue publiquement le 15 Octobre 2004 devant le jury composé de

David Powell	Président
Alexis Bonnecaze	Rapporteur
Marc Joye	Rapporteur
Frédéric Cuppens	Examinateur
Refik Molva	Directeur de thèse
Yves Roudier	Co-encadrant

to Cécile, Quentin, and Margot

Abstract

With the advent of self-organizing systems such as ad hoc networks or pervasive computing, security protocols have to meet a new requirement for establishing trust among parties that have no a priori relationship such as a shared naming structure or a common organization. Trust establishment in this context calls for a new paradigm with respect to classical scenarios whereby entities build trust based on some existing security association. This thesis suggests cryptographic protocols through which some party can build trust based on the history of its interactions with other parties. Those protocols allow a party to get a proof of history, i.e. the evidence that it was involved in some interaction with another party. During further interactions, other parties consider the prover trustworthy based on the verification of the history.

Privacy is an essential requirement for such a protocol since providing a proof of history to several parties without privacy would severely expose the behavior of the prover. In this work, we propose a dedicated scheme for unlinkable credentials that ensures the anonymity of the prover and the unlinkability of its interactions. This scheme is an extension of group signatures and enables the prover to choose which part of his history is disclosed when submitting a proof.

Another approach consists of using evidence of physical location as a means of building trust based on the locality of communicating parties. We define the distance-bounding proof of knowledge scheme that combines a distance measurement technique and a cryptographic mechanism in order to verify the proximity of a party knowing a secret like a private key. This mechanism can be used when delivering a proof of interaction or a location stamp.

Last we consider a possible architecture for establishing trust based on history. Our approach combines unlinkable credentials and distance-bounding proofs of knowledge. Thanks to this new scheme, we can show that trust among unknown parties can be built while preserving their privacy. The results of a preliminary implementation are discussed.

Résumé

Avec l'arrivée des systèmes auto-organisés tels que les réseaux ad hoc ou l'informatique diffuse, les protocoles de sécurité doivent répondre à de nouveaux besoins. Dans ce travail nous étudions comment une relation de confiance peut être établie entre des entités qui ne se connaissent pas a priori. Nous proposons des protocoles de sécurité permettant à une entité de garder un historique de ses interactions : après chaque interaction, une preuve est délivrée par exemple sous la forme d'une recommandation ou d'une preuve d'interaction. Chaque preuve concernant une entité est stockée par celle-ci dans son historique. Les preuves peuvent être sélectivement démontrées, lorsqu'il est nécessaire de dévoiler une partie de son historique pour établir une relation de confiance.

Prouver son historique à différentes entités révèle en général trop d'information et des mécanismes pour protéger la vie privée des utilisateurs sont nécessaires. Dans ce but, nous proposons un mécanisme de certificat non-traçable qui empêche de faire le lien entre plusieurs preuves et qui protège l'anonymat des utilisateurs. Ce schéma est une extension des signatures de groupe où le signataire n'est plus un membre anonyme d'un groupe mais le détenteur d'un historique.

Un autre besoin récurrent de l'informatique diffuse est la création d'un lien entre les entités virtuelles et le monde physique qui les entoure. Dans ce but, nous proposons les preuves de proximité qui combinent une mesure de distance et de la cryptographie dans le but de vérifier la proximité d'une entité connaissant un secret, par exemple une clé privée. Ce mécanisme peut être utilisé durant une interaction pour obtenir une preuve de cette interaction ou une preuve de localisation.

Finalement, nous combinons ces deux mécanismes pour définir une architecture dédiée à l'établissement de confiance basé sur un historique. Ce schéma nous permet de confirmer la thèse qu'il est possible d'établir une relation de confiance en protégeant sa vie privée. Les résultats d'une première implémentation sont également discutés.

Un résumé détaillé des résultats de cette thèse est disponible à la fin de ce manuscrit (pages 185 à 203).

Acknowledgements

This thesis is the result of three and a half years of work whereby I have been accompanied and supported by many people. I have now the opportunity to express my gratitude for all of them.

The first person I would like to thank is my supervisor Refik Molva. I have worked in his team (network-security team at Eurecom) since May 2001 when I started this Ph.D. thesis. During these years he gave me the opportunity to work on mobile code protection, privacy, proof of location, and trust establishment and always supported me. I owe him lots of gratitude for having shown me this way of research.

I wish to express my gratitude to my co-supervisor Yves Roudier who kept a closer eye on the progress of my work. During these years, I visited his office daily and he always was available when I needed his advices.

It has been a pleasure to work in the NS-Team. The valuable critics and comments from Marc Dacier, Pietro Michiardi, Melek Önen, Jean-Christophe Pazzaglia, and Fabien Pouget were very fruitful. Collaborating with Walid Bagga was a pleasure. He helped me to solve a problem that I had in mind since the beginning of this thesis: distance-bounding proofs of knowledge. Stefano Crosta, did a marvelous job during implementation phases and provided me with any available information on XML security. I thank you all.

I would like to express my gratitude to Jacques Labetoulle who convinced me to start working as a researcher and who hired me at Eurecom Institute.

I thank my other co-authors that have not yet been mentioned: Joris Claessens, Jochen Haller, Roger Kilian Kehr, Sergio Loureiro, Joachim Posegga, Philip Robinson, Thomas Walter, and Alf Zugenmaier. It was inspiring to work with you. Thanks to numerous researchers that asked me interesting questions or gave me important feedbacks when I was presenting parts of this work. Moreover, I am grateful to authors of [MVO96, And01, Kob94] which became my preferred reference books.

I would like to thanks Alexis Bonnecaze and Marc Joye for their comments. David Powell and Frédéric Cuppens have honored me by accepting to be part of the thesis committee.

Funds for this work were provided by the Eurecom Institute and by the European Commission (WiTness project funded under Fifth Framework Program).

Last, but certainly not least, I am grateful to Cécile, who became my wife and the mother of two beautiful children while I was working on this thesis.

Laurent Bussard, October 2004

Contents

List of Figures

List of Tables

Acronyms and Abbreviations

AC	Access Control
ACL	Access Control List
AoN	All or Nothing
API	Application Programming Interface
ASN.1	Abstract Syntax Notation One
ATM	Automated Teller Machine
B2B	Business to Business
B2E	Business to Employee
CA	Certifying Authority
CDC	Connected Device Configuration (J2ME)
CLDC	Connected Limited Device Configuration (J2ME)
CRL	Certificate Revocation List
DBP[1]	Distance Bounding Protocol
DBPK[1]	Distance Bounding Proof of Knowledge
DL	Discrete Logarithm
DoS	Denial of Service
GPS	Global Positioning System
GSM	Global System for Mobile Communications
GPRS	General Packet Radio Service
GRBAC	Generalized Role-Based Access Control
IDS	Intrusion Detection System
IFF	Identification Friend or Foe
IrDA	Infrared Data Association
J2ME	Java 2 Micro Edition
JCE	Java Cryptographic Extension
JDK	Java Development Kit
JNI	Java Native Interface
JSR	Java Specification Request
JVM	Java Virtual Machine
$\mathbf{K}_P$	Public Key
$\mathbf{K}_S$	Private Key
LTS[1]	Location-and-Time Stamper
MAC	MAC address: Media Access Control address
OCSP	On-line Certificate Status Protocol

[1] New acronyms and new abbreviations that have been coined in this thesis.

OTC[1]	One-Time Credential
P2P	Peer-to-Peer
P3P	Platform for Privacy Preferences Project
PAN	Personal Area Network
PCE[1]	Pervasive Computing Environment
PDA	Personal Digital Assistant
PGP	Pretty Good Privacy
PIN	Personal Identification Number
PK	Proof of Knowledge
PKI	Public Key Infrastructure
RBAC	Role Based Access Control
RF	Radio Frequency
RFID	Radio Frequency Identification
RSA	Rivest-Shamir-Adleman public key cryptosystem
RTT	Round-Trip Time
SAML	Security Assertion Markup Language
SIM	Subscriber Identity Module
SOAP	Simple Object Access Protocol
SPK	Signature based on a Proof of Knowledge
SPKI	Simple Public Key Infrastructure
TCG	Trusted Computing Group
ToF	Time of Flight
TPH	Tamper-proof Hardware
TTP	Trusted Third Party
UMTS	Universal Mobile Telecommunication System
USIM	Universal Subscriber Identity Module
UTC	Coordinated Universal Time
WAN	Wide Area Network
WiFi	Wireless Fidelity
WLAN	Wireless Local Area Network
XACML	Extensible Access Control Markup Language
XML	Extensible Markup Language
XSLT	Extensible Style-sheet Language Transformation

Glossary

The following notations and definitions aim at reminding common notations as well as defining specific notations of this thesis. Definitions are given in alphabetical order and are based on [PHS03, CS97, MVO96, PK01a, GS00]

Notations

$\mathcal{Z}_n$	Integers modulo n: the set of integers $\{0, 1, \ldots, n-1\}$
$\mathcal{Z}_n^*$	The multiplicative group of $\mathcal{Z}_n$
G	A cyclic group with generator g
$\stackrel{?}{=}$	Check equality
$\mathcal{H}(m)$	Cryptographic hash function: $\mathcal{H} : \{0,1\}^* \rightarrow \{0,1\}^k$
$m_1 \parallel m_2$	Concatenation of strings m_1 and m_2
$\{a, b, c\}$	Set of values
$\in_R$	Randomly chosen value, e.g $r \in_R \mathcal{Z}_n^*$
$PK[...]$	Proof of knowledge
$SPK[...](m)$	Signature based on a proof of knowledge (or signature of knowledge)
K_i	Secret key (symmetric key).
$E_{K_i}(m)$	Encryption of message m using symmetric key K_i
$D_{K_i}(c)$	Decryption of ciphertext c using symmetric key K_i
K_{S_A}	Private key of A.
K_{P_A}	Public key of A.
$E_{K_{S,A}}(m)$	Using private key K_{S_A} on m (signature, decryption)
$E_{K_{P,A}}(m)$	Using public key K_{S_A} on m (verification of signature, encryption)
$\mathrm{SIGN}_\mathrm{A}(m)$	Signature by entity A on message m. E.g. $\mathrm{SIGN}_\mathrm{A}(m) = E_{K_{S,A}}(\mathcal{H}(m))$
$A \xrightarrow{m} B$	Protocol exchange: A sends message m to B.

Terminology

Anonymity: anonymity during an interaction is the state of being not identifiable among all entities that might have take part to this interaction.

Artifact: in the context of pervasive computing, artifact describes any physical object with embedded computation and communication facilities, be it a laptop, a watch, or a shoe with an embedded RFID-tag. Artifact are also referred as communicating devices.

Authentication: *entity authentication* is the process whereby one party is assured of the identity of a second party involved in a protocol.
In this thesis, *artifact authentication* is the process whereby a person or an artifact is assured of the properties (i.e. identity or attributes) of a given artifact that he can touch or see.

Bit commitment: a bit commitment protocol makes it possible to choose a bit and to reveal it later without any possibility of changing this bit in the meantime. This protocols involves two parties (a *prover* and a *verifier*) and relies on two functions ($commit, open$) that satisfies the following conditions: *Binding*: once the prover has committed herself to a bit b by presenting a *blob* to the verifier, she is later unable to change the bit; *Secrecy*: there is no leakage of information about the committed bit from the blob as long as it is not opened by the prover.

Blind signature: a blind signature is a digital signature scheme where the signer is prevented from reading the message to be signed. The requestor obfuscates (or *blinds*) the message m and sends it to the signer. The signer signs this blinded message and sends it back. The requestor is able to treat (or *unblind*) the signature in order to get a valid signature on the initial message m.

Blob: see *Bit commitment.*

Certificate: a certificate is a data structure that associates the public key of the holder to a set of attributes (e.g. identity, role, authorization, or properties). This data structure is signed by an issuer guaranteeing the validity of the attributes. The signature also ensures the integrity of the certificate.

Certification Authority: a certification authority (CA) is a trusted third party that generates and issues certificates.

Challenge-response protocol: a challenge-response protocol is a two-party protocol during which a prover proves the knowledge of a password, a secret key, or a private key to a verifier. In the context of this thesis, we use challenge-response protocols to interactively prove the knowledge of a private key.

Context: the context is the logical state of a party as well as its surrounding physical environment. In this thesis, we focus on provable information on the physical context:

mainly the time, the location, or the proximity of another party.

Credential: A credential is an assertion about a specific party (credential holder) that is digitally signed by a trusted authority (credential issuer). A certificate is a particular instance of credential.

Cut-and-choose protocol: a cut-and-choose protocol is a two-party protocol. One party sends some data and tries to convince the other party that this data was constructed according to a method they agreed upon.

Electronic cash: e-cash aims at defining a new type of money based on digital information. Since an e-coin is just a bunch of bits, it is easy to duplicate. The copy being indistinguishable from the original, a major challenge is to define mechanisms that avoid double-spending. Another important aspect of e-cash is to respect the anonymity offered by classical money.

Group signature: A group signature allows any member of a group to digitally sign a document in a manner such that a verifier can confirm that it came from the group, but does not know which individual in the group signed the document. The protocol allows for the identity of the signer to be discovered, in case of problem, by a designated group manager. Four protocols are generally defined: *Join* is a protocol between the group manager and a user that results in the user becoming a new group member; *Sign* is a protocol between a group member and a user whereby a group signature on a user supplied message is computed by the group member; *Verify* is an algorithm for establishing the validity of a group signature given a group public key and a signed message; *Open* is an algorithm that, given a signed message and a group secret key, determines the identity of the signer.

History: the provable history of a party is the set of credentials that this party received when interacting with other parties. The history can contain proofs of location, recommendations, hierarchical relationships, etc.

Nonce: a nonce is a random challenge (see *challenge-response protocol*).

Non-transferability: A credential is said non-transferable when the holder has to provide a valuable secret to enable another party to use this credential. For instance a certificate cannot be transferred because it implies providing ones private key as well.

Off-line: Off-line (or disconnected) interactions are interactions that involve two or more local parties but that cannot rely on a remote trusted third party.

One-time pad the one-time pad (OTP) is a theoretically unbreakable method of encryption where the plaintext is combined with a random pad that is only used once. The pad has to be the same length as the plaintext to encrypt and thus the major problem is the secure distribution of pads.

Penalty: penalty is the punishment for violating rules or for abusing the system. Penalties are usually monetary fines. In this thesis, penalties enable some additional control on misbehaving users when access control is not sufficient.

Personal area network: a personal area network (PAN) is a network for communication among artifacts surrounding one person. The artifacts may or may not belong to the person in question. The range of a PAN is typically a few meters. A Bluetooth PAN is called a *piconet*.

Pervasive computing: pervasive computing or ubiquitous computing describes the concept of integrating computation into all surrounding objects (artifacts). Embedding computation into the environment would enable people to move around and interact with computers more naturally than they currently do. Another goal of pervasive computing is to enable devices to sense changes in their environment and to automatically adapt and act based on these changes and on user needs and preferences.

Privacy: in this thesis, privacy is defined as the right of an individual to be secure from unauthorized disclosure of information about oneself. Different services are associated with privacy: anonymity, pseudonymity, unlinkability, and unobservability.

Proof of knowledge: a proof of knowledge (PK) is a two-party protocol by which the prover P proves that he knows some secret, e.g. the discrete logarithm of some y to the base g, without revealing this secret. A proof is defined as *zero-knowledge* when it is possible to prove that no information about the secret is revealed.

Pseudonymity: pseudonymity is the use of pseudonyms as identifiers. A digital pseudonym is a unique identifier that can be used to authenticate the holder but that does not reveal his identity.

Selective disclosure: selective disclosure is the technique that enables the holder of some information to choose the granularity of revealed information. For instance the holder of a digital identity card should be able to prove his nationality without revealing any information on his/her name or age.

Signature based on a proof of knowledge: a signature based on a proof of knowledge (SPK) or signature of knowledge (SK) is a non-interactive version of the proof of knowledge. It enables a signer to sign a message as *"someone knowing some secret"* without revealing this secret. The signature can be verified by any party.

String commitment: a string commitment scheme is a generalization of a *bit commitment* scheme. Unlike in a bit commitment, the prover can hide a string of bits in a single blob.

Timestamp: A digital time stamp is an unequivocal proof that a piece of data existed at a point-in-time and that the contents have not changed since that time. A timestamp is a digital notary that is generally used for witnessing intellectual property.

Trust: trust is the firm belief that an entity is competent and willing to act securely and

reliably in a given situation. Trust is established based on knowledge, beliefs, or claims about another party.
In this thesis, we focus on *resource access trust* where a party trusts another party and gives it access to its resources and *service provision trust* where a party trusts another party to provide a service.

Trusted Third Party: a trusted third party is an authority that is trusted by other parties with respect to specific security-related activities (e.g., authentication, timestamps, privacy, or authorizations).

Unlinkability: unlinkability of two or more items (e.g., subjects, messages, events, actions, etc.) means that, these items are no more and no less related than they are related concerning the *a priori* knowledge.

Untraceability: in this thesis we assume that *untraceability* and *unlinkability* are synonym.

Valuable secret: a valuable secret is a secret that cannot be disclosed because it would enable identity theft. For instance, in a public key infrastructure, private keys are valuable secrets.

Introduction

"It was a device by means of which everyone could be surrounded night and day by informers who knew him intimately."

– **George Orwell** (Nineteen eigthy-four)

Thirteen years ago, Mark Weiser proposed the *pervasive computing* paradigm [Wei91]: a world in which computing power and communication facilities could be embedded in any object. In his vision, users would transparently interact with objects without being aware that they indeed interact with computers. Today, pervasive computing is an important research field involving hundreds of researchers worldwide. The outcome of this research already started to impact real life: personal digital assistants (PDAs), smart-phones and other highly communicating devices are ubiquitous; radio frequency identification tags (RFID-tags) will soon become the widest market for chip manufacturers; and smart appliances are more and more common. This evolution entails unforeseen threats and vulnerabilities, and thus protecting users in terms of security and privacy is becoming a major concern. Therefore assuring security and privacy in pervasive computing is at issue in this dissertation.

New Paradigms

Pervasive computing literally denotes a situation in which computers are present and felt everywhere. This, however, may mean many things, and indeed authors have used the term in significantly different ways. Computation and communication facilities can be spread in many manners. Large intelligent environments, be it an office, a building, or a town, are envisioned to offer services to users [CLS$^+$01]. In this case, an infrastructure is deployed to observe users with location sensors, smart floors, or cameras and to provide services such as printing, opening doors, or guiding visitors within a campus. Without infrastructure, intelligent appliances like vending machines or printers are expected to offer wireless interfaces in order to provide (or request) services [KZ03]. Federations of appliances and devices such as PDAs are emerging, and thus pairing mechanisms, which enable the establishment of a secure link between two devices, become important

[HMS+01, SA99]. A more futuristic trend is to embed chips in any object: from clothes and books (RFID tags [Bri03]) to human beings (wearable computers like e-rings [Cur98] or smart watches [NKR+02]). An even wider deployment could be achieved with very small communicating devices like smart dust [KKP99] spread in environments. However, in this thesis we will only focus on communicating devices that are powerful enough for supporting asymmetric cryptography. In other words, we will mainly look at smart environments offering services to a user carrying a trusted device, for example a personal digital assistant, a cell phone, or a smart watch.

Pervasive computing brings together several different research topics: intuitive human machine interactions [Bor00], context awareness (sensing and monitoring) [GBEE02, KH00], software architecture, self-organized networking [MM02], energy management, security, etc. This dissertation specifically addresses security and privacy in pervasive computing.

Research on pervasive computing being quite recent, different designations are actually used: *ubiquitous computing*, *disappearing computing*, and *communicating devices* are synonymous with pervasive computing. In this dissertation, we focus on application level and thus neither address *ad hoc networks*, which mainly focus on routing in autonomous and self-organized networks, nor *sensor networks*, which define networks of devices with very restricted resources.

Today some basic concepts of pervasive computing are becoming a reality for the general public: cell-phones or even cars' computers propose multiple services like location or hotel reservation; more and more sensors are spread in public and private environments; and electronic cash, electronic vouchers, and electronic travel tickets are becoming common. Furthermore, pervasive computing is now a mature research topic and numerous research institutes work on more advanced concepts. At least three international conferences cover it: *conference on ubiquitous computing* (Ubicomp) since 1999, *conference on pervasive computing* (Pervasive) since 2002, and *IEEE conference on pervasive computing and communication* (PerCom) since 2003. Two journals also focus on this topic: *personal and ubiquitous computing* edited by ACM and Springer since 1997 and *IEEE pervasive computing (mobile and ubiquitous systems)* since 2002. Workshops on security and/or privacy in pervasive computing are regularly organized in these conferences and in security conferences as well. In 2003 the *first conference on security in pervasive computing* (SPC) was launched and the first book on this topic [Sta02] was printed in 2002.

New Requirements for Security and Privacy

The large scale deployment of pervasive computing heavily depends on the assurance of essential security properties for users and service providers. Indeed, pervasive computing impacts daily interactions that often involve assets of users and service providers.

For instance, walking through a gate controlling the access to an office may require an authorization or getting on a bus may imply an automatic micro-payment.

In addition to security exposures due to the underlying mobile and wireless communications, pervasive computing brings up new security issues. In pervasive computing, users should not be aware of the presence of computers. Thus interactions between users and artifacts (i.e. objects with embedded computation and communication facilities) have to be intuitive and non-disturbing. In terms of security it means that users cannot be expected to enter a password before interacting with an artifact. Moreover, the association between a physical entity and a virtual service is specific to pervasive computing that thus bridges the gap between physical and virtual worlds [Sat01, IU97, McC01]. This has a strong impact on the meaning of authentication. Indeed, numerous security protocols exist for verifying that a virtual service (e.g. e-banking) is certified by a trusted authority but protocols to check whether a given artifact (e.g. the printer in front of the verifier) is certified are uncommon. This relation with the real world also promotes the use of contextual information. The context can even be used for authentication [MRCM02], access control (only users that are present in a room can turn on the light of this room) [CMA00], or privacy protection (displayed information depends on who can see the screen) [RNP03].

Another particular characteristic of pervasive computing is that it lacks infrastructure in terms of communication and trust. First, communication among artifacts often relies on short-range channels (e.g. Bluetooth) defining personal area networks (PAN) and thus it is realistic to assume that there is no permanent global connectivity [CGS$^+$03]. This is referred to as *disconnected mode* in the remainder of this thesis and means that during some interactions it is impossible to rely on remote trusted third parties such as certification authorities or revocation lists. Moreover, pervasive computing foresees billions of devices leading to frequent interactions among unknown parties. Thus there is potentially a lack of *a priory* trust among parties [BSSW02, CGRZ03] and mechanisms to build trust have to be deployed.

Finally, privacy is also a major concern. Indeed, initial applications of pervasive computing that are already deployed yet reduce the privacy of users: nowadays private data can be automatically logged in order to make a profile of a person. For instance, shopping malls or credit card companies can keep track of users' purchase, Internet service providers can observe Web pages accessed by their clients. The gathering and combination of this data is an important threat against privacy that can lead to targeted advertisement or even weaken political systems [Ger04]. Moreover, real-time location of people can also be achieved thanks to their credit card payments, cell-phones, auto-tolls, or car's security system. The future development of pervasive computing will affect more and more daily interactions and thus if no privacy protection is provided, it will be possible to log any interaction. Indeed, RFID tags, proximity sensors, micro payments, or automatic authentication could feed huge databases logging daily interactions of users and enable to accurately profile users leading to violation of their privacy.

Problem Statement

In this dissertation, we present different solutions to deal with the following scenario: Alice (A) visits an environment managed by Bob (B). We assume a lack of *a priori* trust among A and B. In other words, there is no global trust infrastructure such as a public key infrastructure (PKI) defining a relationship between A and B. Moreover, we observe a lack of communication infrastructure so that a part of the operation has to be done in disconnected mode without relying on centralized authorities. More details on each property are provided in the remainder of this section.

Figure 1 describes the problem we tackle in this dissertation: how to provide rights to parties when there is no *a priori* trust, when permanent connection to trusted third party (TTP) cannot be assured, and when privacy and context are important concerns.

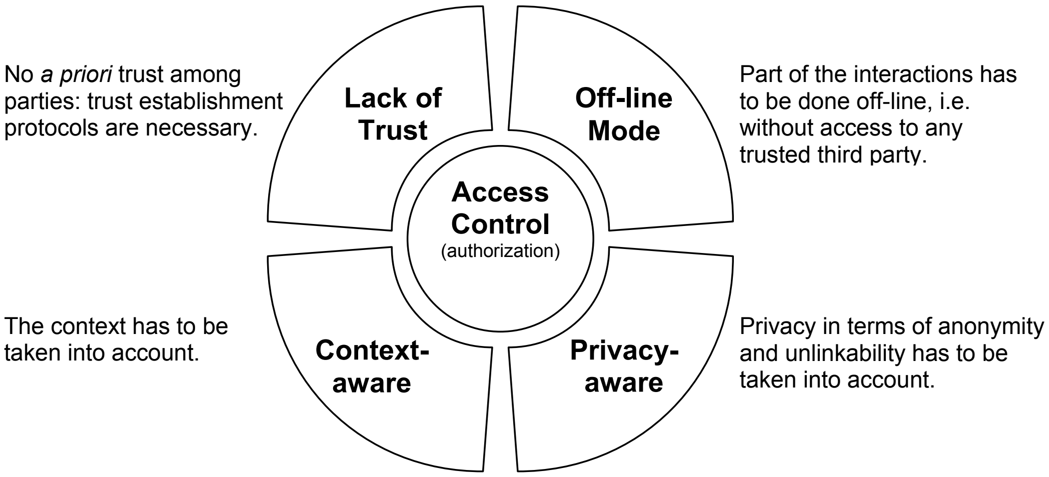

No *a priori* trust among parties: trust establishment protocols are necessary.
Lack of Trust
Off-line Mode
Part of the interactions has to be done off-line, i.e. without access to any trusted third party.
Access Control
(authorization)
The context has to be taken into account.
Context-aware
Privacy-aware
Privacy in terms of anonymity and unlinkability has to be taken into account.

Figure 1: Application-level security in pervasive computing

Lack of Trust Infrastructure

Trust is generally defined as *the belief that one can depend on something or someone.* In this definition, there are clearly notions of uncertainty and risk. A resulting property is that *if party B trusts A then B can grant A some rights.* Likewise, any information that enables B to decide whether he can grant A some rights is part of the trust establishment process.

Trust is usually derived from some existing relationship known as *a priori* trust. For instance, C trusts B and then C trusts A if B says that A is trustworthy. In a web of trust, B is a friend of C, and in the simple public key infrastructure, B is the hierarchical authority of A.

However, in any large environment involving unknown parties, be it peer-to-peer data sharing, pervasive computing, or ad hoc networks, there is a lack of *a priori* trust among parties and thus a new mechanism is required to build trust in an *a posteriori* fashion based on monitored evidence. The following two alternatives appear to be suitable concepts on which to build *a posteriori* trust:

- Reputation (statistical evidence).
- History of interactions (provable evidence).

In this thesis we focus on the latter case. A interacts with B and receives a credential proving that this interaction occurred. Subsequently, A can prove to B or to another party C that this interaction occurred in the past. As a result of this proof, B or C can grant A some rights. History may contain trust-related evidence like recommendation as well as more general evidence like having been at a given location or being a member of a group.

Disconnectivity: Lack of Communication Infrastructure

Defining a security infrastructure supporting disconnected interaction may be questionable since workstations and servers are permanently on-line and even in mobile context, permanent connectivity can be achieved by using local communication infrastructure (e.g. WiFi access points) and/or global cellular networks (e.g. GSM, UMTS). However, an exclusively on-line approach is sometimes simply not affordable for building and securing systems. Long distance communication might simply be impossible because of obstacles (buildings, tunnel, etc.) or because the infrastructure might have been destroyed in a disaster such as an earthquake, even though local appliances might be in reach. Server crash, network problems, or denial of service attacks can also temporarily forbid connection or make it too difficult to use because of an excessive response time. Complexity for deploying a fully connected environment might also be overwhelming. Finally, the cost for permanent mobile communications is still prohibitive.

Thus, despite the ubiquitous and cheap availability of communication channels, disconnected situations are likely to occur.

Privacy Aware Protocols

Users need a way to prove their history of interactions to another party. However, a proof must not be linkable to a previous event (*untraceability*) and the identity of the user must remain secret (*anonymity*). Moreover, when a credential can be used more than once, it is necessary that different proofs are not linkable (*unlinkability*).

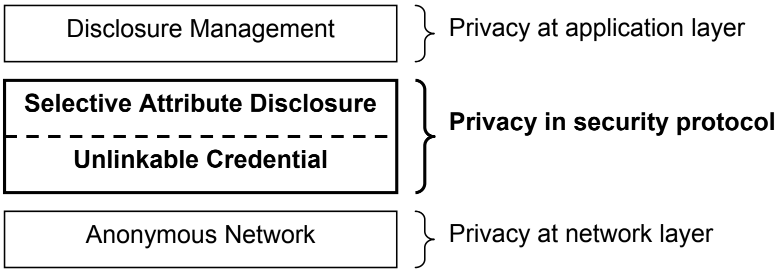

Disclosure Management
Privacy at application layer
Selective Attribute Disclosure
Unlinkable Credential
Privacy in security protocol
Anonymous Network
Privacy at network layer

Figure 2: Three layers for ensuring user's privacy

Our architecture for protecting the untraceability of users in such a context relies on three layers (see Figure 2). First, at the bottom of the architecture, privacy at network level is required in order to prevent tracing of user behavior based on the monitoring of network traffic. In a personal area network (e.g. Bluetooth), MAC addresses should not be visible or should change regularly. And in the Internet, where MAC addresses are not visible, IP addresses should be kept secret by using, for instance, onion routing or mixes [Cha81]. The second layer assures that the credentials delivered to the users cannot be traced. Indeed, private user data (e.g. identity) can be directly exposed through security protocols since most classical security mechanisms are based on identification and authentication. Schemes relying on blind signatures or on signatures of knowledge can solve this problem. Third, at the application layer, the attributes that are revealed have to be carefully chosen in order to avoid traceability based on the attributes. It is obvious that identity certificates enable traceability of holders even when the credential is unlinkable. However, less precise attributes like the birthday or the office number can also be used to trace some person in a small group. Statistical disclosure control [WdW00] aims at controlling what information can be revealed without threatening user's privacy. In this thesis we only focus on the second layer and propose new schemes for unlinkable credentials dedicated to trust establishment.

Context Aware Protocols

In pervasive computing the context refers to any information on the local environment: time, location, brightness, temperature, presence of another entity, etc. On one hand, a solution for monitoring the context and subsequently using this measure as a proof would be useful. On the other hand, only results signed by a trusted sensor could be seen as proof, and, even in this case, it is easy to tamper with context: GPS signal can be spoofed, temperature sensors can be warmed up, and so on.

To enable proofs of context, we defined a protocol that lets a verifier measure the maximum distance to a prover that knows a secret. We state that time and distance is the only contextual information that cannot be manipulated by an attacker. Indeed, time can be taken into account by security protocols and the bounded speed of light makes it possible to associate maximum distance to maximum response time. In this dissertation we will show that distance-bounding protocols are important building blocks for defining security protocols in pervasive computing. Distance-bounding protocols can be used in order to authenticate artifacts: a user verifies that the printer in front of him is indeed certified by a trusted party. These protocols also enable secure device pairing in order to establish a secure channel between two artifacts. Finally, they can be used with a trusted third party that delivers a proof of context. For instance, an appliance verifies the proximity of a party knowing a private key and next delivers a proof of location associated to the corresponding public key. This proof can then be shown to another party to assert the location of the user at a given time.

Access Control in Pervasive Computing

Defining an access control model taking into account the lack of trust and communication infrastructure, and the requirements for privacy and context awareness is a challenging task. To decide whether an entity can be authorized to access a service, it is necessary to rely on trust relationships. When no trust is defined, this means establishing trust based on observation, recommendation, or reputation mechanisms. The fact that the communication media is not reliable disables access control based on trusted third parties and promotes credentials. Privacy, in terms of anonymity of the credential holder and unlinkability of the credential, supports unlinkable credential schemes. Last, to deal with contextual information means to be able to prove the context of a party, e.g. its location at a given time.

Some related approaches support part of the mentioned constraints but no approach deals with all constraints. In this thesis, we will describe different extensions of existing schemes, namely electronic cash, group signature, proof of knowledge, and attribute certificate, to enable trust establishment and access control in disconnected environments requiring privacy and context awareness.

Organization of this Thesis

Each chapter proposes a building block for tackling trust relationships in pervasive computing environments when part of the interactions are done in disconnected mode. For the sake of readability, the state of the art is not centralized but distributed in suitable chapters. Figure 3 summarizes the structure of this dissertation. We start with the most constraining hypothesis: privacy of users is required, no trust infrastructure is available, and it is impossible to build any trust on a history of interactions. This shows that, even without trust relationship, it is possible to define access control enabling clients to use services while protecting service providers. Next, we describe how a history of interactions can be used to establish trust among parties. Because no trust infrastructure leads to strong limitations, we assume that a minimal trust infrastructure exists and assures that each user has a valuable secret, i.e. something equivalent to a private key. History-based trust establishment is defined in this context. Finally, we focus on implementation results related to trust establishment in pervasive computing.

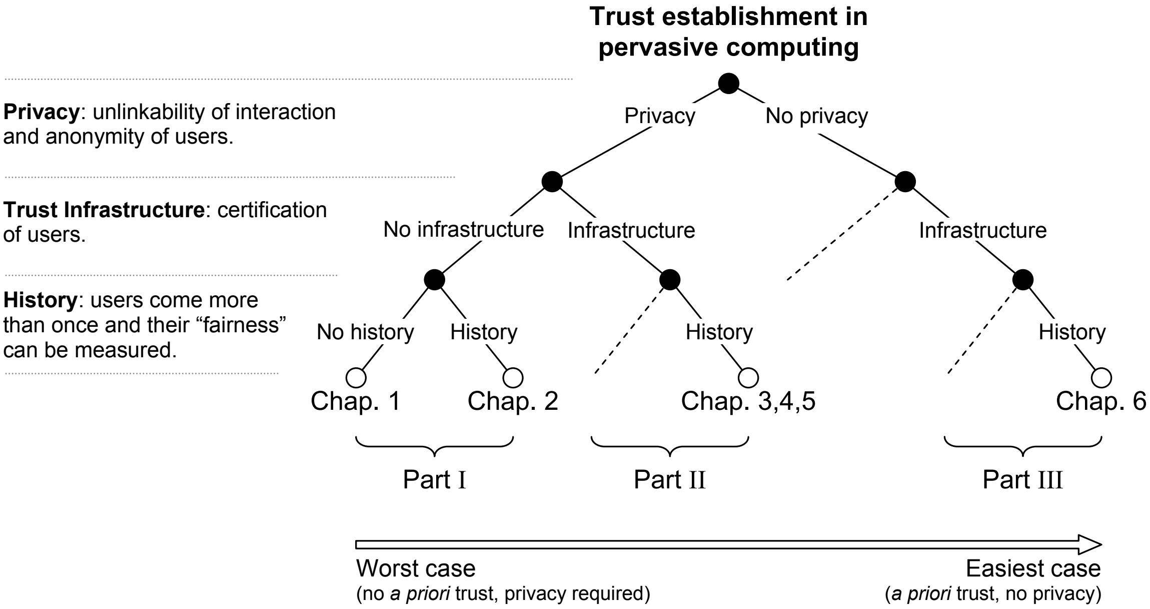

Trust establishment in pervasive computing
Privacy: unlinkability of interaction and anonymity of users.
Trust Infrastructure: certification of users.
History: users come more than once and their "fairness" can be measured.
Privacy
No privacy
No infrastructure
Infrastructure
Infrastructure
No history
History
History
History
Chap. 1
Chap. 2
Chap. 3,4,5
Chap. 6
Part I
Part II
Part III
Worst case
(no a priori trust, privacy required)
Easiest case
(a priori trust, no privacy)

Figure 3: Organization of this thesis

Part I

Part I focuses on environments without authentication mechanisms, i.e. without certification of entities. Preliminary sections of Chapters 1 and 2 introduce few techniques and mechanisms that are developed for this thesis.

First, Chapter 1 studies whether it is possible to provide rights to users when there is no *a priori* trust and no way to build trust. In this chapter we propose a new approach using one-time credentials that can be used off-line, i.e. only relying on local communications. This solution assumes that neither the service provider B trusts the user A nor A trusts B. In this scheme money, i.e. electronic check, is envisioned as a universal penalty mechanism.

Having shown the limitations due to the lack of trust, Chapter 2 proposes an original way to build trust. It is also assumed that there is no *a priori* relationships among client A and service provider B. However, in this case A will request multiple times services of B that can evaluate whether the behavior of A during an interaction was correct. After an interaction B provides a credential to A depending on the result of the interaction. During a subsequent interaction with B or partner of B, A can use this credential to assert some previous relationship. This is an embryo of *history-based trust establishment* that is developed in the second part of this dissertation. Chapter 2 presents a protocol that ensures unlinkability of interaction by defining unlinkable credentials. Attribute values are encrypted to enable positive as well as negative statements.

Part II

Part II is the core of this thesis. It focuses on trust establishment based on evidence like proof of context (e.g. location and time) or proofs of interactions (e.g. recommendations).

The main limitation of the protocol proposed in Chapter 2 is the small cardinality of the set of possible attribute values that makes it unsuitable for defining complex attributes such as time or rights. Chapter 3 describes the first building block of history-based trust establishment: an unlinkable signature scheme. It relies on credentials with cleartext attributes that can be disclosed in a controlled way.

Chapter 4 shows that contextual information (especially time and location) is an important aspect of security in pervasive computing and thus impacts the trust establishment. Some approaches to prove the location of a party are compared and *distance-bounding proofs of knowledge* are introduced. This extension of distance-bounding protocols enables unforgeable proofs of proximity.

Chapter 5 describes the history-based trust establishment protocol that combines unlinkable signatures (Chapter 3) and distance-bounding proofs of knowledge (Chapter 4).

Part III

Part III presents our implementation of trust establishment in pervasive computing. This work was part of a wider project on the security of mobile and pervasive business applications.

Chapter 6 explains why nowadays mobile environments cannot afford privacy. Without taking privacy into account, we describe how trust can be established in a very specific context: a federation of devices and users from different trust domains collaborate in *business-to-employee* and *business-to-business* applications. This chapter describes our contribution to the European project *Wireless Trust for Mobile Business (WiTness).*

Appendices

Each appendix gives more details on implementation or on possible extensions related to a given chapter. Readers can skip the appendices and we do not recommend examining an appendix without having read the related chapter.

Contributions of this Thesis

Privacy and security in pervasive computing are not mature enough and thus this dissertation does not aim at defining finalized security protocols but investigates how new problems could be solved.

Trust Establishment with Privacy

The main contribution of this thesis is to demonstrate that it is possible to build trust relationships while preserving privacy. A framework for trust establishment based on history is defined and a set of security protocols are proposed with different assumptions on the existing infrastructure. Another contribution of this thesis is the implementation of a simple trust establishment protocol in the *WiTness project* (Wireless Trust for Mobile Business). Moreover, the remainder of this section shows that part of the results presented in this dissertation may be used in another context.

Unlinkable Credentials

Unlinkable credentials and trust establishment protocols described in this dissertation have been designed to address specific security problems related to pervasive computing. However, not only future systems but also various emerging mobile computing applications could benefit from this approach. It could also help collaborations over the Internet where correspondents' identities and intentions are difficult to establish.

Proof of Proximity

Distance-bounding proof of knowledge is a generic mechanism to prove the proximity of an entity knowing a secret. This protocol extends the initial distance-bounding protocols [BC93] in order to disable a type of attack referred as "*terrorist fraud*", which was let as an open issue. In this dissertation, this mechanism is used for defining proofs of location that are necessary during trust establishment. However, this mechanism also enables proofs of physical interactions, certification of artifacts, authentication of artifacts, etc. We think that a long-term contribution could be the deployment of distance-bounding mechanisms in any device that needs to verify certified attributes of another device, be it a smart card, a cell-phone, or a trusted computing platform.

Part I

Trust without Infrastructure

Chapter 1

Authorization without Trust

”If, indeed, you want praise, esteem, kindness, and friendship, you are welcome to any amount; but money, that’s a different affair.”

– **Molière** (The Miser, II:V)

This chapter studies whether it is possible to provide rights to users when there is no *a-priori* trust and no way to build trust. We propose a new approach using one-time credentials that can be used in disconnected mode, only relying on local communications during access control. Money, i.e. electronic check, is envisioned as a universal penalty mechanism. This solution assumes that neither the service provider B trusts the user A nor A trusts B.

1.1 Introduction

In this chapter we focus on the worst trust model where neither the client Alice (A) trusts the service provider Bob (B) nor B trusts A and where there is no assurance that A will interact with B more than once. In other words, there is no trust among parties and no way to build some trust based on a history of previous interactions. New solutions addressing these issues are required both for the protection of users, including privacy measures, and for controlling the access to valuable resources like commercial services. We define a new access control scheme suited to pervasive computing with the following requirements:

- The service provider B does not trust the client A: misbehavior of the client leads to direct penalty.

- Lack of permanent communication channels: verification of access rights is performed in a disconnected mode, i.e. without relying on a trusted third party (TTP).

- A does not trust B: privacy concerns are taken into account in order to assure the anonymity of clients and untraceability of transactions.

In this scheme, some rights are granted to any visitor of a pervasive computing environment so that visitors can use local facilities. Before granting rights, authentication may be used in order to define rights according to some role that is checked at a front desk before entering the place. For instance, any visitor in a conference center could be authorized to print few pages or any patient of a physician could get one coffee while waiting.

Controlling access rights is necessary: the suggested access control scheme allows a user to get authorized access to a service based on the one-time credential concept. One-time credentials are provided to let users print n pages, get a coffee, or use a meeting room during one hour. Misbehavior is defined as the attempt to access services beyond authorization, i.e. attempts for multiple uses of one-time credentials. The one-time property and the resulting double use prevention rely on a penalty mechanism whereby a cheating user looses some money she deposited as a guarantee of her loyalty prior to a set of service accesses. The one-time property is enforced by a mechanism based on money.

The verification of the user's credential can be performed without any communication with a third party system, since the validity of each one-time credential can be locally checked by each service provider. Mechanisms to avoid double-use of rights in an off-line context have been largely debated in the context of electronic cash and the work described in this chapter is an extension of existing e-cash schemes. We propose that the double-use of a credential makes the misbehaving user loose some money deposited earlier. Using this access control scheme, the user and the service provider do not need to be part of the same organization or to trust one another.

Protecting the user's privacy is necessary: depending on the application, the user may be authenticated when getting credentials. However, an honest user should not be traceable based on her interactions with various servers. In this chapter we propose untraceable one-time credentials that allow the service provider to cash an electronic check deposited by the client in case of misbehavior by the latter.

We first give a precise description of the target application scenario with respect to the access control problem and analyze the limitations of existing access control solutions in the light of this scenario. Next we describe some mechanisms that will be extended in the remainder of this chapter, namely blind signatures and electronic cash. Our solution based on the one-time credential concept is then introduced first with a high level sketch of the idea. A detailed description is given in terms of a protocol in three phases. The security of the protocol is discussed in the last sections of the chapter.

1.2 Problem Statement

The application scenario envisioned in this chapter consists of several pervasive computing environments (PCE) as shown in Figure 1.1. Each PCE includes a set of appliance servers $(C_1, C_2 \cdots, C_z)$ whose access is controlled by B, the authority of the PCE. A dynamic user population called visitors or clients $(A_1, , A_2, \cdots, A_v)$ randomly visits PCEs and requests services from appliance servers. The authority of each PCE is in charge of providing each visitor with rights to access servers. Due to the possibly large coverage of each PCE and the limited transmission capabilities of pervasive servers, no on-line connectivity between the servers and the authority is required. However, servers periodically exchange some data with the authority. For instance, once a day, vending machines provide data to the authority via the support personnel. Since servers cannot communicate with the authority in a timely and interactive way, we qualify the interactions between the clients and the servers of a PCE as *off-line*. Each client on the other hand can establish interactive exchanges with the server he/she is in touch with. Another characteristic of this environment is the lack of a priori trust between servers and visitors. Servers do not trust visitors and possibly are not even able to identify them. Within a PCE, servers trust the authority with respect to the authorization scheme in that each access requests bearing a valid authorization proof delivered by the authority are granted access by the server to which they are destined. The policy according to which the authority grants access rights is out of the scope of this dissertation. Moreover the authority is only in charge of enforcing access control within a PCE and the fact that there is a single authority in each PCE does not necessarily imply that all servers of the PCE belong to the same administrative domain. Furthermore, multiple servers might offer the same type of service like printing, vending food and beverages, or opening doors.

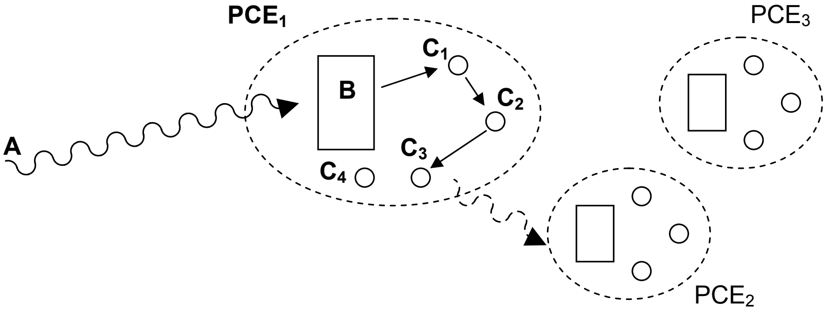


Figure 1.1: Access Control in pervasive computing application scenario

The access control in the pervasive computing application scenario can be illustrated by an example as follows.

1.2.1 Example

A visitor (Alice, A) arrives at the gate of a shopping mall (PCE). Once she is through the registration she passes near a wireless sensor acting on behalf of the authority B. The sensor loads the visitor's personal device (e.g. cell-phone, PDA, or smart card) with a set of rights based on the visitor's attributes (e.g. role, identity, or location) and on the types of services the visitor has subscribed to during the registration. Using the rights she thus obtained, the visitor can access various services like vending machines. The access control is operated by the shopping mall but the services to which access is granted by means of the authorization scheme can be managed by independent service providers. The main security goal in this context is to prevent the visitor from unauthorized access to services even when the services are provided by off-line devices. For instance, if the visitor is authorized to get one coffee and tries to use his right with two different coffee machines that cannot communicate with one another or with the authority, the access control mechanism should detect and prevent the duplicate access attempt.

1.2.2 State of the Art: Access Control and Penalty

In this dissertation, we have chosen to distribute the state of the art within different chapters according to the topic of those chapters. Preliminary sections present more specifically mechanisms and protocols that are used or modified in the dissertation.

A straightforward solution to deal with access control consists of access control lists (ACL) that would be supported by the servers of the pervasive computing scenario. Whereas ACLs satisfy off-line requirement by allowing each server to be able to locally take the access control decision pertaining to its resources, because ACL relies on authentication this approach cannot be used in pervasive computing where the identity of visitors cannot be known in advance and authentication thus is impossible. Indeed, not only visitors and servers do not belong to the same security domain but there is not even a common naming convention for all visitors and servers on which to build an authentication mechanism. The simple public key infrastructure (SPKI/SDSI) [EFL$^+$99] proposes authorization certificates to deal with unknown entities. The drawback of this approach is the complexity of revocation. It is indeed extremely difficult to distribute and update revocation lists and the only realistic solution for revocation of authorizations during off-line interactions is thus based on short-term certificates that would require the visitors to frequently communicate with the authority B to get new certificates and that would not allow detection of duplicate access with several off-line servers. Addressing the privacy requirement [Bra02, Bra00] presents digital credentials with selective disclosure. Idemix [CH02, CL01] offers an interesting approach to create non-transferable anonymous multi-show credentials assuring unlinkability: each user has different pseudonyms that cannot be correlated across various access attempts. In this scheme, credentials cannot be transferred because the transfer thereof would require sharing a pseudonym with another entity. Even though perfectly suitable for addressing privacy concerns like unlinkability and non-transferability, Idemix would not meet off-line revocation requirements of the ubiquitous application scenario. Trusted environments like the platform proposed by the Trusted Computing Group (TCG) or smartcards could be viewed as a viable alternative for building a trusted reference monitor in each server but the access control mechanism implemented by the trusted environments would still suffer from the lack or limitation of *a priori* trust in our scenario. Chaum's electronic cash [CFN89] offers a one-time credential that suits the off-line nature of the servers in the ubiquitous application scenario. Moreover, in this scheme, unlinkability is assured by blind signatures [CR82]. When an electronic coin is used twice, it is possible to retrieve the identity of the cheater and the bank that issued the coin can act against this cheater. An access control scheme suitable for the ubiquitous application scenario could be envisioned based on an extension of electronic cash whereby the amount in the electronic cash is replaced by the encoding of some rights. A strong requirement of electronic cash is the existence of a widespread banking organization that issues electronic coins to users and performs compensation for merchants. The main deterrent to double spending in this scheme is thus based on the disclosure of cheaters' identity by the banking organization that can debit the bank account of the cheater. This requirement is relaxed in the pervasive application scenario whereby assuming a shared organization within a PCE or across several PCEs is not realistic.

Access Control and Penalty in Pervasive Computing

Pervasive computing environments limit the use of existing access control mechanisms: part of the interactions are off-line, i.e. when a visitor asks for a service, the server cannot rely on any centralized authority to verify whether the request is authorized; there is no *a priori* organizational relationship like a hierarchical structure between visitors and environments, it thus is difficult to prevent cheating thanks to hierarchical pressure; and, last, privacy of visitors in terms of anonymity and untraceability is a major concern in such an environment.

Off-line Access Control: because certificate revocation lists cannot be used in off-line context, validity of multi-show credentials has to rely on time, i.e. validity end. Unfortunately, it is often not sufficient to give access to some service during a period of time. Service providers often want to restrict accesses more precisely by limiting the number of successful attempts to use resources by a user. For instance, an unknown visitor may be authorized to print a limited number of pages or may receive only one special discount when visiting a given area. N-times credentials seem appropriate as a more refined technique assuring fine-grained control of access attempts. In this chapter, we thus propose a new *one-time credentials* scheme.

One-time authorization: there are two ways to avoid double usage of one-time authorizations during off-line interactions: on one hand, it is possible to rely on a neutral device that controls the credential, i.e. use it once and delete it. Such architecture could be based on TCG or smart cards but the wide deployment of neutral hardware is difficult to achieve. On the other hand, it is possible to deliver the service, postpone the verification to the next on-line interaction, and punish cheaters. This approach is used to avoid double spending of electronic cash [CFN89]. However, a *hierarchical relationship* between cheaters and service providers is required to enable punishment. For instance, a client can get some electronic cash from his bank or an employee could receive some credential from his employer. In both case, bank and employer have a way to press cheaters. In pervasive computing environments, this relationship does not exist.

Privacy: last but not least, privacy concerns are very important and it is necessary to ensure anonymity as well as untraceability of users.

1.3 Preliminaries

The work described in this chapter is an extension of earlier electronic cash schemes whereby electronic coins are used as untraceable credentials for authorization. This section only describes blind signatures and electronic cash schemes that will be used subsequently.

1.3.1 Blind Signature

First proposed by David Chaum [CR82], a blind signature scheme consists of a two-party protocol that allows a party to get a message signed by another party without revealing any information about the message to the signer.

The main goal of a blind signature is to assure that once the signature is issued, the signer cannot trace back the signed message to the requestor of the signature.

More formally, a *requester* A sends a piece of information m' to a *signer* B. B signs m' and returns it to A. From this signature, A can compute B's signature on an *a priori* message m of A's choice. After the protocol, B knows neither the message m nor the value of the signature on this message $s = \text{SIGN}_B(m)$.

Chaum's Protocol

The RSA public and private keys of the signer (B) are (n, e) and (d, p, q), respectively. r is a random secret integer chosen by A satisfying $1 \leq r \leq n-1$ and $\gcd(n, r) = 1$ where $n = pq$.

Requester A receives a signature (s') of B on a blinded message (m'). From this, A

computes B's signature (s) on the message (m) chosen a priori by A, $0 \leq m \leq n-1$. B has no knowledge of message and signature (respectively m and s).

Details of the protocol are given in Table 1.1.

A		**B**
chooses m and k		
computes $m' = m \cdot r^e \mod n$		
	$\xrightarrow{m'}$	
		computes $s' = (m')^d \mod n$
	$\xleftarrow{s'}$	
computes $s = r^{-1} \cdot s' \mod n$		

Table 1.1: Chaum's blind signature scheme

The blind signature works because A can get the signature s from the blind signature s' as follows:

$$s = r^{-1} \cdot s' = r^{-1} \cdot (m')^d = r^{-1} \cdot (m \cdot r^e)^d = r^{-1} \cdot m^d \cdot r = m^d \mod n$$

Now s is a signature on A's message m that cannot have been generated without the help of B. This signature scheme is secure provided that factorization and root extraction remain difficult. However, regardless of the status of these problems the signature scheme is unconditionally *blind* since r is randomly chosen in the set $\{1, \ldots, n-1\}$. In this dissertation, we will use $\in_R$ to describe the random selection of an element of a set. The random r does not allow the signer to learn about the message even if the signer can solve the underlying hard problems.

Blind signatures have numerous uses including time-stamping, anonymous access control, and digital cash. It is thus not surprising that there are numerous blind signature schemes. The first scheme proposed by Chaum and summarized in this section is based on RSA. However, work on blinding different signature schemes has been carried out. For instance, [Ram99] defines a blind version of the Schnorr's signature scheme, which is presented in Chapter 2.

1.3.2 Electronic Cash

This section briefly describes a simple e-cash mechanism that will be modified in the remainder of this chapter. This scheme is based on the blind signature scheme described in Section 1.3.1. We focus on mechanisms to avoid double spending of coins in an off-line context.

Basic Scheme

Digital money or electronic cash is the most ambitious solution for electronic payment systems [She00]. Indeed, e-cash aims at defining a new type of money based on digital information while respecting the properties of classical money that are, anonymity, untraceability, and difficulty of counterfeiting. Table 1.2 describes the basic protocol of electronic cash that assures the untraceability of clients. The bank has a public RSA key (e, n) and a private RSA key (d, p, q): $n = pq$ where p and q are large primes, e is selected so that $1 < e < \phi(n)$ and $gcd(e, \phi(n)) = 1$ with $\phi(n) = (p-1)(q-1)$, and d is computed such that $ed = 1 \mod \phi(n)$ and $1 < d < \phi(n)$. The client chooses randomly a blinding factor r in $\{1, 2, \ldots, n-1\}$ and computes the serial number sn (e.g. $m = \mathcal{H}(sn)$ where $\mathcal{H}$ is a public hash function). In steps 1) and 2), A gets a signature s' of B on the blinded message m' and computes a signature $s = s' \cdot r^{-1}$ on message m. In step 3), A pays the vendor C with the e-coin and in 4), the vendor cashes the e-coin.

Unlinkability and integrity can be achieved by using a blind signature or a proof of knowledge. However, unlinkability makes double spending prevention a difficult task in disconnected context. Indeed, vendors being not connected, it is not possible to forbid a malicious user from showing a one-time credential to multiple vendors. And, once the misbehavior is discovered, unlinkability disables any action against the cheater that remains anonymous. Next section presents a simple mechanism that prevents double spending.

Preventing Double Spending

Different approaches exist to forbid multiple uses in disconnected mode, e.g. [CFN89, Fer94, Bra93]. They always rely on the same principle: when an electronic coin is used by A, the vendor C sends a challenge c to A that replies with a response $resp = f(c, x)$ where x is a valuable information. Knowing one pair $(c, resp)$ does not reveal any information on x but knowing two pairs $\{(c_0, resp_0), (c_1, resp_1)\}$ allows to retrieve x. In e-cash, the secret x is the identity of the client. When the same e-coin is used by A to pay two vendors C_1 and C_2, the bank gets enough information to identify and penalize A (e.g. debit a second time A's account). The challenge response can be based on cut-and-choose protocols or on proofs of knowledge. In this chapter we will extend the *cut-and-choose*

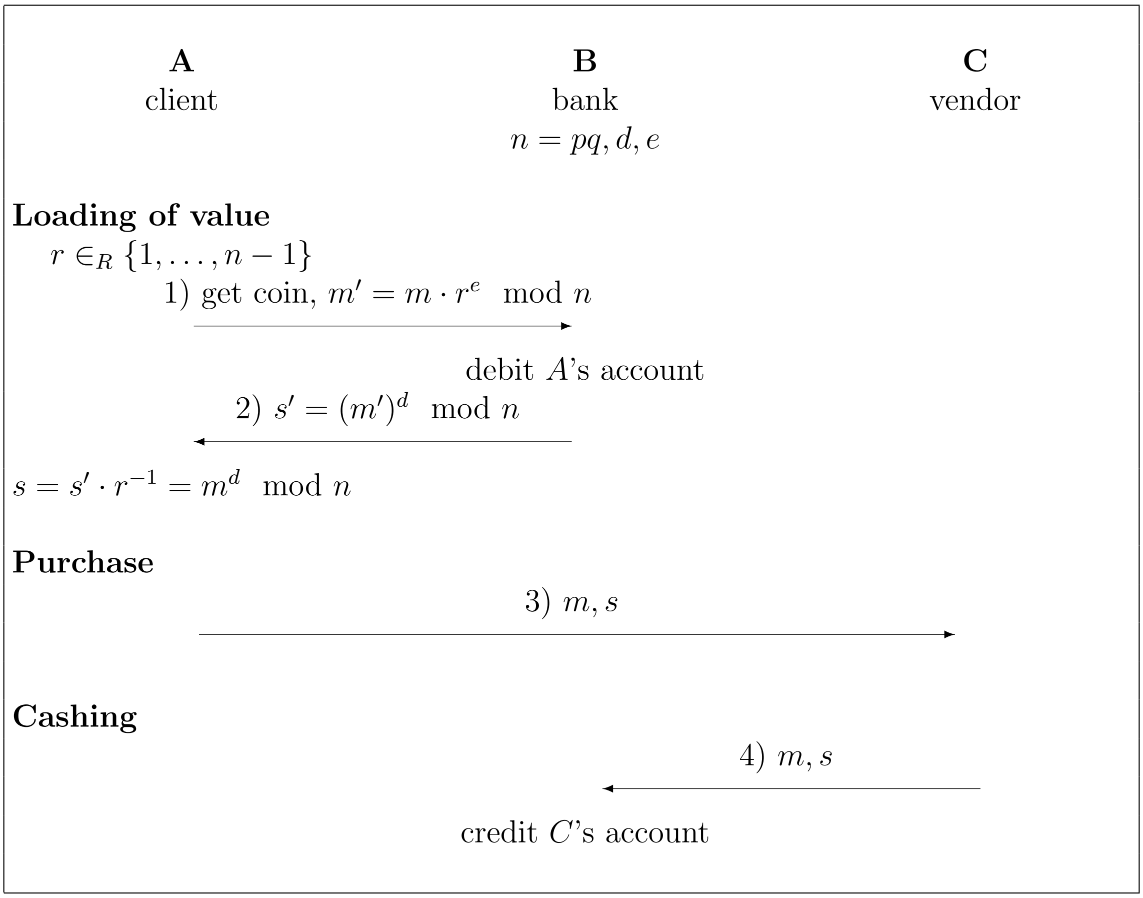


Table 1.2: Simple e-cash system ensuring client's untraceability

protocol [CFN89] that works as follows:

Alice A has a bank account numbered u and the bank keeps a counter v associated with it. Let f and g be two-argument collision-free functions; that is, for any particular such function, it is infeasible to find two different inputs that yield the same output.

1.1) A chooses a_i, c_i, d_i, and r_i independently and uniformly randomly in $\mathcal{Z}_n$ for all i such that $0 \leq i \leq k-1$ where k is a security parameter that is multiple of 4.

1.2) A forms and sends to the bank k blinded candidates m'_i

$$m'_i = r_i^e \cdot m_i \mod n \quad \text{for all } i \in \{0, 1, \ldots, k-1\}$$

where

$$m_i = f(x_i, y_i), \quad x_i = g(a_i, c_i), \quad y_i = g\left(a_i \oplus (u||(v+i)), d_i\right)$$

1.3) The bank chooses a random subset of $k/2$ blinded candidate indices $R = \{i_j\}$ where $i_j \in \{0, 1, \ldots, k-1\}$ for all j such that $0 \leq j \leq k/2 - 1$ and transmits it to A. The complementary set $\bar{R}$ is the set of $k/2$ blinded candidate indices that are not sent to A.

1.4) A reveals the a_i, c_i, d_i, and r_i values for all i in R, and the bank checks them as follows:

$$m_i' \stackrel{?}{=} f\left(g\left(a_i, c_i\right), g(a_i \oplus (u||(v+i)), d_i)\right) \cdot r_i^e \quad \text{for all } i \in R$$

Where $(u||(v+i))$ is known to the bank.

1.5) The bank charges Alice's account, increments her counter and sends her:

$$s_{\bar{R}}' = \prod_{i \notin R} s_i' = \prod_{i \notin R} (m_i')^d = \prod_{i \in \bar{R}} (m_i')^d \mod n$$

1.6) Alice can then easily extract the signature $s_{\bar{R}}$ on the electronic coin $m_{\bar{R}}$:

$$s_{\bar{R}} = s_{\bar{R}}' \cdot \prod_{i \in \bar{R}} r_i^{-1} = \cdot \prod_{i \in \bar{R}} (m_i')^d \cdot r_i^{-1} = \cdot \prod_{i \in \bar{R}} (m_i)^d = (m_{\bar{R}})^d \mod n$$

This process allows Alice to get a valid electronic coin that is signed by the bank but that cannot be traced when used only once. To pay the vendor C, Alice and the vendor proceed with the following steps of the protocol.

2.1) A sends the coin $s_{\bar{R}}$ to the vendor C.

2.2) C chooses T a random subset of $\bar{R}$ such that $T = \{i_j\}$, where $i_j \in \{0, 1, \ldots, k/2-1\}$ for all j such that $0 \leq j \leq k/4 - 1$ and C transmits T to A.

2.3) Let us define $\bar{T} = \bar{R} \backslash T$. For all $i \in \{0, 1, \ldots, k/2-1\}$, A responds as follows:

- If $i \in T$, then A sends a_i, c_i, and y_i to C.
- If $i \in \bar{T}$, then A sends x_i, $(a_i \oplus (u||(v+i)))$, and d_i to C.

2.4) C verifies that A's response is consistent with $s_{\bar{R}}$ and m_i where $i \in \bar{R}$ based on the following tests:

- If $i \in T$, then $m_i = f\left(g(a_i, c_i), y_i\right)$
- If $i \in \bar{T}$, then $m_i = f\left(x_i, g(a_i \oplus (u||(v+i)), d_i)\right)$

$$s_{\bar{R}}^e \stackrel{?}{=} \prod_{i \in \bar{R}} m_i \mod n$$

2.5) C subsequently sends $s_{\bar{R}}$ and A's responses to the bank, which verifies their correctness and credits C's account.

The bank must store $s_{\bar{R}}$ and A's responses. If A uses the same coin twice, then she will have a high risk of being traced: with high probability, vendors will define different subsets T_1 and T_2. The bank will have both a_i and $a_i \oplus (u||(v+i))$ when $i \in T_1$ and $i \in \bar{T}_2$ or when $i \in \bar{T}_1$ and $i \in T_2$. The bank can thus trace the payment to A's account.

1.4 Our Solution: One-time and Off-line Credentials

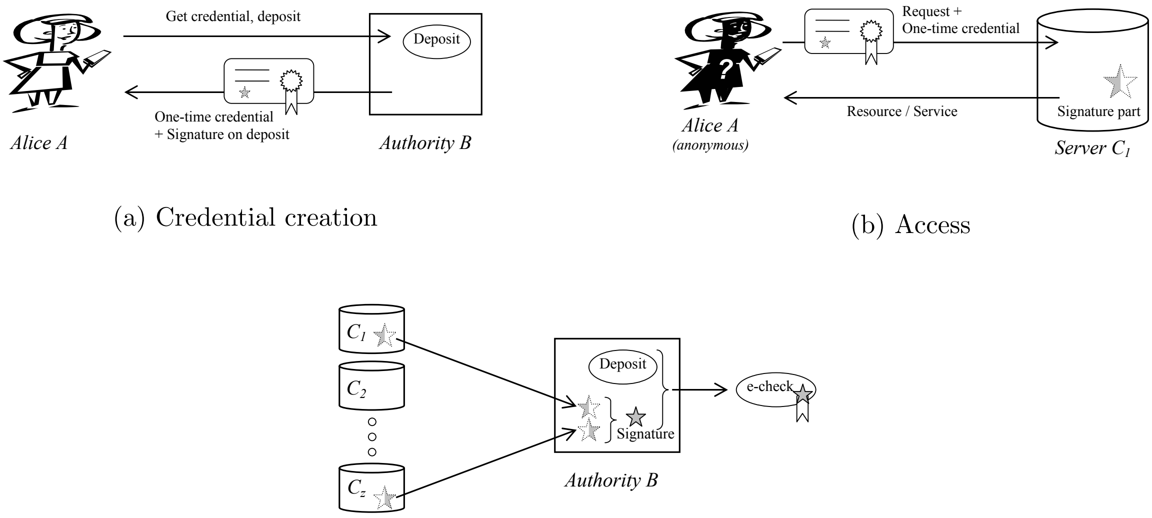
Section 1.3.2 shows that verification and penalty have to be postponed when one-time credentials are used off-line without client side trusted tamper-resistant devices such as smart cards. The penalty is generally based on retrieving the identity (or account number) of the cheater in order to debit the account of the cheater. In pervasive computing, due to the absence of a global organization, disclosure of identity cannot serve as a deterrent against double spending. We thus propose a more direct penalty mechanism whereby the cheater immediately looses money in case of double use of a one time credential.

In order to come up with an access control solution that meets the requirements of the ubiquitous application scenario, we introduce the concept of one-time credential (OTC): rights granted to the holder by the issuer and only usable once. The OTC issued by the authority represents the right to perform a single access to a resource. A OTC can be verified by a server off-line, that is, without any interaction with another server or with the local authority. The validation of the access right encoded in the OTC does not require the authentication of the visitor that issued the request including the OTC; the visitor only needs to prove that it is the party to whom the OTC was granted. The ultimate issue in this context is the assurance of the one-time property with off-line servers. Our solution to this problem is based on the postponed punishment principle, inspired by electronic cash, that if a visitor uses an OTC more than once then the violation of the one-time property will necessarily be detected later and cause the punishment of the cheating visitor with a penalty mechanism. Unlike electronic cash whereby the penalty consists of the disclosure of the cheater's identity and compensation of double spending by a banking organization that is trusted both by the payers and the payee, the OTC penalty mechanism does not require a unique banking organization or access control authority for all visitors and servers. The OTC penalty mechanism is based on a universal payment order or an electronic check (e-check). The payment order or the e-check do not need to be issued by a unique banking organization, any order issued by a financial organization recognized by the authority is suitable for the purpose of the OTC mechanism. Since visitors mutually distrust the authority and the servers, the payment order (called the *e-check* for the sake of simplicity) embedded in an OTC has to be protected against possible misbehavior as follows:

- The authority or the server should not be able to cash the e-check if the OTC is properly used (only once) by the visitor.
- The authority should be able to verify that a valid e-check is embedded in the OTC.

Finally, the one-time credential mechanism ensures untraceability of fair users thanks to blind signatures.

The solution consists of three phases: First, during the *credential creation* phase (Figure 1.2(a)) the authority B provides a set of OTC to a visitor entering the PCE.



(a) Credential creation

(b) Access

(c) Detection of double use

Figure 1.2: Different steps of one-time credential lifecycle

Apart from the classical access control operations through which access rights will be granted in terms of credentials, the main purpose of the protocol in this phase is twofold: to prove the authority that it will be able to cash the e-check if the visitor misbehaves (uses the OTC more than once within the PCE) and to assure that the authority cannot cash the e-check if the visitor properly behaves. These purposes are fulfilled by a new mechanism that allows B to verify that the e-check is properly filled and that the OTC includes a valid signature on this e-check that is revealed only in case of misbehavior.

In the second phase, the visitor uses a OTC to *access* resources kept by various servers (Figure 1.2(b)). The resource access takes place off-line, that is, the server cannot rely on the authority to verify the credential. When the visitor proves that she knows the secret corresponding to the credential, part of the information to retrieve the signature is provided to the server. This information is not sufficient to get a valid signature but prevents double use of the OTC.

Last, *detection of double use* is necessary to identify and punish visitors that use an OTC more than once (Figure 1.2(c)). This phase is postponed as long as the servers are not on-line. With off-line servers, visitor access logs will be provided in batch by servers to the authority (for instance through a daily data collection by service personnel). When the use of the same OTC appears in more than one server's log, the authority B is able to retrieve the signature of the e-check embedded in the OTC and to cash this e-check.

1.5 Protocol

This section presents the one-time credential protocols for credential creation, service access, and detection of double use. In this section we keep the notations of Section 1.3 for RSA parameters (n, p, q, d, e) and security parameter (k).

1.5.1 Penalty without Hierarchical Relationships

In the OTC mechanism, an electronic check serves as a deterrent against double use of a one-time credential. Common e-checks [She00] are not sufficient in this context because it is not possible to verify that they have been signed without revealing the signature that allows one to cash them. A new signature scheme has thus to be used when the client A signs the e-check, when service provider B verifies the signature, and when B cashes this e-check. This mechanism can replace signature of on-line as well as off-line e-checks. For the sake of simplicity we only present a basic scheme where e-checks are on-line payment orders: A orders his bank to transfer some amount from his account to B's account.

During the creation of a one-time credential, the client A can prove to the service provider B that a secret K such that $\mathcal{H}(K) \in HK$ where $HK = \{\mathcal{H}(K_0), \mathcal{H}(K_1), \ldots, \mathcal{H}(K_{k-1})\}$ will be revealed in case of double use (Section 1.5.2). An unsigned e-check is defined as $uc = \text{SIGN}_{\text{bank}}\,(A, B, \text{amount}, \text{sn})$ where A and B could be substituted with the account identification of A and B, respectively. This unsigned e-check is created during an on-line interaction with the bank. A provides as deposit a signed e-check sc such that $sc = \text{SIGN}_{\text{A}}(uc, HK)$. This deposit can only be cashed by the service provider B when one of the secret $K \mid \mathcal{H}(K) \in HK$ is known. A valid e-check vc consist of a deposit and a corresponding secret: $vc = \{sc, K\}$. It can be endorsed and cashed by B. Table 1.3 summarizes those different e-checks.

short name	**Description**
uc	unsigned e-check (filled by V and signed by the bank). $uc = \text{SIGN}_{\text{bank}}(\text{payer}, \text{beneficiary}, \text{amount}, \text{sn})$ For instance, $uc_1 = \text{SIGN}_{\text{Bank}}(PK_A, PK_B, 10\$, 001)$
sc	signed e-check (also called deposit). $sc = \text{SIGN}_{\text{payer}}(uc, HK)$ where $HK = \{\mathcal{H}(K_0), \mathcal{H}(K_1), \cdots, \mathcal{H}(K_{k-1})\}$ For instance, $sc_1 = \text{SIGN}_{\text{A}}(uc_1, HK_1)$
vc	valid e-check (that can be directly cashed). $vc = \{sc, K\}$ where $\mathcal{H}(K) \in HK$.

Table 1.3: Notations for various states of an electronic check

1.5.2 Phase 1: Credential Creation

One-time credentials are created by the authority B that optionally can authenticate visitor A. The protocol guarantees that a penalty (i.e. a signature on the deposit) is embedded within the credential and that B cannot obtain a valid electronic check during this process (see Figure 1.2(a)).

Authentication is optional in this protocol. Any entity able to provide an e-check could receive some rights. The only requirement in this phase is that the requestor has a valid account. However, some authorizations can be restricted to visitors having some attributes, e.g. employee of a partner company or role. In this case authentication would be required. To that effect, attribute certificates and challenge-response protocols or face to face authentication based on paper id-cards could be used. The following protocols do not address this point.

The visitor and the authority have to negotiate rights to be granted and the corresponding deposit. The authority B proposes some authorizations to the visitor A and asks for a corresponding deposit. At the end of this phase, A and B have agreed on the authorization (rights) that will be provided to A and the penalty (e-check) that will be cashed if the credential is used twice. An unsigned e-check is generated: $uc_A = \text{SIGN}_{\text{Bank}}(A, B, \text{amount})$

Let A and B define k that is the size of the set that will be used in the cut-and-choose protocol [CFN89] during the creation of one-time credentials and the access to services. k is a security parameter that determines the probability of undetected double use (see Section 1.6).

1.1) A generates keys K_i for all $i \in \{0, 1, \ldots, k-1\}$
A commits keys $HK = \{\mathcal{H}(K_0), \mathcal{H}(K_1), \cdots, \mathcal{H}(K_{k-1})\}$

$K_0, K_1, \ldots, K_{k-1}$ are kept secret by A and HK is a commitment on those secrets that is partially opened by A during further steps of the protocol.

1.2) A chooses $a_i, c_i, d_i, r_i \in_R \mathcal{Z}_n$ for all $i \in \{0, 1, \ldots, k-1\}$
A computes $m'_i = m_i \cdot r_i^e \mod n$ for all $i \in \{0, 1, \ldots, k-1\}$
where $m_i = \mathcal{H}(x_i || y_i)$
where $x_i = \mathcal{H}(a_i || c_i)$ and $y_i = \mathcal{H}((a_i \oplus data_i) || d_i)$
and where $data_i = \{K_i || \text{sn}\}$

This construction is necessary for avoiding double use of one-time credentials. It assures that $data_i$ can only be revealed when a_i and $(a_i \oplus data_i)$ are known, i.e. when the credential has been used twice (see Step 3.2). a_i, c_i, d_i and r_i are random numbers. This mechanism is similar to the one used in electronic cash to reveal the identity of double spenders. Here, due to the lack of a shared organization, disclosure of identity has no effect. With this protocol, cheating results in the cashing of the e-check: $data_i$ contains a secret K_i and a reference to a deposit (check number sn). A valid e-check is obtained when combining the secret and the deposit.

1.3) $A \rightarrow B$ $HK,\, m'_0, \ldots, m'_{k-1},\, uc_A$

A sends k blind candidates $(m'_0, \ldots, m'_{k-1})$, k commitments on keys (HK), and the deposit to the authority B in order to create the one-time credential. Before releasing the one-time credential, B verifies that a valid signature on the deposit is embedded in the credential.

1.4) $B \rightarrow A$ $R \subset \{0, \ldots, k-1\}$ where $|R| = \frac{k}{2}$

The authority B chooses randomly a subset R (half of the blind candidates) for verification purposes and requests A to send details on how each m'_i where $i \in R$ is constructed.

1.5) $A \rightarrow B$ a_i, c_i, d_i, r_i, K_i for all $i \in R$

A discloses the details to construct each $m'_i \mid i \in R$ and open commitments on related keys for verification purposes.

1.6) B verifies $\mathcal{H}(K_i) \overset{?}{\in} HK$ for all $i \in R$

$m'_i \overset{?}{=} \mathcal{H}\left(\mathcal{H}(a_i||c_i)||\mathcal{H}((a_i \oplus data_i)||d_i)\right) \cdot r_i^e$ for all $i \in R$

B verifies the results. When all m'_i with $i \in R$ are well-constructed, there is a high probability (see Section 1.6) that other m'_j where $j \in \bar{R}$ contain a secret K_j that is required to generate a valid e-check from the deposit.

1.7) B computes $s'_{\bar{R}} = \prod_{i \in \bar{R}} (m'_i)^d \mod n$

B computes $HK_{\bar{R}} = \{\mathcal{H}(K_i) \mid i \in \bar{R}\}$

Both B and A suppress verified secrets (in R) and keep unrevealed secrets (in $\bar{R}$) to build the one-time credential and the deposit. The authority B requests the deposit.

1.8) $A \rightarrow B$ $sc_A = \mathrm{SIGN_A}(uc_A, HK_{\bar{R}})$

A signs the set of unrevealed secrets. This is the deposit that has to be combined with one of the secrets K_i where $i \in \bar{R}$ to get a valid e-check. However, the protocol assures that K_i can only be revealed when A attempts to use his credential more than once.

1.9) $B \rightarrow A$ $s'_{\bar{R}}$

B stores the deposit and provides the one-time credential to A. As with a simple public key infrastructure, authorizations are application specific. The actual format and details of authorizations thus are out of the scope of this chapter. A unblinds the credential that will be used in the second phase of this protocol as follows:

$$s_{\bar{R}} = s'_{\bar{R}} \cdot \prod_{i \in \bar{R}} r_i^{-1} \mod n$$

1.5.3 Phase 2: Service Access with One-time Credential

Visitor A shows her credential to one of the servers $C_1, C_2, \cdots, C_z$ in order to get access to resources or services (see Figure 1.2(b)). Servers cannot rely on the authority B during this phase. This phase is very similar to spending an e-coin and the specific properties of our protocol do not appear here.

2.1) $A \rightarrow C$ Resource access request, $s_{\bar{R}}$

A interacts with a server C that trusts authority B. A requests a resource and provides the one-time credential $s_{\bar{R}}$ to prove that the operation is authorized. C verifies with respect to the security policy that the resource request can be authorized, checks that the credential is signed by B and that it is still valid.

2.2) $C \rightarrow A$ $T \subset \bar{R}$ where $|T| = \frac{|\bar{R}|}{2}$

C starts a challenge-response based on cut and choose: The server chooses randomly a subset T (half of the set $\bar{R}$) and sends it to A. This step has two goals: verifying that the visitor knows the secret corresponding to the credential and forcing this visitor to reveal some information in order to forbid double use of the credential.

2.3) $A \rightarrow C$ a_i, c_i, and y_i for all $i \in T$
x_i, $(a_i \oplus data_i)$, and d_i for all $i \in \bar{T}$

A reveals half the information for the set $\bar{R}$ to prove that it can construct $m_{\bar{R}}$. However, C has no way to get any $data_i$ where $i \in \bar{R}$ and thus cannot collude with B to cash the e-check.

2.4)	C computes	$m_i = \mathcal{H}\left(\mathcal{H}(a_i \| c_i) \| y_i\right)$	for all $i \in T$
		$m_i = \mathcal{H}\left(x_i \| \mathcal{H}((a_i \oplus data_i) \| d_i)\right)$	for all $i \in \bar{T}$
	C verifies	$(s_{\bar{R}})^e \overset{?}{=} \prod_{i \in \bar{R}} m_i$	

C verifies that the visitor knows the secrets corresponding to the credential. A can be authorized to access the resource.

1.5.4 Phase 3: Detection of Double Use and Penalty

Servers $C_1, C_2, \cdots, C_z$ are periodically on-line and thus able to send data to the authority B. If a one-time credential has been used twice, there is a high probability that B can retrieve the embedded penalty (i.e. a valid e-check) and use it.

3.1)	$C_1 \rightarrow B$	$s_{\bar{R}}$	
		a_i, c_i, y_i	for all $i \in T_{C_1}$
		$x_i, (a_i \oplus data_i), d_i$	for all $i \in \bar{T}_{C_1}$

Periodically, when the server C_1 is on-line, it sends relevant data to B (dotted star of Figure 1.2(c)). The set T_{C_1} has been randomly chosen by C_1 and is different for each server and for each credential. As long as A does not cheat, i.e. does not use twice the same OTC, those data are useless.

3.2)	$C_2 \rightarrow B$	$s_{\bar{R}}$	
		a_i, c_i, y_i	for all $i \in T_{C_2}$
		$x_i, (a_i \oplus data_i), d_i$	for all $i \in \bar{T}_{C_2}$

If the same one-time credential has been used with servers C_1 and C_2, there is a high probability that there exists an i such that a_i and $(a_i \oplus data_i)$ are known. B can thus retrieve $data_i$ and the secret K_i. This secret combined with the deposit is a valid e-check: $vc_A = (sc_A, K_i)$ where $\mathcal{H}(K_i) \in HK_{\bar{R}}$. The authority B can send the e-check to the bank in order to cash it.

1.5.5 Defining Attributes

Attributes have to be associated to the one-time credential in order to define rights or roles. This could be achieved by using different keys for different rights like printing a page or getting a coffee. More efficiently, this can be done by slightly modifying the RSA blind signature: instead of defining s' as $s' = (m')^d$, we use $s' = (m')^{d'}$ where $d' = \prod d_i$

mod $\phi(n)$ with e_i is the ith odd prime.

The decimal attribute value 12_d or its binary representation 01100_b can thus be defined as $s' = (m')^{d_3 d_4}$ because the third and fourth bit (numbering from the least significant bit) are set to one. More details on this approach are given in Section 3.5.1.

1.6 Security Evaluation

Previous sections define a solution to avoid double use of one-time credentials in off-line context. When a credential is used more than once, there is a high probability that an electronic check is revealed.

The first requirement for one-time credential scheme presented in this chapter is that credentials can only be issued by the legitimate issuer as summarized by the following definition.

Requirement 1.1 ***(Signature Integrity)*** *it is infeasible to generate a one-time credential without knowing the private key of the service provider B.*

Proposition 1.1 *The one-time credential scheme is conformant to Requirement 1.1*

Proof: Generating a blind signature with attributes requires the knowledge of the private key of B. $s'_{\bar{R}} = \prod_{i \in \bar{R}} (m'_i)^{d'}$ where $d' \cdot e' = 1 \mod \phi(n)$ and $e' = \prod e_j$ where e_j are primes. d' cannot be computed from e' without knowing $\phi(n) = (p-1)(q-1)$ and d' can thus only be computed from d_0, d_1, etc. where $d_0 \cdot e_0 = 1 \mod \phi(n)$. In other words, breaking the blind signature scheme is equivalent to breaking the RSA signature scheme. □

The second requirement for one-time credential scheme is that credential holders are not traceable.

Requirement 1.2 ***(Untraceability)*** *The credential provided by B to user A does not enable servers C or authority B to trace A.*

Proposition 1.2 *The one-time credential scheme is conformant to Requirement 1.2*

Proof: The credential $s_{\bar{R}}$ is unconditionally independent of $s'_{\bar{R}}$. Our scheme respects this property because the blinding factors r_i ensure that $s_{\bar{R}}$ is unconditionally independent of

$s'_{\bar{R}}$. Indeed, $s_{\bar{R}} = s'_{\bar{R}} \cdot \prod_{i \in R} r_i^{-1}$ where r_i are random values. Moreover, the $k/2$ remaining blinded candidates are reordered so that there is no more trace of $\bar{R}$. □

The third requirement for one-time credential scheme is that the service provider has the guarantee that he will be able to cash the deposit in case of misbehavior.

Requirement 1.3 ***(Server Side Safety)*** *A valid electronic check will be revealed with a high probability in case of double use of a one-time credential*

Proposition 1.3 *The one-time credential scheme is conformant to Requirement 1.3.*

Proof: The probability that a one-time credential without valid e-check exist depends on k and can be chosen as small as required. Attacks against this scheme require that the visitor A can obtain a valid credential that does not embed an e-check. The credential creation protocol ensures that the secrets of half the set are verified. Thus, when an attacker tries to generate a credential that will never reveal a valid e-check, she has to provide $k/2$ invalid secrets. The probability that the service provider B does not verify one of those invalid data is:

$$\mathbf{p}_{\text{no e-check}} = \underbrace{\frac{\frac{k}{2}}{k}}_{\text{valid } m'_1} \cdot \underbrace{\frac{\frac{k}{2}-1}{k-1}}_{\text{valid } m'_2} \cdot \quad \cdots \quad \cdot \underbrace{\frac{1}{\frac{k}{2}+1}}_{\text{valid } m'_{k/2}} = \frac{\frac{k}{2}!}{\frac{k!}{\frac{k}{2}!}} = \frac{(\mathbf{\frac{k}{2}!})^{\mathbf{2}}}{\mathbf{k!}} \qquad \left(= \frac{1}{C_k^{k/2}} \right)$$

By properly setting the size k of the cut and choose protocol, it is possible to choose a probability of successful attack as small as required. For instance, if $k = 100$, $p_{\text{no e-check}} \cong 2^{-96}$ □

The fourth requirement for one-time credential scheme is that the service provider cannot cash the deposit without misbehavior. It is important to protect the visitor against a malicious service provider trying to get a valid signature for an existing deposit in order to retrieve a valid e-check.

Requirement 1.4 ***(Client Side Safety)*** *It is infeasible to obtain a valid e-check when the user A is behaving fairly.*

Proposition 1.4 *The one-time credential scheme is conformant to Requirement 1.4.*

Proof: The attacker has to find a valid K_i corresponding to an embedded $\mathcal{H}(K_i)$ where $i \in \bar{R}$. The birthday attack is not relevant in this case and the probability of a successful brute force attack against the hash function is:

$$p_{disclose} = \underbrace{\frac{k}{2}}_{|\bar{R}|} \cdot \underbrace{2^{-l}}_{hash} \qquad \text{where } l \text{ is the size of the hash output.}$$

For instance, using $k = 100$ and hash function SHA-1 ($l = 160$ bits), $p_{disclose} \cong 2^{-154}$ □

Impact of Multiple Use

Because each server has to keep track of credentials it already received, double use attempts performed with the same server are detected by the server itself. Double use of an OTC with different servers on the other hand will be detected by the authority based on the protocol and the penalty mechanism. But when the same OTC is used more than twice with different servers, only one e-check can be cashed as part of the penalty mechanism. The degree of the penalty (the amount of the e-check) should thus be set according to the type of resource and to the threat model: when access control matters, the penalty has to be sufficiently important in order to eliminate possible advantages of multiple uses beyond the double use; when one-time credentials are used to disable denial of service attacks, small penalties can be sufficient.

Protection of the Visitor

It is important to assure that credentials of a user and corresponding secrets cannot be disclosed by intruders since based on this information an intruder could get a valid e-check. It is also necessary to prevent any sequence of operations that could cause unintended double use by the visitor.

Moreover, a rogue server could perpetrate a denial of service attack by getting a one-time credential without delivering the requested service. Even though this type of attack does not provide any benefit to the attacker, it would prevent further legitimate access to the service by the visitor. To restrict denial of service attacks, it could be necessary that the authority certifies the servers so that visitors can verify them before providing one-time credentials.

Validity End

Off-line scenarios, cannot rely on certificate revocation lists to verify the validity of long-term credentials. The alternative to revocation, which still does not suit the off-line nature of the ubiquitous application scenario, consists of short-term certificates that in turn require frequent interaction between the holder of the certificate and the issuer for the renewal of certificates. As opposed to these alternatives, the validity of one-time credentials presented in this chapter does not rely on time. However, it can be interesting to introduce a lifetime for OTC and deposits in order to limit the storage of data in time. When the appliances $C_1, \ldots, C_z$ cannot afford a real-time clock, the authority B can act as a trusted time server to synchronize the servers when getting in touch with them to collect the access logs. B can thus discard deposits that are no more required by any server.

1.7 Conclusion

This chapter presents a mechanism for implementing off-line one-time credentials. It focuses on the separation of duty in trust model: the bank guarantees that a visitor A will be able to pay in case of misbehavior but this bank does not deal with access rights; the service provider B defines rights without having to rely on an *a priori* relationship with A but only has to check that A can sign e-checks; and A has the guarantee that as long as she behaves fairly, no valid e-check can be computed by the service provider or by the bank. This scheme seems to be promising for countering denial of service (DoS) attacks in pervasive computing. For instance, a shopping mall offering free printing facilities to visitors may want to avoid DoS attacks whereby malicious clients print anonymously hundreds of pages to exhaust printer resources. In this case our scheme would assure that attackers would loose their deposit. Money would then be a good deterrent against such attacks.

This scheme is based on the cut-and-choose protocol of the classical e-cash scheme and it would be more efficient to use other schemes [Fer94, Bra93] that do not rely on this protocol. However, it is still an open issue whether such schemes could fulfill our requirements. Our one-time credential scheme requires a dedicated e-check mechanism and the deployment of our proposal would thus need that banks accept to deal with such e-checks. *Verifiable encryption* schemes [CD00a] enable to verify that a valid signature is encrypted without revealing this signature. It seems to be a promising approach to prove that credentials will indeed reveal a valid signature in case of double use. Moreover verifiable encryption could perhaps allow embedding ”standard” e-check within our scheme.

The solution that has been presented in this chapter is suitable to scenarios where there is no way to establish trust among parties. However, impossibility of establishing trust leads to an extreme case with several limitations. First, one-time credentials have limited coverage with respect to access control. Second, the concept of a penalty mechanism based on real money can also be questionable in some scenarios despite the fairness guarantee of our scheme. In order to cope with those limitations, the remainder of this thesis will focus on trust establishment where these limits do not exist. First, in Chapter 2 we will assume that users visit more than once a pervasive computing environment and that it thus is possible to build trust based on previous interactions. The second part of this thesis, Chapters 3, 4, and 5 proposes an architecture to establish trust relationships based on a history of interactions when users have a valuable secret.

Chapter 2

Establishing Trust without Infrastructure

> *"Hence, when able to attack, we must seem unable; when using our forces, we must seem inactive; when we are near, we must make the enemy believe we are far away; when far away, we must make him believe we are near."*
>
> – **Sun-Tzu** (The Art of War)

This chapter proposes an original way to build trust. It is assumed that there is no *a priori* relationships among the client A and the service provider B. We assume multiple interactions between A and a set of service providers B, B', etc. so that trust can be based on a history of interactions. B can provide some untraceable credential to A defining the quality of the interaction that occurred. During a subsequent interaction with B or B', which is a partner of B, A can show this credential to demonstrate some previous relationship. In this thesis, such mechanism is called *history-based trust establishment.* This chapter presents a protocol implementing history-based trust establishment with a particular accent on privacy in terms of untraceability of interactions.

2.1 Introduction

Chapter 1 has shown that a service provider can give access rights to another party without requiring trust relationships when neither *a priori* trust nor trust establishment apply. However a guarantee of direct penalty is necessary, and the solution presented in Chapter 1 thus relies on the fact that service providers can cash a deposit in case of misbehavior.

This chapter proposes to establish trust relationships when *a priori* trust among parties is lacking. In this case, it is still impossible to rely on a public key infrastructure and identity based authentication is meaningless [SFJ$^+$03] but trust relationships can however be established. Moreover, lack of identification mechanisms does not prevent malicious parties from monitoring user behavior, thus privacy still is a major concern in this environment.

Trying to build history-based trust establishment protocols with privacy, we face a dilemma: on one hand, B wants to keep track of previous interactions with A in order to be able to evaluate whether A is trustworthy and thus whether some rights can be granted to A, on the other hand privacy calls for unlinkability among interactions. A tradeoff between both properties is that B delivers A unlinkable credentials that encode the degree of trust granted to A by B. The degree of trust should be encrypted in order to assure that A will show negative as well as positive statements. Moreover, encryption of attributes is a pragmatic way to ensure non-transferability of credentials that cannot be linked to a valuable secret like a private key. A party would thus be able to prove that she was previously recognized as reliable by the entity she is interacting with. After each interaction, a credential is provided in order to subsequently assert what happened.

Assuring privacy in this context means that credential issuers cannot trace users through the credentials they delivered them. More precisely, a credential has to be created in a way that prevents the issuer from recognizing the credential when it is presented. As in Chapter 1, we use blind signature mechanisms to ensure unlinkability of the message and the signature. It is therefore required to have a way to verify that the secret attribute is the encryption of one element of a public set of cleartexts. Otherwise, if the holder could embed any encrypted attribute, he could attach a unique identifier to each credential in order to trace holders. We thus suggest to use a mechanism proving that the secret is the encryption of an element of a public set of values. This makes it possible for a credential holder to prove his history of interactions to the issuer or to one of the issuer's partners without being linkable to a previous event (untraceability) and without revealing his identity (anonymity).

2.2 Problem Statement

Throughout this chapter a basic scenario will be used to illustrate our solution. In this scenario, Bob (B) meets Alice (A), he interacts with her and gives her a credential as an evidence of the interaction. Subsequently, A comes back and shows her credential to B. The new interaction depends on the first one but should be unlinkable as long as B provides enough credentials to other users.

For instance, A interacts with B that provides three types of credentials (*excellent*, *good*, or *poor*) to qualify an interaction. A is qualified as *good* by B. We assume that tens of other users interacts with B before A comes back and show her credential to B or to a partner of B. The unlinkability of the credential makes it impossible to recognize the credential and the small set of attribute values disable B to know whether he is interacting with A or any other user tagged as *good*.

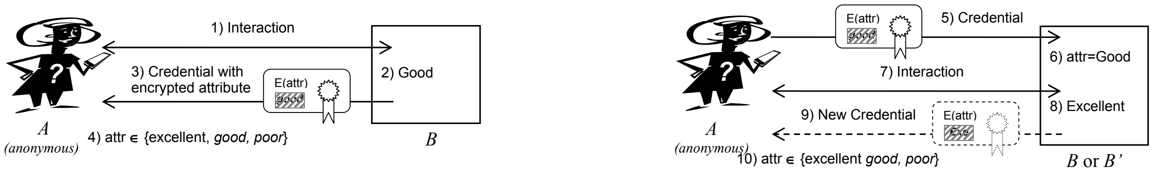
It is obvious that the cardinality (u) of the set of possible attributes has to be small. Indeed, a malicious service provider could reserve up to $u-1$ attributes values for tracing up to $u-1$ clients. This approach however ensures the unlinkability of the majority of users. In the previous example where the cardinality is equal to three and assuming a population of one hundred users, the privacy of ninety-eight users is assured and only two users could be traced in the worst case.

2.2.1 Expected Features

This section summarizes the different properties that are expected when defining an unlinkable credential scheme dedicated to trust establishment in pervasive computing.

- **(Disconnected verification)** It is possible to locally verify the validity of an unlinkable credential without relying on classical protocols based on on-line trusted third parties.
- **(Credential Integrity)** It is infeasible to generate an unlinkable credential without knowing the private key of the issuer.
- **(Credential Unlinkability)** The issuer B or any other party cannot trace the credential. In other words, the issuer B must not be able to recognize the credential that has been unblinded and thus when A comes back, B does not know that she is talking to the same entity.
- **(Attribute Secrecy)** As in Chapter 1, we assume that users do not have a valuable secret certified by some trusted entity, unlike in a public key infrastructure. In order to avoid transferability and credential trading, we thus propose to keep secret attribute value of credentials. In order to achieve attribute secrecy while protecting A's privacy, the credential scheme has to fulfill the following requirements:
  - A cannot decrypt the attribute of a credential: A cannot know whether she was described as *good* or *poor*.
  - A can verify that the credential's secret attribute is part of a public set of values, e.g. {*very poor, poor, fair, good, excellent*} or {0, 1, 2}. The cardinality gives to A a good estimate of the absence of risk that B may use secret values as covert channels for tracing A.
  - Probabilistic encryption ensures that A has no way to check whether two credentials embed the same secret attribute value.

- **(Attribute Decryption)** The issuer B and his trusted partners called B' can retrieve secrets embedded in credentials signed by B.



(a) Get credential

(b) Show credential

Figure 2.1: Creation and use of a credential with secret attribute

Figure 2.1(a) presents the steps for getting a credential and Figure 2.1(b) shows how such a credential can be used in order to assert previous degree of trust.

1) An interaction takes place between a client A and a service provider B.
2) B decides to tag A as *good* according to some measure of her fairness.
3) The credential containing the tag is provided to A
4) The attribute of the credential is encrypted so that A cannot know its value but she can verify that the value is an element of a restricted set of possible values, e.g. *excellent*, *good*, or *poor*.

The credential is unblinded by A so that it cannot be traced by B. Figure 2.1(b) presents the different steps that are required when using a credential.

5) The user A shows her unblinded credential to the service provider B or to B'. The possibility that the credential contains a positive feedback, which A cannot evaluate, represents an incentive for A to show her credential to B or B'.
6) The service provider B or B' can open the credential and retrieve the encrypted tag. However, he cannot trace the credential back to A.
7) The new interaction depends on history but cannot be linked to any specific previous interaction like step 1 and cannot be linked to any party, i.e. A.

After an interaction, the service provider can deliver a new credential as follows:

8) B decides to tag A as *excellent*.

9) The credential containing this tag is provided to A

10) As in step 4), the attribute of this credential is encrypted so that A cannot know its value, cannot know whether the attribute changed, but can verify that the value is an element of a restricted set of possible values.

2.3 Preliminaries

This section presents some mechanisms that will be used in the remainder of this dissertation: proofs of knowledge, signatures based on a proof of knowledge, and group signatures. Blind versions of those mechanisms are shortly described and are only used in this chapter.

- A *group signature* scheme allows group members to sign messages on behalf of the group. Signatures can be verified with a group public key but do not reveal the signer's identity. Only the group manager can *open* signatures, i.e. reveal the identity of the signer.

- A *blind signature* is a protocol in which a signer signs some message m without seeing this message (See section 1.3.1). It was first introduced by Chaum [CR82] to ensure untraceability of electronic cash.

- A *group blind signature* is a protocol in which a group member blindly signs a message. Only the manager can know who signed the message and no party can recognize the unblinded message.

All existing group blind signature schemes [Ram99] and [NMV99] are based on the group signature schemes proposed by Camenisch in [CS97]. The conclusion of [CM98] gives a quick sketch of two other possible approaches. In this chapter, we only describe the first blind group signature scheme [Ram99, LR98] and modify it so that it fulfills requirements of untraceable secret credentials. However, it seems possible to modify other schemes as well. The remainder of this section briefly presents the first group signature schemes that relies on signatures based on a proof of knowledge.

As a basis for the discussion on group signatures let us define some basic terminology: $n = pq$ where p and q are two large primes; $\mathcal{Z}_n = \{0, 1, 2, \ldots, n-1\}$ is a ring of integers modulo n; $\mathcal{Z}_n^* = \{i \in \mathcal{Z}_n \mid \gcd(i, n) = 1\}$ is a multiplicative group; $G = \{1, g, g^2, \ldots, g^{n-1}\}$ is a cyclic group of order n; g is a generator of this group G; $a \in \mathcal{Z}_n^*$ is an element of the multiplicative group.

2.3.1 Interactive Proof of Knowledge

The protocol of Table 2.1 shows an example of interactive proof of knowledge $\mathrm{PK}[\alpha \mid z = g^{(a^\alpha)}]$ where some prover A proves to a verifier B that she knows x the double discrete logarithm of z to the bases g and a without revealing any information on x. It is assumed that there is an upper bound of λ on the length of x, i.e. $0 \leq x < 2^\lambda$ (e.g. $\lambda = |n|$). The security parameter $\epsilon > 1$ defines the probability of disclosing information on x in statistical zero knowledge.

A		**B**
Prover claims to know double discrete log x such that $z = g^{(a^x)}$.		Verifier tests if A knows double discrete log of z to the bases g and a.
$r \in_R \{2^\lambda, \ldots, 2^{\lambda\cdot\epsilon} - 1\}$ $w = g^{(a^r)}$		
	witness w $\xrightarrow{\hspace{6em}}$	
		Flip a coin : $c \in_R \{0, 1\}$
	challenge c $\xleftarrow{\hspace{6em}}$	
$s = \begin{cases} r & \text{if } c = 0 \\ r - x & \text{otherwise} \end{cases}$		
	response s $\xrightarrow{\hspace{6em}}$	
		$w \overset{?}{=} \begin{cases} g^{(a^s)} & \text{if } c = 0 \\ z^{(a^s)} & \text{otherwise} \end{cases}$

Table 2.1: Interactive proof of knowledge of double discrete logarithm

The proof is successful if the results satisfy the following verification equations:

$$\begin{array}{lll} \text{if } c = 0 \ : & w \overset{?}{=} g^{(a^s)} & \left(= g^{(a^r)} = w \right) \\ \text{if } c = 1 \ : & w \overset{?}{=} z^{(a^s)} & \left(= \left(g^{(a^x)}\right)^{(a^s)} = g^{(a^x \cdot a^s)} = g^{(a^{x+s})} = g^{(a^r)} = w \right) \end{array}$$

To be sure that A knows the double discrete logarithm of z (i.e. with a low enough probability of successful attack), this protocol has to be run l times where l is a security parameter. Note that if the order of $a \in \mathcal{Z}_n^*$ is known, s can be computed modulo this order.

2.3.2 Schnorr Digital Signature Scheme

This section introduces the digital signature scheme proposed by Schnorr in [Sch89]. This scheme will be used to explain non-interactive proofs of knowledge. The Schnorr signature is a variant of the ElGamal scheme [MVO96] and is also based on the intractability of the discrete logarithm problem.

Principle

We describe the Schnorr signature using the group previously described: G is a cyclic group of order n. A generator $g \in G$ is chosen so that computing discrete logarithm in G is difficult. x is the private key of the prover A and $y = g^x$ is her public key. The method also requires a collision-resistant hash function $\mathcal{H} : \{0,1\}^* \rightarrow \mathcal{Z}_n$.

For a message $m \in \{0,1\}^*$, a pair (c, s) is said to be a valid Schnorr signature on m if it satisfies the following verification equation:

$$c = \mathcal{H}(m \parallel g^s y^c)$$

The variable c occurs in both the left hand and right hand side of the equation. Given that $\mathcal{H}$ is collision resistant, it appears to be quite difficult to construct such a valid signature. It turns out that it is feasible for one knowing the discrete logarithm of y to the base g. A knows $x = \log_g(y)$ and can compute a signature as follows:

- Choose $r \in_R \mathcal{Z}_n$
- Let $c = \mathcal{H}(m \parallel g^r)$
- Choose $s = r - cx \mod n$

(c, s) is a valid Schnorr signature of m because:

$$g^s y^c = g^{r-cx}(g^x)^c = g^{r-cx} g^{cx} = g^r$$

Hence, $c = \mathcal{H}(m \parallel g^r) = \mathcal{H}(m \parallel g^s y^c)$. A blind versions of this signature scheme exist [Ram99] and is used for defining group blind signatures.

2.3.3 Signature Based on a Proof of Knowledge

A *signature based on a proof of knowledge* (SPK) or *signature of knowledge* (SK) is a non-interactive version of a proof of knowledge where challenges are replaced by a message m. Such a signature proves that some party knowing a secret (e.g. the discrete logarithm of some public value) signed the message.

The signature based on a proof of knowledge of a double discrete logarithm of z to the bases g and a, on message m, with security parameter l is denoted $\text{SPK}_l[\alpha \mid z = g^{(a^\alpha)}](m)$. It is a non-interactive version of the protocol depicted in Section 2.3.1. The signature is an $l+1$ tuple $(c, s_1, \ldots, s_l)$ satisfying the equation:

$$c = \mathcal{H}_l(m \parallel z \parallel g \parallel a \parallel P_1 \parallel \ \ldots \ \parallel P_l) \quad \text{where } P_i = \begin{cases} g^{(a^{s_i})} & \text{if } c[i] = 0 \\ z^{(a^{s_i})} & \text{otherwise} \end{cases}$$

It is computed as following:

1. For all $1 \leq i \leq l$, generate random r_i.
2. Set $P_i = g^{(a^{r_i})}$ and compute $c = \mathcal{H}_l(m \parallel z \parallel g \parallel a \parallel P_1 \parallel \ \ldots \ \parallel P_l)$.
3. Set $s_i = \begin{cases} r_i & \text{if } c[i] = 0 \\ r_i - x & \text{otherwise} \end{cases}$

2.3.4 Group Signature

The group signature scheme in [CS97] is based on two signatures of knowledge: one that proves the signer knows some secret and another one that proves this secret is certified by the group manager. The scheme relies on the hardness of computing discrete logarithm, double discrete logarithm and eth root of the discrete logarithm.

The public key of a group is (n, e, G, g, a, λ) where e is chosen so that $\gcd(e, \phi(n)) = 1$ where $n = pq$. The private key of the manager is (p, q, d) where $de = 1 \mod \phi(n)$. When Alice *joins* the group, i.e. becomes a member, she uses her secret x to compute a membership key (y, z) where $y = a^x \mod n$ and $z = g^y$. A sends (y, z) to the group manager, proves that she knows x and receives a group certificate $(y+1)^d \mod n$ corresponding to her secret x. In order to sign a message m, A chooses $r \in_R \mathcal{Z}_n$ and computes $\tilde{g} = g^r$, $\tilde{z} = \tilde{g}^y$ $(= z^r)$, and two signatures:

$$V_1 = \text{SPK}[\alpha \mid \tilde{z} = \tilde{g}^{(a^\alpha)}](m)$$
$$V_2 = \text{SPK}[\beta \mid \tilde{z}\tilde{g} = \tilde{g}^{(\beta^e)}](m)$$

V_1 is a signature of knowledge of a double discrete logarithm that can be computed when knowing some secret x. Similarly, V_2 is a signature of knowledge of an eth root of the discrete logarithm that can be computed using the certificate $(y+1)^d \mod n$. The group signature of message m is $(\tilde{g}, \tilde{z}, V_1, V_2)$.

The verifier checks that V_1 and V_2 are valid signatures of m. Both signatures together mean that $\tilde{g}^{(\beta^e)} = \tilde{z}\tilde{g} = \tilde{g}^{(a^\alpha+1)}$ and thus $\beta = (a^\alpha + 1)^d \mod n$. The verifier knows that the signer holds a certified secret x. However, the verifier cannot get any information on x. In other words, the identity of the signer is preserved: this is a group signature.

2.3.5 Group Blind Signature

In the remaining of this chapter, we use the group blind signature scheme proposed in [Ram99], which is a blind version of the group signature scheme of Camenisch presented in the previous section. A more efficient group blind signature scheme is described in [NMV99]. It is based on another group signature scheme described in [CS97] and could potentially be used for our purpose but its security remains uncertain.

The following notations are used: the public key of group G is $K_{P_G} = \{n, e, G, g, a, \ldots\}$, the private key of group member A is $K_{S_A} = \{x, (a^x+1)^d\}$.

2.4 Untraceable Signature of Secret

This section shows how the group blind signature scheme presented in Section 2.3 can be used to define an untraceable signature of secret, which constitutes a basic building block of privacy-preserving trust establishment.

2.4.1 Principle

Untraceability is guaranteed by the blind signature mechanism. However, it is necessary to associate some attribute value to this signature. We propose to assign each signer B a set of private keys, e.g. $\{K_{S_{B,0}}, K_{S_{B,1}}\}$ and to let signer choose one of the key based on the attribute value. For instance, a random number signed with key $K_{S_{B,0}}$ should have a different meaning than this random number signed with key $K_{S_{B,1}}$. Attribute secrecy is assured through the group signature scheme: when all private keys are part of a same group, the verifier A cannot know which key was chosen and thus cannot discover the attribute value.

A new group is created for each entity that signs secrets. The group key becomes the entity's public key and the same signer uses different private keys according to the value of the attribute that has to remain secret (see right part of Figure 2.2). In other words, the blind signature assures the untraceability of the credential and the group signature assures the secrecy of attributes.

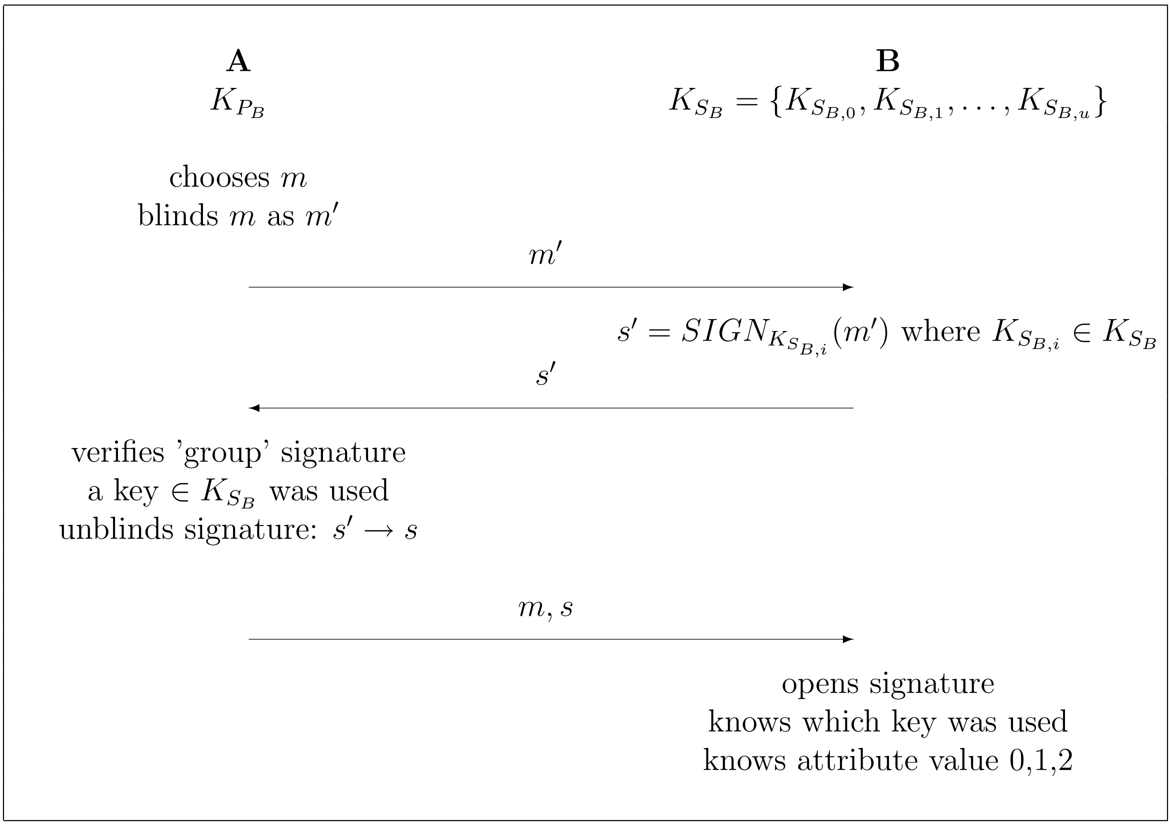

A
K_{P_B}
B
$K_{S_B} = \{K_{S_{B,0}}, K_{S_{B,1}}, \ldots, K_{S_{B,u}}\}$
chooses m
blinds m as m'
m'
$s' = SIGN_{K_{S_{B,i}}}(m')$ where $K_{S_{B,i}} \in K_{S_B}$
s'
verifies 'group' signature
a key $\in K_{S_B}$ was used
unblinds signature: $s' \rightarrow s$
m, s
opens signature
knows which key was used
knows attribute value 0,1,2

Table 2.2: Blind signature with secret attribute i

For instance (see Figure 2.2), a signer B that can use attribute values from the set $\{$*0:poor, 1:good, 2:excellent*$\}$, will have a group public key K_{P_B} and three private keys $K_{S_{B,0}}$, $K_{S_{B,1}}$, and $K_{S_{B,2}}$. When the signer wants to encrypt the value *excellent*, he signs with the corresponding private key $K_{S_{B,2}}$. Anybody can verify that the unblinded message has been signed with a private key corresponding to the public key K_{P_B} without knowing which key was used. When the unblinded message is subsequently shown to the signer, he cannot trace the holder but can *open* the signature to know which key was used and can thus retrieve the secret value.

2.4.2 Restricting Possible Values of a Signed Secret

Unfortunately, the chosen group signature scheme allows new members to join the group without modifying the group public key. In other words, it is not possible to know how many private keys exist for a given group public key. In the context of secret attributes, it means that the cardinality of the set of possible attributes cannot be deduced from the public key. To solve this problem it is necessary that the 'group manager' role be assumed by a trusted third party (TTP). This TTP would provide a set of private keys to the signer and certify the public key along with the number of related private keys that have been created. The certificate issued by the TTP is denoted by *CERT* in the following protocols. It could then be possible to ensure that the set of keys is fixed and that the secret attributes can only be the encryption of an element of a public set (see Figure 2.2).

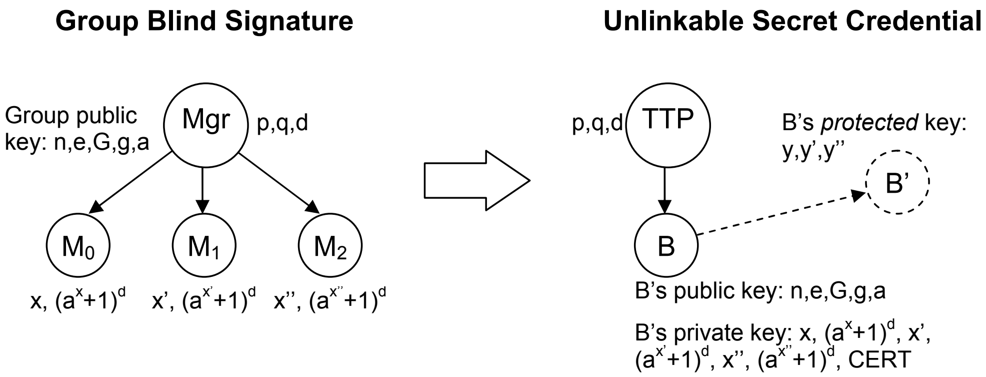

Group Blind Signature
Unlinkable Secret Credential
Group public key: n,e,G,g,a
Mgr
p,q,d
M0
M1
M2
x, (a^x+1)^d
x', (a^x'+1)^d
x'', (a^x''+1)^d
p,q,d
TTP
B's protected key: y,y',y''
B'
B
B's public key: n,e,G,g,a
B's private key: x, (a^x+1)^d, x', (a^x'+1)^d, x'', (a^x''+1)^d, CERT

Figure 2.2: Group blind signature scheme used as unlinkable secret credential

2.4.3 Protected Keys

In a group signature scheme, only the group manager can open signatures. In the context of unlinkable secret credentials, this means that only the issuer B of a credential can read the secret attribute value. This section shows how an issuer can let some trusted partners read secret attributes. Table 2.3 shows a three stage keying architecture: the *private key* is used to sign a credential with a secret attribute and is kept secret by signers. The *protected key* enables the signature verification and access to the secret attribute value and is only distributed to trusted partners of the signer. The *public key* enables the verification of the signature and the set of possible values without revealing the secret value. The new type of key, which we introduced, is said to be a *protected* key: this terminology was chosen by analogy with object oriented programming languages where access to methods can be defined as public, protected, or private. Protected keys are only distributed to trusted partners that are thus able to decrypt attributes. For instance, each employee of a company could be allowed to open credentials signed by coworkers in order to establish trust relationships in a distributed way.

2.5 Trust Establishment Protocol

We now introduce though a generic scenario the trust establishment protocol that is based on the previous building blocks. In this scenario, B is a service provider that issues credentials to entities that interact with him. A is a holder that collects credentials from different entities in order to build a history. TTP is a trusted third party that issued B's keys. B' is a partner of B that knows the protected key of B.

2.5.1 Protocol Description

Before any interaction, B starts u times a *join* protocol with the TTP, u being the number of different values that can be attached to a credential, e.g. with {*0:poor, 1:fair, 2:good*}, $u = 3$. As shown in the right part of Figure 2.2, B knows u secrets $x, x', \ldots$ and receives u membership certificates $(y+1)^d, (y'+1)^d, \ldots$. B also receives a public key certificate $\mathrm{CERT} = \mathrm{SIGN}_{TTP}(K_{P_B}, u)$, which guarantees a set of possible values. Private, protected, and public keys are distributed according to Table 2.3.

When B wants to provide a credential to A, the following exchange occurs: A chooses a random message m, B blindly signs this message with the private key corresponding to the chosen attribute value. A verifies with the public key of B that the signature is correct. The certificate *CERT* is public and defines the set of possible values in the secret attribute.

Subsequently, when A gets in touch with B or B', she shows an unblinded version of the credential and B or B' *opens* the signature to retrieve the secret attribute. The only information available to B is that he is interacting with an entity that was previously tagged as *good*.

Credential	**TTP**	**B**	**B'**	**A**
B's public key $\{n, e, G, g, a\}$, CERT	×	×	×	×
B's protected key $\{y, y', \ldots\}$	×	×	×	-
B's private key $\{x, (y+1)^d, x', \ldots\}$	-	×	-	-
TTP's secret on B $\{p, q, d\}$	×	-	-	-
Verify signature of B and attr $\in$ set	×	×	×	×
Retrieve value of secret attribute	×	×	×	-
Sign credential as B	-	×	-	-
Define set of attribute values	×	-	-	-

Table 2.3: Distribution of secrets among parties

2.5.2 Security Evaluation

This chapter proposed to use group blind signature schemes to define unlinkable secret credentials. Assuming that the group blind signature scheme proposed in [Ram99, CS97] is secure, evaluating the security of our scheme is straightforward.

Proposition 2.1 ***(Credential Integrity)*** *It is infeasible to generate an unlinkable secret credential without knowing the private key of the service provider B.*

The first requirement for unlinkable secret credential scheme presented in this chapter is that credentials can only be issued by the legitimate issuer. Each credential is signed with a group member key corresponding to the public key of B. The group signature scheme ensures that only group members can generate valid signatures. The TTP could generate a new group member key and use it for signing as B. However in this case, B could prove that this TTP misbehaves.

Proposition 2.2 ***(Credential Unlinkability)*** *the issuer B or any other party cannot trace a credential to a user except for a very small number of users.*

The second requirement for unlinkable secret credential scheme presented in this chapter is that it is not possible to link the credential that is blindly created and signed by B and the credential that is shown subsequently to B or B'.

First, the credential m, s is unconditionally independent of m' and s'. By definition, any blind signature fulfills this property.

Next, the attribute value cannot be used to trace all users. It is necessary that the cardinality u of the set of possible attribute values be as small as possible. For instance, defining three different attribute values ($u = 3$) when thousands of entities receive credentials assure the 'average unlinkability' of users. The cardinality of the set does not give a direct information on the entropy of a given attribute value. A malicious environment could thus spot up to $u - 1$ specific users and reserve one attribute value for each one in order to trace them. Even in this case, the unlinkability of all other entities is assured.

Proposition 2.3 ***(Attribute Secrecy)*** *The secret attribute cannot be decrypted by A.*

The third requirement for unlinkable secret credential scheme presented in this chapter is that what B said about A is kept secret.

Group signature schemes ensure that the signature does not reveal any information on the effective signer. It is not possible to find out which member of a group generated a given signature. Because each attribute value is attached to one of the group key it is not feasible to find which key, i.e. which attribute, was used.

Moreover, without knowing the value of the attribute, the credential holder can verify that it is part of a finite set of public attributes. The cardinality of the set of possible attributes is equal to the number of member keys known by B because each key is related to one attribute value. The TTP provides the member keys and certifies the number u of keys. To assure that A cannot generate a new member key even if she acts as multiple group members and knows related secrets and certificates, it is necessary that the group signature scheme be resistant to coalition attacks. The initial *join* protocol of [CS97] has to be replaced by a more secure one. This modification is taken into account in the group blind signature scheme [Ram99].

Proposition 2.4 ***(Attribute Decryption)*** *Only the issuer B and his trusted partners (e.g. B') can retrieve secrets embedded in credentials signed by B.*

The fourth requirement for unlinkable secret credential scheme presented in this chapter is that the service provider B can share information on A with partner B' that knows the protected key of B.

Distributing *protected keys* (y, y', etc.) to partners (e.g. B') does not weaken the scheme. Indeed, partners as well as manager cannot impersonate group members and partners cannot enable covert channels (new members) because they do not have access to TTP's secrets.

2.6 Conclusion

This chapter presents a technique of untraceable secret credentials to establish history-based trust relationships in a privacy-preserving way. Secrecy ensures that positive as well as negative statements can be used in the behavior description attached to an entity. Blind group signature scheme is shown to be a possible mechanism to achieve untraceability.

The principle of unlinkable secret credentials has inherent limitations. Knowing the cardinality of the set of possible attributes is not sufficient to know the entropy of each encrypted attributes. In other words, a given attribute can be used to trace a given user. The 'average untraceability' is ensured by the scheme: only a small subset of the user group can potentially be traced. To ensure unlinkability, it is necessary to have a small set of possible attributes. The attribute can thus define a degree or a rank but cannot define elements of large sets like integers or dates. In chapter 3 we will show another approach that assures untraceability of attributes by enabling selective disclosure of attributes.

Implementing unlinkable secret credentials based on the blind group signature scheme [Ram99] leads to other limitations. Non-transferability is weakly assured. However, it seems realistic to assume that secrecy of credentials makes it impossible to trade them but a better approach would require to link the credential to a valuable secret of its holder. Such an approach requires a minimal trust infrastructure and will be presented in Chapter 3. Another limitation of this scheme is that a trusted third party is required to certify that the number of possible values of secret attributes is restricted. Indeed, knowing the public key is not sufficient to determine the number of group members, i.e. the set of possible attribute values. Whether it is possible to render blind a group signature scheme with a public key depending on the number of members, and thus to do without TTP, is still an open issue.

This chapter and chapter 1 have tackled the difficult problem of achieving non-transferability of credentials without a minimal infrastructure. In the second part of this dissertation, we will investigate solutions that rely on some infrastructure. We will show that a minimal infrastructure is sufficient to overcome restrictions on attributes while preserving untraceability.

Part II

History-Based Trust Establishment:

Establishing Trust while Preserving Privacy

Chapter 3

History-based Signature Scheme

"He said the right thing to that cat. I like his prudence-
He learned it through experience.
Which taught that lesson of maturity:
Mistrust is the mother of security."

– **Jean de la Fontaine** (The Cat and the Old Mouse, III)

The main limitation of the protocol proposed in Chapter 2 is the small cardinality of the set of possible attribute values that makes it unsuitable for describing complex interactions because only simple attributes are supported. This chapter describes another solution relying on credentials with cleartext attributes that can be disclosed in a controlled way. This mechanism allows a full featured history-based trust establishment yet protects the privacy of users.

3.1 Introduction

We introduce a protocol through which an entity can give cryptographic evidence on the history of its past interactions with other parties. This history makes it possible to evaluate the past behavior of that entity, and accordingly regard it more or less trustworthy when performing other interactions. Privacy is an essential requirement for such a protocol that builds up trust at the expense of exposing the intimate behavior of a user: for instance, an electronic discount coupon valid for various locations other than the shopping mall where the coupon was generated should not enable the tracing of clients. The anonymity of the history proving entity and the unlinkability of its interactions thus was a design objective for the history proving protocol detailed here.

This chapter presents a new signature scheme that can be used for anonymously signing and for proving the validity of attributes of the signer. This scheme enables non-interactive signatures as well as challenge-response protocols without the attributes being traceable. This mechanism will be used in Chapter 5 to prove previous interactions when establishing trust.

Verifying the reliability of a piece of information without revealing the identity of its source is becoming an important privacy requirement. Indeed, on one hand, anybody can easily broadcast inaccurate or even deliberately deceptive information as illustrated by urban legends or hoaxes. Author authentication thanks to the signature of the document is frequently used when the author is known by the reader. Indeed, according to the trustworthiness of the author, it is possible to determine whether the document is accurate or misleading. On the other hand, protecting the privacy of signers is necessary. When people are exchanging ideas in a public forum, anonymity may be a requirement in order to be able to state some disturbing fact or even simply so that contributors not be traced based on their opinions. When users have a way to attach comments to surrounding physical objects like a painting in a museum [Ing03], the chance that statistics be made on their interests might simply refrain them from commenting at all.

An unlinkable credential aims at proving that some entity has a given attribute in an unlinkable way. For instance Alice may have to prove her nationality to enter an embassy, reveal her age to enter a movie theater, or prove that she has a driving license. This can be done in a face-to-face interaction, e.g. when asked by a policeman, or remotely, e.g. when accessing a Web service. In both cases, only the necessary attributes should be proven and no information on the identity (or public key) of the user should be revealed. In our scheme, the challenge response protocol is implemented as the signature of a random nonce and relies on the anonymous signature scheme.

The requirements of anonymous signature are partially met by the group membership notion in group signature schemes [CS97] that proves the existence of a relationship with other members of the group while ensuring the anonymity of the signer. This chapter extends the group membership concept and associates embedded attributes within a signature in order to prove that attribute without revealing the identity of the signer.

In an open environment where trust can only be based on the past behavior of a user A, we propose to implement the history as a set of credentials. When signing, A chooses which part of her history (a subset of credentials) she wants to reveal. For instance, a report relating some event can be signed by *a person who was there when this event occurred*; an e-mail can be signed by *an inhabitant of a given district of a town.* Like this, the signature is not based on the identity of the signer anymore but rather on her history. Such a history is defined as a set of the context (time and location), group memberships (reporter, trade unionist), and recommendations (defined by Bob as a trusted party). The signer chooses the accuracy of the attributes she discloses, e.g. someone that can prove that he was in Paris on the 15^{th} of January could choose to sign a document as someone who was in France in January.

3.2 Principle

Users anonymously collect credentials, which are proofs of their activity, and store them as a provable history. In Figure 3.1, a user gets a credential with some attributes. To associate a credential with the user, non-transferable credentials are used, i.e. credentials attached to a valuable secret. Credentials can define group membership, location- and time-stamps, recommendations, etc. As depicted in Figure 3.1, obtaining a credential is performed in three steps:

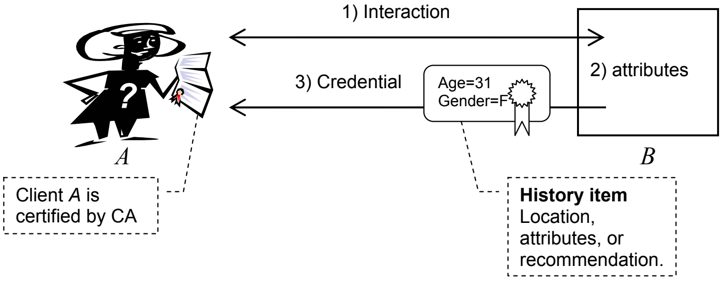


Figure 3.1: Getting history items

1. Some interaction occurs between A and B. This step can require an authentication, a payment, or even a real world protocol (e.g. handshake, graduation ceremony).
2. The party B who is in charge of providing the credential defines the attributes. For instance, an Id-card will contain a name, a surname, a birth date, and a nationality; a bus pass will include a validity end date; and a recommendation will define whether B trusts the holder A.
3. The credential is provided to A that stores it in her history database.

Such a credential can be used for signing a document (Figure 3.2) or during an interactive proof (Figure 3.3).

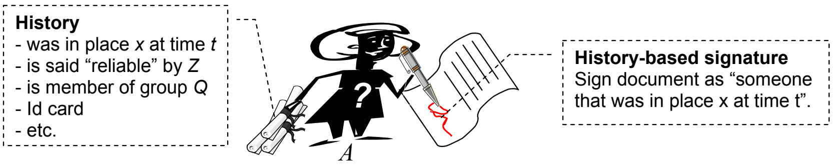

History
- was in place x at time t
- is said "reliable" by Z
- is member of group Q
- Id card
- etc.
A
History-based signature
Sign document as "someone that was in place x at time t".

Figure 3.2: History-based signature

When signing a document, the author A chooses some credentials in her history, modifies them, and signs the document with those credentials. In Figure 3.2, a user is able to prove that she was at location x at time t, that she is said reliable by some entity Z, that she is a member of group Q, and that she has a given name and address (electronic id card). She chooses to sign the document as *someone that was at location x at time t* or as *someone that is older than 18*. The signature does not reveal more information on the signer and it is even impossible to link two signatures of the same signer. To ensure untraceability, it is necessary to avoid being too precise: it is indeed easier to identify a person that signed as having been in a given room at a precise time than to recognize this person based on the knowledge that she was in the building at some time.

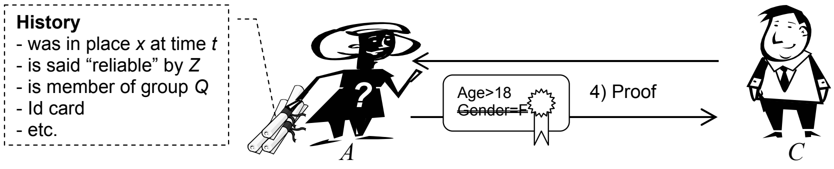


Figure 3.3: Proof of History

Credentials have to fulfill the following requirements to build a provable yet anonymous history:

- *Non-transferability*: credentials should not be transferable to another party.
- *Anonymity*: use of history-based credentials should not reveal the identity of the author.
- *Untraceability*: it is not possible to link different documents signed by a same person even when the same credential is used.

Non-transferability is indeed achieved because credentials can only be used by the owner of some valuable secret (equivalent to the private key in public key infrastructures). This secret is critical and thus will not be transferred to another entity. As a result, credentials cannot be transferred.

3.3 State of the Art: Unlinkable Credentials

Common attribute certificates such as X.509 and SPKI [EFL$^+$99] are based on a public key infrastructure and thus cannot be made unlinkable because the public key of a user can be recognized.

The schemes presented in [CL01, Bra02] already allow for privacy-preserving attribute verification. The target of these schemes is anonymous attribute certificates and untraceable access control. Credentials defined in Idemix [CL01] rely on pseudonyms and thus cannot be used for non-interactive signatures. Credentials defined by Brands [Bra02] weakly assure non-transferability and have to be used as one-time credentials to ensure untraceability. The one-time usage of these credentials also does not suit multiple interactions as required by our scenario.

Table 3.1 compares different approaches with our scheme. All approaches provide a way to prove some attributes without being traceable and thus are types of *unlinkable* credentials. However, in group signature as well as in our scheme, the group manager or CA can trace users. *Selective disclosure* means that the user can choose to reveal a part of the attribute when signing. For instance, a credential asserting that its holder is thirty years old can be used so that it only discloses that its holder is older than eighteen. More complex disclosure schemes are defined in some related work. *Non-transferability* means that a user cannot let another person use one of her credentials without revealing some valuable data. In a public key infrastructure (PKI), the valuable data is the users private key. In Idemix, a model referred as ”all or nothing” (AoN) is introduced so that transferring one credential reveals all pseudonyms and credentials. The *number of use* is another important parameter for history: it is mandatory that credentials can be used multiple times yet ensure unlinkability. The possibility of *combining credentials* when signing is another important issue. Indeed, a document signed by a doctor that has seen a given patient is different from a document that is signed by a doctor and by someone else that have seen this patient. *Non-interactive signature* is necessary when the scheme is used to sign public documents, e.g. web pages, and authorize off-line verification by unknown parties. For instance, Idemix is a pseudonym scheme and thus cannot be used to sign documents. Finally, a scheme *integrated with a distance-bounding protocol* (DBP) is necessary to prove that A is close enough to a location- and time-stamper.

Properties	**Group signatures**	**Brands' credentials**	**Camenisch Idemix**	**History-based Sig**
Unlinkable	×	×	×	×
Unlinkable by CA	-	×	×	-
Selective Disclosure	-	×	×	×
Complex Disclosure Schemes	-	×	×	-
Non-transferable	PKI	Biometrics	AoN or PKI	PKI
Number of Use	∞	1	1 or ∞	∞
Combining Credentials	-	×	×	×
Non-interactive Signature	×	×	-	×
Integration with DBP	?	?	?	×

Table 3.1: Comparison of different schemes

3.4 Protocols

History-based signature is an extension of the group signature scheme described in Section 2.3. Let Alice (A) be the signer: she collects credentials to subsequently prove a history of her interactions. For instance, A holds credentials to prove that she has been in some place. When A is traveling or visiting partners, she collects location stamps. A has credentials to prove her affiliation to a company, her membership to the IEEE computer society, her participation in some project, her membership to a golf club, her citizenship of a particular country, etc. Finally, A can collect recommendations asserting her collaboration with other entities. All those credentials define her provable history. Each credential can be used as a proof during a challenge-response protocol or as an attribute of a signature.

Let us define the following elements for computing that signature: a RSA modulo $n = pq$ where p and q are two large primes, a cyclic group G of order n with generator g, and an element of the multiplicative group $a \in \mathcal{Z}_n^*$.

3.4.1 Zero-Knowledge versus Statistical Zero-Knowledge

In [Sti02], Stinson defines a *Zero-Knowledge Proof of Knowledge* as a protocol that allows some prover A to prove to some verifier B that it knows a secret x without revealing any information on this secret x.

In a proof of knowledge of a double discrete logarithm (see Section 2.3.1) PK$[\alpha \mid z = g^{(a^{\alpha})}]$, the response is $s = r - c \cdot x$ where c is the challenge bit and r is randomly chosen. If the order of $a \in \mathcal{Z}_n^*$ is known, s can be computed modulo this order. But when this order is not known or when a proof of knowledge involves different orders, it is not possible to compute responses modulo. In this case zero-knowledge cannot be achieved and *statistical zero knowledge* is necessary. Statistical zero knowledge proposes to compute s in a larger group in order to make it more difficult to get information on x. A security parameter $\epsilon > 1$ defines this larger group: random parameter r is chosen in $\{2^{\lambda}, \ldots, 2^{\lambda \cdot \epsilon} - 1\}$ where λ defines an upper bound on the length of x, i.e. $0 \leq x < 2^{\lambda}$.

In the remaining of this thesis we use the term *Proof of Knowledge* for a protocol that allows some prover A to prove to some verifier B that it knows a secret x when it is computationally impossible for B to discover any information on x, be it zero-knowledge or statistical zero-knowledge.

3.4.2 Certification

The main difference between this chapter and Part I of this thesis is that here, user A has a valuable secret x. In other words, credentials can be linked to this secret to avoid transferability. We use certification to prove that a secret is valuable. To initiate the system, each entity has to get some certificate proving that it has a valid secret, i.e. a secret linked to its identity. This part is similar to the join protocol of Camenisch's group signature scheme. However, we use a modified version proposed in [AT99, Ram99] because a coalition attack exists against the initial scheme.

A		**CA**
		private: p_{ca}, q_{ca}, d_{ca} public: $n_{ca}, e_{ca}, G_{ca}, g_{ca}, a_{ca}, \lambda_{ca}$
1.1) chooses random secret x' $x' \in_R \{0, 1, ..2^{\lambda_{ca}-1}\}$		
	1.2) $y' = a_{ca}^{x'} \mod n_{ca}$ $\longrightarrow$	
	1.3) $\xi \in_R \{0, 1, .., 2^{\lambda_{ca}-1} - 1\}$ $\longleftarrow$	
1.4) computes $x = x' + \xi$ $y = a_{ca}^{x} \mod n_{ca}$ commits to $z = g_{ca}^{y}$		
	1.5) y, z $\longrightarrow$	
	1.6) $\text{PK}[\alpha \mid y = a_{ca}^{\alpha}]$ $\longleftrightarrow$	
		1.7) verifies $y \overset{?}{=} y' \cdot a_{ca}^{\xi}$
	1.8) $\text{cert}_{1ca} = (y+1)^{d_{ca}} \mod n_{ca}$ $\longleftarrow$	

Table 3.2: Creation and first certification of A's secret x

As depicted in Table 3.2, A generates some secret x with the help of a certification authority (CA) or a group manager. Moreover, A receives a certificate for this secret x: $\text{cert}_{1ca} = (a_{ca}^{x} + 1)^{d_{ca}} \mod n_{ca}$. From that moment on, A is certified and can act anonymously as a member of a group or as an entity certified by a given CA in order to get credentials and build a provable history.

3.4.3 Obtaining Credentials

Once certified, A can visit different entities that will provide proofs of location, proofs of interaction, recommendations, etc. A provable history is a set of such proofs. Table 3.3 shows how A can get a credential from B. The identity of A is not known but B verifies that this entity is certified by some known CA or Group manager. It is always necessary to have some trust relationship with previous signers when providing credentials or when verifying history. In this example, B has to trust CA otherwise the certification protocol has to be done once more. However, when an entity C needs to verify the signature of A on some document, C only has to know B.

A		**B**
private: $x, (a_{ca}^x + 1)^{d_{ca}}$		private: $p_b, q_b, d_b, d_{b_1}, \dots d_{b_k}$ public: $n_b, e_b, e_{b_1}, \dots e_{b_k}$, $G_b, g_b, a_b, b_b, \lambda_b$
2.1) $y_2 = a_b^x \mod n_b$ $\widetilde{g_{ca}} = g_{ca}^r$ for $r \in_R \mathcal{Z}_{n_{ca}}$ $\tilde{z} = \widetilde{g_{ca}}^y$ (i.e. $\tilde{z} = z^r$)		
	2.2) y_2 $\longrightarrow$	
	2.3) pk_2: PK$[\alpha \mid y_2 = a_b^\alpha \wedge \tilde{z} = \widetilde{g_{ca}}^{(a_{ca}^\alpha)}]$ pk_3: PK$[\beta \mid \tilde{z}\widetilde{g_{ca}} = \widetilde{g_{ca}}^{(\beta^{e_{ca}})}]$ $\longleftrightarrow$	
		2.4) $t \in_R \{0, 1, \dots, 2^{\lambda_b} - 1\}$ $\text{cert}_{1b} = (a_b^x + 1)^{d_b}$ $\text{cert}_{2b} = (a_b^x + b_b^t)^{d_h}$ $\text{cert}_{3b} = (b_b^t + 1)^{d_b}$ where $d_h = \prod_{i \in S} d_{b_i}$
	2.5) t, cert_{1b}, cert_{2b}, cert_{3b}, S $\longleftarrow$	

Table 3.3: Obtaining some credential to build history

Two proofs of knowledge are done at step 2.3). The first one proves that y_2 is based on some secret. Combining the both proofs shows that this secret has been certified by CA. Indeed, $\widetilde{g_{ca}}^{(\beta^{e_{ca}})} = \tilde{z}\widetilde{g_{ca}} = \widetilde{g_{ca}}^{(a_{ca}^\alpha)}\widetilde{g_{ca}} = \widetilde{g_{ca}}^{(1+a_{ca}^\alpha)}$ and thus $1 + a_{ca}^\alpha = \beta^{e_{ca}}$. It means that A knows $\beta = (1 + a_{ca}^\alpha)^{d_{ca}}$ that is a certification of α, which is also the discrete logarithm of y_2 to the base a_b. In other words, y_2 has been computed from the same secret x.

At step 2.4) A receives a new credential $\text{cert}_{2b} = (a_b^x + b_b^t)^{d_h} \mod n_b$ from B that will be used to prove some history. b_b as well as a_b are elements of $\mathcal{Z}_{n_b}^*$, x prevents the transferability of credentials, and t is different for each credential to forbid a user from forging attributes by combining multiple credentials (see Section 3.7). The attribute value, be it a location or a recommendation, is defined using a technique that comes from electronic cash: $d_h = \prod_{i \in S} d_{b_i}$ where S is a set that generally defines the amount associated with an e-coin but can be used for other attributes as well. The construction of d_h is described in Section 3.5. Two other credentials can be provided: $\mathrm{cert_{1b}} = (a_b^x + 1)^{d_b} \mod n_b$ is a certification of the secret that can replace $\mathrm{cert_{1ca}}$. To prevent a potential attack (see Section 3.7), we add $\mathrm{cert_{3b}} = (b_b^t + 1)^{d_b} \mod n_b$.

3.4.4 Using History for Signing

This section shows how Alice (A) can sign a document as the holder of a set of credentials. A knows a secret x, the certification of this secret ($\mathrm{cert_{1b}}$), and some credential that is part of her history ($\mathrm{cert_{2b}}$). Using these credentials, she can compute a signature on some message m. A generates a random number $r_1 \in_R \mathcal{Z}_{n_b}$ and computes:

$$\hat{g}_b = g_b^{r_1},\ \hat{z}_2 = \hat{g}_b{}^{y_2},\ \text{and}\ \hat{z}_3 = \hat{g}_b{}^{(b_b^t)}$$

$$spk_1 = \mathrm{SPK}[\alpha \mid \hat{z}_2 = \hat{g}_b{}^{(a_b^\alpha)}](m)$$

$$spk_2 = \mathrm{SPK}[\beta \mid \hat{z}_2\hat{g}_b = \hat{g}_b{}^{(\beta^{e_b})}](m)$$

$$spk_3 = \mathrm{SPK}[\delta \mid \hat{z}_3 = \hat{g}_b{}^{(b_b^\delta)}](m)$$

$$spk_4 = \mathrm{SPK}[\gamma \mid \hat{z}_2\hat{z}_3 = \hat{g}_b{}^{(\gamma^{e_{h'}})}](m) \quad \text{where} \quad e_{h'} = \textstyle\prod_{i \in S'} e_{b_i} \text{ and } S' \subseteq S$$

$$spk_5 = \mathrm{SPK}[\epsilon \mid \hat{z}_3\hat{g}_b = \hat{g}_b{}^{(\epsilon^{e_b})}](m)$$

The signature of message m is $\{spk_1, spk_2, spk_3, spk_4, spk_5, \hat{g}_b, \hat{z}_2, \hat{z}_3, S'\}$. The signatures of knowledge spk_1 and spk_2 prove that the signer knows $\mathrm{cert_{1b}}$: $\beta = (1 + a_b^\alpha)^{d_b} \mod n_b$. The signatures of knowledge spk_1, spk_3 and spk_4 prove that the signer knows $\mathrm{cert'_{2b}}$: $\gamma = (a_b^\alpha + b_b^\delta)^{d_{h'}} \mod n_b$. The signatures of knowledge spk_3 and spk_5 prove that t was generated by B: $\epsilon = (1 + b_b^\delta)^{d_b} \mod n_b$.

When credentials from different entities (e.g. B and CA) must be used together, it is necessary that A generate a random number $r_2 \in_R \mathcal{Z}_{n_{ca}}$ and compute $\hat{g_{ca}} = g_{ca}^{r_2}$ and $\hat{z} = \hat{g_{ca}}^y$. spk_1 and spk_2 are then modified as follows:

$$spk'_1 = \mathrm{SPK}[\alpha \mid \hat{z}_2 = \hat{g}_b{}^{(a_b^\alpha)} \wedge \hat{z} = \hat{g_{ca}}^{(a_{ca}^\alpha)}](m)$$

$$spk'_2 = \mathrm{SPK}[\beta \mid \hat{z}\hat{g_{ca}} = \hat{g_{ca}}^{(\beta^{e_{ca}})}](m)$$

spk'_1 and spk'_2 prove that the signer knows $\mathrm{cert}_{1\mathrm{ca}}$: $\beta = (a_{ca}^{\alpha}+1)^{d_{ca}} \mod n_{ca}$ and spk'_1 proves that $\mathrm{cert}_{1\mathrm{ca}}$ and $\mathrm{cert}_{2\mathrm{b}}$ are linked to the same secret x. spk'_1 is a signature based on a proof of equality of two double discrete logarithms (see Section 3.6.2). The new signature of message m is $\{spk'_1, spk'_2, spk_3, spk_4, spk_5, \hat{g_{ca}}, \hat{z}, \hat{g_b}, \hat{z_2}, \hat{z_3}, S'\}$. Similarly, it is possible to link multiple credentials when signing a document.

3.5 Encoding Attribute Values

In Section 3.4, the user receives $\mathrm{cert}_{2\mathrm{b}}$ and signs with $\mathrm{cert}'_{2\mathrm{b}}$ to hide part of the attributes when signing. This section presents a flexible mechanism for attribute encoding that allows the user to choose the granularity of attributes.

A straightforward solution to define attributes with various levels of granularity would be based on multiple credentials. For instance, a location stamper would provide credentials defining room, building, quarter, town, state, etc. The holder would thus be able to choose the granularity of the proof of location. Unfortunately, combinatorial attributes (e.g. longitude, latitude, or time) with different granularities would lead to the distribution of too many credentials.

3.5.1 Principle

In our scheme, each authority that delivers certificates (time stamper, location stamper, group manager, etc.) has a public key: a RSA modulo (n), and a public set of primes $e_1, \ldots, e_m$ where $\forall i \in \{1, \ldots, m\} \mid \gcd(e_i, \phi(n)) = 1$. Each e_i correspond to a mark whose meaning is public as well. Each authority also has a private key: p, q, and $\{d_1, \ldots, d_m\}$ where $pq = n$ and $\forall i \in \{1, \ldots, m\} \mid e_i \cdot d_i = 1 \mod \phi(n)$.

The signature $SIGN_{(S,n)}(m) = m^{d_h} \mod n$ of a message m, where S is a set of indices defining the attribute and $d_h = \prod_{i \in S} d_i$, can then be transformed into a signature $SIGN_{(S',n)}(m) = m^{d_{h'}} \mod n$, where S' is a subset of S and $d_{h'} = \prod_{i \in S'} d_i$. The attribute value is coded as S corresponding to the indices of the bits equal to one in the binary representation of this attribute. This signature based on set S can be reduced to any subset $S' \subseteq S$:

$$\begin{aligned} SIGN_{(S',n)}(m) &= \left(SIGN_{(S,n)}(m)\right)^{\left(\prod_{j \in \{S \setminus S'\}} e_j\right)} \\ &= m^{\left(\prod_{i \in S} d_i \cdot \prod_{j \in \{S \setminus S'\}} e_j\right)} = m^{\left(\prod_{i \in S'} d_i\right)} \mod n \end{aligned}$$

An entity that received some credential $\mathrm{cert}_{\mathrm{2b}}$ is thus able to compute a derived $\mathrm{cert}'_{\mathrm{2b}}$ and to sign a document with this new credential.

$$\mathrm{cert}'_{\mathrm{2b}} = (\mathrm{cert}_{\mathrm{2b}})^{\prod_{j\in\{S\setminus S'\}} e_j} = \left(\left(a_b^x + b_b^t\right)^{\prod_{i\in S} d_i}\right)^{\prod_{j\in\{S\setminus S'\}} e_j} = \left(a_b^x + b_b^t\right)^{\prod_{i\in S'} d_i}$$

This technique ensures that part of the signed attributes can be modified. For instance, the decimal attribute value $v = 13_d$ is equivalent to the binary string 01101_b and can be encoded as $S = \{4, 3, 1\}$, i.e. 4^{th}, 3^{rd}, and 1^{st} bits set to one. $d_h = d_4 \cdot d_3 \cdot d_1 \mod \phi(n)$. Knowing $\{e_i \mid i \in S\}$, the following transformations are possible: $S' \in \{\{4, 3, 1\}; \{3, 1\}; \{4, 3\}; \{4, 1\}; \{4\}; \{3\}; \{1\}\}$ and thus $v' \in \{13, 5, 12, 9, 8, 4, 1\}$. Any bit i equal to one can be replaced by a zero (by using e_i) but any bit j equal to zero cannot be replaced by a one (because d_j is private).

3.5.2 Possible Codes

Choosing different ways to encode data enables to define which transformations of the attribute values are authorized:

- *greater-or-equal*: values are encoded so that they can only be reduced. For instance, $v = 13_d \rightarrow 01101_b \rightarrow S = \{1, 3, 4\}$. Because each bit equal to one can be replaced by zero, value 01101_b can be transformed into 01100_b, 01001_b, 01000_b, 00101_b, 00100_b, or 00001_b, i.e. $v' \in \{13, 12, 9, 8, 5, 4, 1\}$.

  An attribute defined with this code can only be reduced. For instance, someone able to show a capability with the age attribute set to eighteen at least, can be assumed older than eighteen.

- *less-or-equal*: values are encoded so that they can only be increased. For instance, $v = 13_d \rightarrow 10010_b \rightarrow S = \{2, 5\}$. It can be transformed into 10010_b, 10000_b, or 00010_b, i.e. $v' \in \{13, 15, 29\}$.

  An attribute defined with this code can only be reduced. For instance, a capability with the price attribute set to hundred at most ensures a negotiated price below one hundred euros.

- *unary more-or-equal*: the problem with binary encoding is that it cannot be reduced to any value. For instance, $7_d = 111_b$ can be shown as 7, 6, 5, 4, 3, 2, or 1 but $6_d = 110_b$ can only be shown as 6, 4, or 2. This limitation can be solved by using a binary representation of unary counting: $v = 6_d = 111111_u \rightarrow 0111111_b \rightarrow S = \{1, 2, 3, 4, 5, 6\}$ can be shown as $v' \in \{6, 5, 4, 3, 2, 1\}$. The overhead is important (l bits data is encoded with 2^l bits) and the unary representation thus has to be restricted to small values.

- *unary less-or-equal*: a similar approach can be used for less-or-equal as well: $v = 2_d \rightarrow 1111100_b \rightarrow S = \{3, 4, 5, 6, 7\}$ can be transformed into $v' \in \{2, 3, 4, 5, 6\}$.

- *frozen*: values are encoded so that they cannot be changed. In this case, the number of bits has to be larger: l bits become $l + \lfloor \log_2(l) \rfloor + 1$ bits. For instance, $13_d \rightarrow 0001101_b, c = 100_b \rightarrow 0001101|100_b \rightarrow S = \{7, 6, 4, 3\}$. The checksum c represents the number of bits equal to zero, any modification of the value increases the number of zeroes but the checksum can only be decreased. It is not possible to change such frozen values.

- *blocks*: data are cut into blocks. Each block is encoded with one of the previous schemes.

Selective disclosure is an important feature for preserving the privacy of users. Indeed, even when credentials are unlinkable, showing too precise attributes can reveal a lot of information about a user. For instance, assuming a system that generates less than one certificate a minute, multiple shows of an unlinkable credential with a validity end equal to ”26/05/2005 09:07” could be linked if this validity end is visible. Similarly, a credential proving that some entity has been in a given place at a given time ”13:04, 15/10/2004, 43.6265^o, -007.0470^o” is too precise and enables a server to trace users.

This problem can be avoided by only revealing necessary information. When the validity is checked, it is possible to prove that the validity is still valid instead of revealing the validity end (e.g. ”xx/xx/2005 xx:xx”. If the system has to know whether the user was in Sophia Antipolis this year, ”xx:xx, xx/xx/2004, 43.6xxx^o, -007.0xxx^o” is enough. Similarly, a company can qualify customers as *Platinum, Gold, or Silver* and the customer can prove that she is a least gold; a state can provide digital Id cards to citizen to certify gender, name; birth date and the citizen can prove that she is older than eighteen. In any case, the ability of selecting which attribute is displayed is very important to protect privacy.

3.6 Proof of Knowledge

This section defines the new proof of knowledge PK$[\alpha \mid y_2 = a_b^\alpha \wedge \tilde{z} = \tilde{g_{ca}}^{(a_{ca}^\alpha)}]$ that is necessary when obtaining a credential (see Section 3.4.3) and the new signature based on a proof of knowledge SPK$[\alpha \mid \hat{z_2} = \hat{g_b}^{(a_b^\alpha)} \wedge \hat{z} = \hat{g_{ca}}^{(a_{ca}^\alpha)}](m)$ that is necessary when using a credential (see Section 3.4.4).

3.6.1 Proof of Equality of a Log and a Double Log

The execution of this proof of knowledge is as follows: PK$[\alpha \mid y = a_1^{\alpha} \wedge z = g_2^{(a_2^{\alpha})}]$ Given $y = a_1^x$ and $z = g_2^{(a_2^x)}$, the interactions below are performed t times as long as the verification step is performed successfully.

1. P $\longrightarrow$ V : $w_1 = a_1^r$ and $w_2 = g_2^{(a_2^r)}$ where $r \in_R \{2^{\lambda}, \dots 2^{\epsilon\cdot\lambda} - 1\}$
2. V $\longrightarrow$ P : $c \in_R \{0,1\}$
3. P $\longrightarrow$ V : $s = r - cx$
4. Verification :
   if $c = 0$: $w_1 \overset{?}{=} a_1^s$ and $w_2 \overset{?}{=} g_2^{(a_2^s)}$
   if $c = 1$: $w_1 \overset{?}{=} y \cdot a_1^s$ and $w_2 \overset{?}{=} z^{(a_2^s)}$

Indeed, if $c = 1$:
$$y \cdot a_1^s = a_1^x \cdot a_1^s = a_1^{x+r-x} = w_1 \text{ and}$$
$$z^{(a_2^s)} = \left(g_2^{(a_2^x)}\right)^{(a_2^s)} = g_2^{(a_2^x)\cdot(a_2^s)} = g_2^{(a_2^{x+s})} = w_2$$

3.6.2 Signature Based on a Proof of Equality of Double Log

History-based signature uses a signature based on a proof of equality of two double discrete logarithms.

$$\text{SPK}_l[\alpha \mid y_1 = g_1^{(a_1^{\alpha})} \ \wedge \ \cdots \ \wedge \ y_k = g_k^{(a_k^{\alpha})}](m)$$

where l is a security parameter. The signature is an $l+1$ tuple $(c, s_1, \dots, s_l)$ satisfying the equation

$$c = \mathcal{H}\left(m\|k\|\{y_1 \dots y_k\}\|\{g_1 \dots g_k\}\|\{a_1 \dots a_k\}\|\{P_{1,1} \dots P_{1,l}\}\| \cdots \|\{P_{k,1} \dots P_{k,l}\}\right)$$

$$\text{where } P_{i,j} = \begin{cases} g_i^{(a_i^{s_j})} & \text{if } c[j] = 0 \\ y_i^{(a_i^{s_j})} & \text{otherwise} \end{cases}$$

The signature can be computed as follows:

1. For all $j \in \{0, 1, \ldots, l-1\}$, generate random $r_j \in_R \{2^\lambda, \ldots, 2^{\epsilon \cdot \lambda} - 1\}$
2. For all $i \in \{0, 1, \ldots, k-1\}$ and for all $j \in \{0, 1, \ldots, l-1\}$, set $P_{i,j} = g_i^{(a_i^{r_j})}$
3. Compute $c = \mathcal{H}\left(m \| k \| \{y_1 \ldots y_k\} \| \{g_1 \ldots g_k\} \| \{a_1 \ldots a_k\} \| \{P_{1,1} \ldots P_{1,l}\} \| \cdots\right)$
4. Set $s_j = \begin{cases} r_j & \text{if } c[j] = 0 \\ r_j - x & \text{otherwise} \end{cases}$

The verification works as follows for all $i \in \{0, 1, \ldots, k-1\}$:

$$\text{if } c[j] = 0\text{:} \quad P_{i,j} = g_i^{(a_i^{r_j})} \overset{?}{=} g_i^{(a_i^{s_j})}$$

$$\text{if } c[j] = 1\text{:} \quad P_{i,j} = g_i^{(a_i^{r_j})} \overset{?}{=} y_i^{(a_i^{s_j})}$$

$$\text{Indeed, if } c = 1\text{:} \quad y_i^{(a_i^{s_j})} = \left(g_i^{(a_i^x)}\right)^{(a_i^{s_j})} = g_i^{(a_i^x \cdot a_i^{s_j})} = g_i^{(a_i^{x+s_j})} = g_i^{(a_i^{r_j})}$$

It is not possible to reduce s_j because the order of $a_1 \in \mathcal{Z}_{n_1}^*$ is generally different from the order of $a_2 \in \mathcal{Z}_{n_2}^*$.

3.7 Security Evaluation

The security of the scheme is based on the assumptions that the discrete logarithm, the double discrete logarithm and the roots of discrete logarithm problems are hard problems. In addition it is based on the security of Schnorr and RSA signature schemes and on the additional assumption of [CS97] that computing membership certificates is hard.

Our proposal is based on the group signature scheme of [CS97], whose join protocol is subject to a collusion attack [AT99]. Modifications suggested in [Ram99] that prevent this attack have been taken into account (see Table 3.2).

The first requirement for the history-based signature scheme presented in this chapter is that credentials can only be issued by the legitimate issuer B:

Requirement 3.1 ***(Credential unforgeability)*** - *It is infeasible to generate an unlinkable credential without knowing the private key of the service provider B.*

Proposition 3.1 *The unlinkable credential scheme is conformant to Requirement 3.1.*

Proof: In order to encode attribute values, a set of different e_i and d_i are used with the same modulo n. However, the common modulus attack does not apply here because d_i's are kept secret and known by a single entity as with the standard RSA. Because there are multiple valid signatures for a given message, this scheme seems to make brute force attacks, which aim at creating a valid signature for a given message, easier: an attacker can choose a message m and a random $d_R \in_R \mathcal{Z}_n^*$ and compute a signature $m^{d_R} \mod n$. If e_i and d_i are defined for $i \in \{1, \ldots, k\}$, there are $2^k - 1$ valid sets S and thus $2^k - 1$ possible $d = \prod_{i \in S} d_i$. The probability that a random d_R be acceptable is $2^k - 1$ times higher than with standard RSA (where $k = 1$). However, even if the number of valid signatures for a given message increases, an attacker has to find out the set S (i.e. e_R) corresponding to the randomly chosen signature. In other words, the attacker has to test for all S, whether $m \stackrel{?}{=} (m^{d_R})^{\prod_{i \in S} e_i} \mod n$. There are $2^k - 1$ possible sets S to check and the security of this scheme thus is equivalent to RSA.

In some cases, the signature scheme can allow combining attributes of two credentials in order to create a new one: naive credentials $(a^x + 1)^{d_{h_1}}$ and $(a^x + 1)^{d_{h_2}}$ could be used to create $(a^x + 1)^{d_{h'}}$ where $S' \subseteq S_1 \cup S_2$. If S_1 states that Alice was present from 8 a.m. to 10 a.m. and S_2 states that she was present from 4 p.m. to 6 p.m., it is necessary to forbid that Alice could create S' stating that she was present from 8 a.m. to 6 p.m. To avoid this attack, a unique secret t is associated with each credential. Hence $(a^x + b^{t_1})^{d_{h_1}}$ cannot be combined with $(a^x + b^{t_2})^{d_{h_2}}$. □

The second requirement for the unlinkable credential scheme presented in this chapter is that it is not possible to link the signature of A on some message with the identity of A and that it is not possible to link different signatures done with the same credential. Note that the scheme presented here is an extension of group signatures and thus assures unlinkability from any party but the credential issuer (see Table 3.1).

Requirement 3.2 ***(Signature anonymity and unlinkability)*** - *The anonymity of signers and the unlinkability of signatures are assured.*

Proposition 3.2 *The unlinkable credential scheme is conformant to Requirement 3.2.*

Proof: Linking two signatures $\{spk_1, spk_2, spk_3, spk_4, spk_5, \hat{g_b}, \hat{z_2}, \hat{z_3}\}$ and $\{spk'_1, spk'_2, spk'_3, spk'_4, spk'_5, \hat{g_b}', \hat{z_2}', \hat{z_3}'\}$, i.e., deciding whether these signatures have been issued by the same user A, is only possible by deciding whether $\log_{\hat{g_b}}(\hat{z_2}) = \log_{\hat{g_b}'}(\hat{z_2}')$ or deciding whether $\log_{\hat{g_b}}(\hat{z_3}) = \log_{\hat{g_b}'}(\hat{z_3}')$, which is equivalent to solving the discrete logarithm problem that is considered a hard problem. The signatures generated by the credential scheme are therefore unlinkable. The party B, be it a certification authority or a group manager, certifies the secret x of A and knows the identity of A. Other parties have no information on the holder of x and A can thus sign anonymously. □

As in the solutions proposed in Chapters 1 and 2 where the same credential is used in two signatures, the attribute revealed should be different or imprecise enough in order to prevent linkability.

The third requirement for the history-based signature scheme presented in this chapter is that a signature cannot be generated without holding the correct credentials:

Requirement 3.3 ***(Signature unforgeability)*** - *It is infeasible to sign with respect to some attribute without holding a credential with this attribute.*

Proposition 3.3 *The unlinkable credential scheme is conformant to Requirement 3.3.*

Proof: The signature of knowledge spk_1 proves that the signer knows his secret, spk_3 proves that the signer knows a credential's secret, and spk_4 proves that the signer knows a credential corresponding to both secrets. That is, spk_1 and spk_3 respectively show that

$$\hat{z}_2 = \hat{g}^{(a^\alpha)} \quad \text{and} \quad \hat{z}_3 = \hat{g}^{(b^\delta)}$$

and therefore:

$$\hat{z}_2 \hat{z}_3 = \hat{g}^{(a^\alpha + b^\delta)}$$

Whereby α and δ are known by the signer. In addition, spk_4 proves that

$$(a^\alpha + b^\delta) = \gamma^{e_{h'}}$$

for some γ that the signer knows. Under the hardness assumption on the unforgeability of credentials, this can only happen if the signer received a credential $(a^x + b^t)^{d_{h'}}$. □

The fourth requirement for history-based signature scheme presented in this chapter is that credentials cannot be transferred:

Requirement 3.4 ***(Credential non-transferability)*** - *Credentials are strongly linked to a valuable secret of the holder and thus cannot be transferred.*

Proposition 3.4 *The unlinkable credential scheme is conformant to Requirement 3.4.*

Proof: Each credential is linked to the valuable secret x of its holder. However, even when the signature of a message cannot be forged, a desirable goal is to be able to assure that it is not possible to find another message with the same signature. Violation of this property with our protocol would require the generation of two pairs (x, t) and (x', t') so that $a^x + b^t = a^{x'} + b^{t'} \mod n$. In order to prevent transferability based on such a generation of equivalent pairs, $cert_{3b}$ and spk_5 were included in the protocol. Computing (x', t') from a credential based on (x, t) would thus require computing $x' = \log_a(a^x + b^t - b^{t'})$ which is equivalent to solving the discrete logarithm problem. Our protocol thus assures that the credential received as a proof of context or as a recommendation cannot be transferred. □

A proof that the generation of suitable pairs is equivalent to a difficult problem (e.g. the discrete logarithm problem) would allow for important simplifications of the history-based signature scheme.

3.8 Conclusion

This chapter introduced a *history-based signature* scheme that makes it possible to sign data with one's history. In this scheme, signers collect unlinkable credentials in order to build a provable history. This scheme preserves the privacy of users and makes a large variety of attributes possible for defining trust: recommendations, contextual or location-related proofs, reputation, and even hierarchical relationships.

This scheme can be useful in different situations. For instance, any visitor of a pervasive computing museum could be allowed to attach digital comments to painting and to read comments of previous visitors. Notes could be signed by an *art critic that visited the museum one week ago.* In this example, we assume that the critic received some credential to prove that he is an expert (e.g. electronic diploma when completing study) and that he can prove that he visited the gallery. Each visitor will filter the numerous notes according to parameters defining trustworthiness as he envisions it, e.g. is the note authored by an art critic, at the museum, or is the author recommended by the museum. The authors of notes have a guarantee that they cannot be traced by any visitor. In another situation, the signature of an article written by a journalist could require one credential to prove that the author was where the event occurred and another credential to prove that he is a reporter.

In chapter 5, we will show how this scheme can be used to build trust relationships while preserving privacy.

Chapter 4

Distance-Bounding Proof of Knowledge

"I'll put you through into Looking-glass House. How would you like that?"

– **Lewis Carroll** (Through the Looking-Glass)

This chapter focuses on the link between the physical world of artifacts and the virtual world of logical entities. We show that authentication in pervasive computing requires proving that an artifact, i.e. a physical entity, knows a secret. The *distance-bounding proof of knowledge* paradigm is introduced and a possible implementation is proposed. Chapter 5 will combine this mechanism with *history-based signature* in order to define history-based trust establishment.

4.1 Introduction

In daily interactions, location provides privileges. For instance, a person has to be present in a room to be able to use the switch that turns on the light of this room. It means that this service (lightning) is restricted according to some user context. The widespread of wireless communications renders the verification of location difficult. This chapter studies how location, proximity, and physical interactions can be asserted and chapter 5 will use those contextual assertions to establish and enforce trust relationships.

Ubiquitous computing [Wei91], context aware computing [SDA99], and augmented reality [Ing03] have been topic of research since the 1990s. Those topics are related by a desire to merge the physical world with the virtual world of electronic services and applications (see Figure 4.1). One important effect of this merge is that applications need to know the physical location of artifacts so that they can record them and report them to us [IU97, BK, McC01]: what lab bench was I standing by when I prepared these tissue samples? How should our search-and-rescue team move to quickly locate all the avalanche victims? Can I automatically display this chart on the video projector I am standing next to? Who is the owner of this object? And so on.

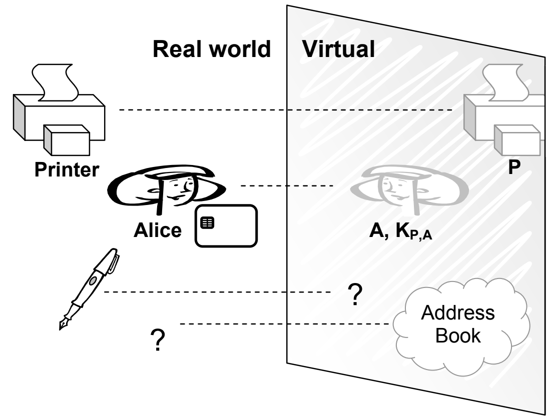


Figure 4.1: Links between artifacts and their virtual representations

Taking physical location into account has a strong impact on security. Indeed, is it possible to authenticate a physical object: is it possible to certify and verify attributes of a given artifact? How is it possible to prove one's location to enable context-based access control? How is it possible to prove that some entity was in some place? In general, authentication is based on the knowledge of a secret, like a private key. When location matters, authentication means verifying that a physical entity knows a secret. In other words, new mechanisms are necessary to verify that a secret is locally known.

In this chapter, we focus on applications combining physical proximity and cryptographic identification schemes. More precisely, we are interested in a quite recurring family of applications in cryptography where a prover tries to convince a verifier of some assertion related to his private key. The assertion in our case is that the prover is within a certain physical distance. Brands and Chaum [BC93] were the first to specifically address this problem. They introduced *distance-bounding protocols* that allow to determine a practical upper bound on the distance between two communicating entities. The verification is performed by timing the delay between sending out a challenge bit and receiving back the corresponding response bit, the number of challenge-response interactions being determined by a chosen security parameter. This approach is feasible if and only if the protocol uses very short messages (one bit) on a dedicated communication channel (e.g. wired or infrared communication medium) and if nearly no computation is required during each challenge-response exchange (few logical operations). These features enable round-trip times of few-nanoseconds.

The protocols given in [BC93] address the case where an intruder sits between a legitimate prover and a verifier and succeeds in performing the distance verification process.

Here, we provide an extension of those protocols. Our solution addresses the case where the prover and the intruder collaborate to cheat the verifier.

4.2 Problem Statement: Authentication in Pervasive Computing

In this section, we first study the requirements of authentication in pervasive computing. Next, we describe three types of attacks that we tackle further on in this chapter, namely *distance frauds*, *mafia frauds*, and *terrorist frauds*.

4.2.1 Redefining Authentication

Authentication aims at proving the validity of a claim to a verifier. This claim is often the identity of the prover but it can be the fact that she is part of a group or even more generic attributes. The authentication of human beings relies on:

- What the prover is (biometrics)
- What the prover knows (password, PIN code, etc.)
- What the prover has (tokens like smart cards, etc.)

In this dissertation we only focus on the latter case. We assume that each user carries a trusted personal device that may be tamper-resistant and that protects its owner's secrets (secret key, private key, etc.). Authenticating this artifact is equivalent to authenticating the user. Using the personal device can require biometric or password-based access control.

In pervasive computing, authentication has to be redefined because the number of artifacts and the lack of trust infrastructure makes identity-based authentication meaningless [CGRZ03] and because the authentication of a physical object does matter when services are provided by artifacts. It is necessary to have a way to certify attributes of artifacts: the identity of the manufacturer, the guarantee that an artifact was controlled by a trusted party, the identity of the owner, or any other characteristic should be associated to the artifact in a tamper-proof manner.

For the sake of clarity, let us take a very simple example: Alice is in a public place with a PDA. The PDA contains confidential data which she wants to print out. Alice wants to use a wireless link to send data to a chosen printer. Authenticating the chosen printer, i.e. proving some of its attributes can be split into two parts:

1. Linking the chosen artifact to a public key (e.g. the digest or fingerprint of the artifact's public key might be written on this artifact).
2. Linking the public key of the artifact to some attributes (e.g. an attribute certificate might be generated by the artifact owner).

The former part links physical and virtual worlds and is subject to new types of attacks. Those attacks and solutions to define artifact authentication will be discussed in the remaining of this chapter.

4.2.2 New Attacks against Authentication

In this section we will present attacks against authentication in pervasive computing as outlined above. We only focus on *real-time attacks*, i.e. attacks that need exchanges with a legitimate prover. In other words, we assume that the private key of the prover cannot be stolen by an intruder and that this private key is valuable so that the prover will not disclose it to another party. The attack scheme is always the same: the verifier V, which is in front of an intruder I, runs an authentication protocol, and gets convinced that he is in front of a prover P.

This attack can seem similar to *man-in-the-middle* attacks. In a man-in-the-middle attack, the intruder generally runs a security protocol with both legitimate parties. For instance, P and V want to create a shared secret using Diffie-Hellman key agreement. Unfortunately P is running the security protocol with the attacker I and this attacker I is also running the protocol with V. As a result, instead of creating a secure channel between P and V, two secure channels are created between P and I and between I and V. I can observe and/or modify any information transiting between P and V. Fortunately, this type of attack can be easily avoided when authentication is possible. In our case, we focus on a similar problem: the *mafia fraud* attacks.

Classical Mafia Fraud Attack

The mafia fraud also involves a middleperson (intruder) but contrary to man-in-the-middle attacks, this intruder does not perform any cryptographic operation based on the security protocol and only acts as a proxy that forwards challenges and responses. The attack was first introduced in [Des88].

A classical example is given in Figure 4.1. A prover P wants to access some server I that requires authentication, for instance movies are only delivered if the customer can prove how old she is. The attack works as follows: the server I is owned by the mafia and, instead of providing a random challenge, it forwards a challenge required for accessing

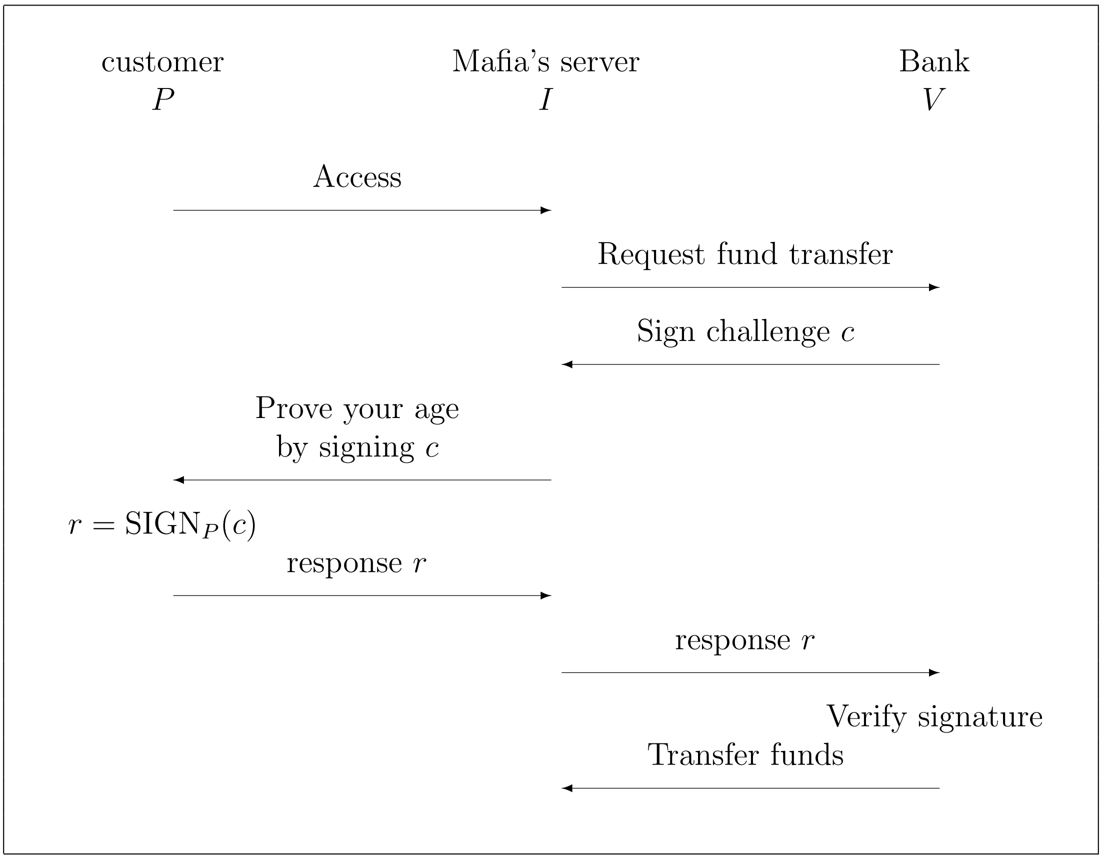


Table 4.1: Mafia-fraud attack using access control protocol

another server V. By signing the challenge sent by V and forwarded by I, P lets I access V. This attack seems naïve because it relies on the following assumptions:

- P is using the same asymmetric key pair to sign documents and during challenge-response protocols.
- The challenges and responses are not specific to the protocol, e.g. the challenge is a random number and the response is a signature on this nonce.

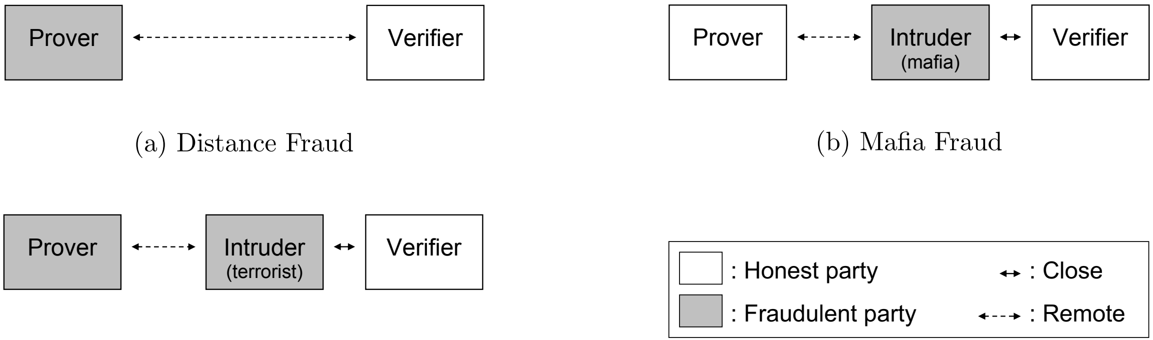
In other words, to avoid mafia frauds, it is mandatory to use different keys for different purposes or to use more complex challenges.

Attacks in Pervasive Computing

Mafia frauds are more realistic and even easier to implement in pervasive computing when the problem is to be sure that the verifier is in front of the certified prover that might offer physical interfaces (e.g. keyboard, display) or deliver "physical goods" (e.g. cash, printout). Brands and Chaum [BC93] were the first to specifically address this problem. The principle of this attack is very simple. Let us imagine that Alice carries some wireless token that automatically responds to challenges sent by the environment. When arriving at home Alice is authenticated by some sensor and the front door is opened. When Alice sits in front of her terminal, she is automatically logged in. Some pairing mechanism ensures that only authorized challenges are accepted. In other words, the token will not respond to a challenge coming from an unknown source. We assume that the pairing is secure and that man-in-the-middle attacks are not possible. We can even assume that this token is tamper resistant or physically protected by its owner who carries or wears it. However, a very simple attack can be performed: Eve is in front of the door of Alice, Eve receives some challenge c that is directly forwarded to an accomplice Edgar who is waiting close enough to Alice. Edgar sends this challenge to Alice's token that returns a correct response which is forwarded to the sensor with the help of Eve. The door is opened and Eve can enter the house. The couple Eve-Edgar acts as an intruder (or proxy). This attack is realistic if it is possible to forward the challenge-response protocol without being detected. This happens in numerous situations because the round trip time is often long [BB04a]: the signature is time-consuming especially when computed by limited tokens (e.g. smart cards) and the protocol used for local communication adds delay. Moreover due to the heterogeneous hardware that might be used by the prover, the verifier generally tolerates the worst round-trip time.

4.2.3 Definitions

Distance-bounding protocols have to take into account the three real-time frauds that are depicted in Figure 4.2. These frauds can be applied in zero-knowledge or minimal disclosure identification schemes.



(a) Distance Fraud

(b) Mafia Fraud

(c) Terrorist Fraud

Figure 4.2: Three Real-Time Frauds

The first fraud is called the *distance fraud* and is defined in the following (Figure 4.2-a)

Definition 4.1 (Distance Fraud) *In the* distance fraud*, two parties are involved: one of them (the verifier V) is not aware of the fraud that is going on, the other one (the fraudulent prover P) performs the fraud. The fraud enables P to convince V of a wrong statement related to its physical distance to V.*

The distance fraud has been addressed in [BC93]. This fraud consists in the following: if there is no relationship between the challenge bits and the response bits during the distance verification and if the prover P is able to know at which times the challenge bits are sent by the verifier V, he can make V compute a wrong upper-bound of his physical distance to V by sending out the response bits at the correct time before receiving the challenge bit, regardless of his physical distance to V.

The second fraud is called the *mafia fraud* (Figure 4.2-b):

Definition 4.2 (Mafia Fraud) *In the* mafia fraud*, three parties are involved: two of them (the honest prover P and the verifier V) are not aware of the fraud that is going on, the third party (the intruder I or mafia) performs the fraud. The fraud enables I to convince V of an assertion related to the private key of P.*

The mafia fraud has been first described in [Des88]. In this fraud, the intruder I is usually modeled as a couple $\{\bar{P}, \bar{V}\}$ where $\bar{P}$ is a dishonest prover interacting with the honest verifier V and where $\bar{V}$ is a dishonest verifier interacting with the honest prover P. Thanks to the collaboration of $\bar{V}$, the fraud enables $\bar{P}$ to convince V of an assertion related to the private key of P. The assertion is that the prover is within a certain physical distance. This fraud was also called *Mig-in-the-middle attack* in [And01].

The third fraud is called the *terrorist fraud* (Figure 4.2-c):

Definition 4.3 (Terrorist Fraud) *In the* terrorist fraud*, three parties are involved, one of them (the verifier V) is not aware of the fraud going on, the two others (the dishonest prover P and the intruder I or terrorist) collaborate to perform the fraud. The help of P enables I to convince V of an assertion related to the private key of P.*

The terrorist fraud was first described in [Des88]. In this fraud, the prover and the intruder collaborate to perform the fraud whereas in the mafia fraud the intruder is the only entity that performs the fraud. Note that the prevention of terrorist frauds assures the prevention of mafia frauds.

4.2.4 Attack Examples

Nowadays, wireless technology is easy to integrate and might be used to mount mafia frauds against deployed services. For instance, such an attack could be performed against point of sale terminals even if terminals and credit cards are tamper-resistant and certified. We think that such attacks will spread quickly when numerous daily interactions involving micro payments and access control will happen in pervasive computing environments.

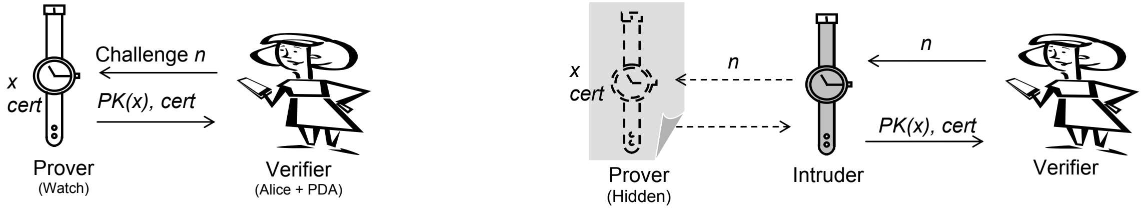
This section presents two types of attack against systems that do not address artifact authentication properly. First, a dummy artifact acts as a proxy in order to impersonate the original artifact that is hidden. This attack aims at verifying attributes or transferring rights to a wrong object. Next, a malicious visitor acts as a proxy in order to get location stamps for remote peers.

Artifact Impersonation

The first type of physical proxy attack aims at impersonating some artifact. We assume that artifacts are able to prove that they know some secret (e.g. a private key) and that this secret is protected by some tamper-resistant hardware:

- A watch could embed a tamper-resistant core (e.g. a smart card) that protects its private key (see Figure 4.3(a)). The watch's public key is certified by the manufacturer and the public key of the manufacturer is known by the verifier.

- An automatic teller machine (ATM) is shaped as a safe and protected by alarms and thus cannot be tampered with. Each ATM has a private key and a public key, which is certified by the bank. The verifier knows the public key of the bank.



(a) Expected Interaction

(b) Attack

Figure 4.3: Attack against the authentication of an artifact: the attribute certificate displayed by the PDA is not related to the chosen artifact (watch).

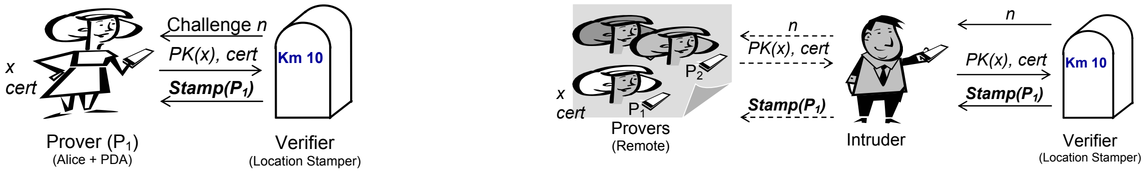
Impersonating the device means that, during the interaction, a dummy device is presented to the user. In Figure 4.3(b), a user wants to buy some valuable artifact (e.g. a Swiss watch). To fight against forgery, the manufacturer has embedded a chip that enables digital certification. A malicious seller presents a faked watch that acts as a proxy and forwards the challenge-response protocol to the real watch that is hidden in the pocket of the malicious seller. In another situation, a user checks whether he is in front of a certified ATM, the dummy ATM acts as a proxy and forward the challenge responses. When the user requests 100$, the cash is delivered in another place where the attacker is waiting.

This attack is meaningful when physical entities (printer, ATM, watch, etc.) offer physical services (printout of confidential data, cash delivery, changing ownership of consumer electronics, etc.). This attack is meaningless in purely virtual interactions (accessing a server, obtaining a map, etc.).

User Impersonation and P2P Privilege Exchange

The second example of attack aims at impersonating a user (e.g. Alice) so that a verifier will think that she is physically present when only a colluding partner is present and acts as a proxy. We assume that persons are able to prove that they know some secret (e.g. their private key) and that this secret is valuable or not available, i.e. generated and protected by a tamper-resistant module. Different attack cases can be proposed:

- Some building access control mechanism using sensors or card readers can easily be misled and open the door to an attacker.



(a) Expected Interaction

(b) Attack

Figure 4.4: Attack against location service. A malicious visitor acts as a proxy and forwards the challenges and responses of peers in order to let them get discount without visiting the shop.

- Deploying terminals to provide location- and time-stamps could be interesting to let users prove that they were somewhere at some time (see Figure 4.4(a)). Such a system could be defeated by intruders acting as proxies for someone else. Alice could thus prove that she went to some place without having moved (see Figure 4.4(b)).
- Discounts could be offered to people that frequently visit some shop or to people that participated to some event. The attack would result in a discount offered to friends of such people, defeating the commercial policy of the shop.

Figure 4.4(b) shows how provers P_1 and P_2 (peers of the intruder) can receive location stamps without visiting the delivery place. Tokens of malicious users are not tampered with.

4.3 State of the Art: How to Prove One's Location

Examples of Section 4.2.4 show in what attacks against a combination of physical and virtual entities is counterintuitive. Each time we presented this work [BR02] and [BR03b], it led to interesting exchanges, numerous questions, and multiple proposals. Unfortunately, careful verifications have often shown security holes approaches that seemed promising. This section presents a state of the art listing possible approaches and explaining why a large part of them do not fulfill our threat model or are not practical enough.

There are many techniques to get one's location or to locate devices [HB01], the global positioning system (GPS) being one of the most widely deployed. Cell-phones operators can locate phones knowing the cell and using triangulation [GW98]. In that manner they can propose location service to their customers. Active Badge [WHFG92] has been the first indoor location system. Badges used infrared to emit identifiers that were collected by fixed infrared sensors and transmitted to a server. Active Bat [HHS$^+$99] uses ultrasound time of flight to provide more accurate physical positioning than Active Badge. Users and objects carry Active Bat tags. A controller sends a short range radio signal that synchronizes sensors and makes the bat emit an ultrasonic pulse that is detected by surrounding sensors. Each sensor computes the distance to the bat and sends it to a server that find out the location of the bat. Cricket [PCB00] proposes a passive technique for indoor location similar to GPS: a set of ultrasound emitters sends signals to objects that perform their own triangulation to know their location. Radar [BP00] is a building-wide tracking system based on WLAN. Base stations use the signal strength of wireless devices to evaluate their distance. Magnetic tracker generates axial magnetic-field pulses so that receivers can compute their orientation and location. Finally, authors of [GLHB03] propose the combination of different location techniques and works on location in ad-hoc and sensor networks [BHE00] explore distributed approaches.

Unfortunately, the approaches presented in the previous paragraph do not address security. Indeed such techniques enable the location of an honest user but do not tackle malicious users pretending to be at a specific location: location systems can be attacked, e.g. GPS signal can be spoofed. Moreover, knowing its location is not sufficient to prove it. The following subsections present different approaches that can be used to prove one's location. Table 4.2 summarizes the evaluation of such approaches.

4.3.1 Location-Limited Channels

Location-limited channels (or constrained channels [KZS02]) aim at exchanging some secret between two physical entities and thus assure the proximity of two devices. An obvious implementation is to have a physical contact or a wire between both artifacts. This mechanism can be used during the whole interaction, e.g. smart card plugged in a point of sale terminal, or during some initialization or pairing protocol. In other words, two devices can be physically linked for exchanging their public keys or for sharing a secret key. Thanks to this initial key exchange, confidentiality, integrity, and authentication can be ensured on any wireless communication channel. [SA99] proposed to initially share a secret through a contact channel in order to secure subsequent interaction based on wireless channels. This model was extended to address peer-to-peer interactions [Sta00].

This basic concept has been extended in order to go without any wire or contact. Active badges were already using infrared as a local channel. In [BSSW02], IrDA is used as a location-limited channel and physical contact is also seen as an option. IrDA is interesting because it is short-range and directional enough to select a specific device. Limited-range radio channels allow local exchanges of secrets [CN02, CD00b].

This scheme works only when the attacker is not physically present. It can only protect a system against distance frauds.

Mechanism Description	**Disable distance fraud**	**Disable mafia fraud**	**Disable terrorist fraud**	**Mutual authent.**	**User-friendly**
General Purpose Location: no security consideration (e.g. GPS, Cricket)	-	-	-	-	×
Location Limited Channels: exchange secret trough local channel (e.g. wire, IrDA)	×	-	-	×	(×)
Context Sharing: create a secret from contextual data (e.g. local beacon, movement patern)	×	-	-	×	×
Proximity Evaluation: evaluate proximity thanks to physical effects (e.g. sound propagation, signal strength)	×	-	-	×	×
System Observation: trace devices and/or users in order to check the coherence of there actions (cannot be in two places at the same time, etc.).	(×)	(×)	(×)	-	-
Certification of Location: trusted third party certifies that some appliances is fixed at some location.	×	×	×	-	-
Isolation: isolates the artifact when verifying that it knows some secret.	×	×	×	-	-
Unforgeable Channel: uses channels that are difficult to forward without knowing a secret.	×	(×)	-	×	×
Quantum Cryptography: based on quantum effect that forbid a proxy to forward photons.	×	×	-	×	×
Radio Frequency ToF: measure round trip time at the bounded speed of light.	×	(×)	-	×	(×)
Distance-bounding Proof of Knowledge: the solution presented in this chapter.	×	×	×	×	(×)

Table 4.2: Comparison of location mechanisms. ×: fulfill the requirement; (×): partially fulfill the requirement; $-$: do not fulfill the requirement.

4.3.2 Context Sharing

A straightforward extension of location limited channels is to use some contextual data to initiate the key exchange: all devices within some space can observe the environment in order to share some local knowledge. For instance, a beacon can broadcast some secret to all devices within a room. This secret can be used to bootstrap some secure group among entities that are present.

In [DM96] the signal of Global Positioning System (GPS) satellite is used. The GPS signal of a set of up to twelve satellites is used to compute a local secret and exchange keys. Users can be involved to force some context sharing. Bluetooth, in its most secure configuration requires the user to enter a PIN into both devices in order to secure their first communication. The pairing mechanism of Smart-Its Friends [HMS$^+$01] also involves the user who has to shake artifacts together in order to create a common movement pattern that is subsequently used to bootstrap the security of communications.

Some secrets, especially GPS signal, can be computed by remote attackers and thus do not protect the initialization. However, it is possible to provide secure enough contextual secrets that are not available to attackers. This approach prevents distance frauds and makes mafia frauds more difficult to mount.

4.3.3 Proximity Evaluation

Different solutions exist to evaluate the distance between two devices. This subsection describes various approaches and subsection 4.3.8 focuses on solutions relying on the speed of light that ensure the bounded round-trip time of an electromagnetic or electric signal.

Sound and especially ultra-sound is interesting to measure distance because it is slow enough to authorize computation without reducing the accuracy of the measure. Authors of [SSW03] go one step forward: the verifier sends a radio challenge to the prover that responds with an ultrasound response. The processing delay due to the computation of the response is shorter than the propagation time of the ultrasound media and only slightly reduces the location precision. Sound-based approaches cannot protect against physically present attackers and thus can only prevent distance frauds.

4.3.4 System Observation

Keeping a trace of all interactions involving a user can allow real-time verification. For instance, a user cannot enter a building twice without leaving or a user cannot get a proof of location in Europe and a few minutes later get another proof of location in the USA. Such a system could be seen as a real-world Intrusion Detection System (IDS). Sensors, be they embedded into the floor [OA00] or cameras [KHM$^+$00] could also be involved. Such a scheme could be deployed in a restricted perimeter (military base, airport, etc.) but is very difficult to use for daily interactions. Moreover it is incompatible with privacy concerns.

4.3.5 Certification of Fixed Location

Another approach to verify proximity is to compare the location of two devices. The verifier is mobile and knows his location (e.g. GPS, cell phone network). The prover is fixed and is certified as standing in a given place.

The verifier obtains his own location and gets the certificate of the server (e.g. ATM) that should be in front of him. The verifier checks the certificate, computes the distance and direction of the server, and displays it. The user can verify that he is in front of an appliance certified by some entity, e.g. a bank.

This solution avoids physical proxy attacks but imposes that only fixed devices can be verified. In other words, the prover can prove his location to the verifier but the verifier cannot prove his location. This scheme can only work when one device is fixed.

4.3.6 Isolation

A widely deployed solution to check whether a physical entity knows a secret is to isolate this artifact during a challenge-response protocol. Using a Faraday cage during identification is not new [BBD$^+$91] and is used in ATM to check that a smart card embeds a private key.

This solution prevents distance frauds, mafia frauds as well as terrorist frauds. However, it is difficult to deploy, not user-friendly, and it does not allow mutual authentication.

4.3.7 Unforgeable Channel

The goal of unforgeable channels is to use communication media that are difficult to create without knowing some secret. For instance, channel hopping or RF watermarking makes it difficult to transfer data necessary to create the signal to another place. In Appendix A, we propose the use of quantum cryptography as an ultimate channel protection scheme.

Authors of [AS02] propose a new implementation of *Identification between Friend or Foe* (IFF). Since IFF systems were introduced during World War II, they become today an essential part of military equipment, be it a ship, an aircraft, or even a soldier. Authors propose to use channel hopping: tamper resistant devices secretly choose the next channel to use. An intruder has to listen to and forward all channels because it does not know which channel is listened by the impersonated device. When numerous channels are available to a large set of devices, the bandwidth necessary to forward potential messages becomes unaffordable.

Another approach is to hide some geographical information within data packets (packet leashes [HPJ03]) or even within the radio frequency signal form (RF watermarking). A proxy thus cannot forward the signal when each communicating artifact knows its own location.

Channel hopping seems very promising because it could be integrated within standard communication systems (GSM, UMTS, Bluetooth, 802.11, etc.). Unfortunately, when two given artifacts are communicating, it is easy to detect which channels are used. Channel hopping seems more appropriate when a large infrastructure exists, like in a cell-phone network. In this case, the proxy cannot detect which channel is used by the infrastructure to communicate with the real device and thus has to forward all channels used by the infrastructure.

This scheme protects against distance frauds and the solution proposed in [AS02] can prevent mafia frauds as well when it is not possible to identify communication sources.

4.3.8 Radio Frequency ToF

Measuring the time of flight (ToF) of an electromagnetic or electric signal assures that cheaters can only pretend being farther by adding delays. Indeed, the speed of light is an upper bound for the round-trip time. This approach assures security properties that cannot be achieved with sound-based distance measurement schemes but very short delays and very precise time measurements are required to obtain precise proximity information. Indeed, one meter accuracy implies responding within a few nanoseconds and thus cannot be done through standard communication channels nor rely on cryptography [WF]. The solution proposed in [BC93] uses very short challenges and responses (one bit) and only Boolean operations for verifying the distance between prover and verifier. Cryptographic functions are combined with this measurement scheme in order to prove that some entity is in the vicinity. This scheme prevents both distance and mafia frauds. More details on this approach are given in the next section and the remainder of this chapter extends this scheme.

4.4 Distance Bounding Protocol

This section discusses why contact-based distance bounding protocols have been chosen for the authentication of artifacts. Table 4.2 summarizes the different approaches proposed in the previous section.

Only *isolation* and *distance-bounding protocols* are secure and precise enough to defeat an intruder who is in front of the verifier (i.e. mafia and terrorist frauds). Isolation is not user-friendly enough and cannot ensure mutual authentication. We thus focus on an extension of the distance-bounding protocols proposed in [BC93]. In our solution, we keep the initial constraints: single bit exchanges and no cryptography during the challenge-response bits exchange.

We introduce *distance-bounding proof of knowledge* protocols that are adequate combinations of distance-bounding protocols [BC93], bit commitment schemes [PHS03] and zero-knowledge proof of knowledge protocols. Our proposal prevents the three frauds described above.

4.4.1 Principle

This section describes the work of Brands and Chaum [BC93] that addresses distance and mafia frauds but that lets terrorist frauds as an open issue. In Section 4.5, we will present our scheme that addresses distance, mafia, and terrorist frauds.

Distance-bounding protocols allow determining a practical upper bound on the distance between two communicating entities. Distance measurement is associated with cryptographic properties in order to prove that a secret is known within a physical area. The measurement is performed by timing the delay between sending out a challenge bit and receiving back the corresponding response bit. The number of challenge-response interactions being determined by a chosen security parameter. This approach is feasible if and only if the protocol is very simple:

- Asymmetric cryptography, symmetric cryptography and even hash functions cannot be used during the protocol (at least during the rapid exchange).
- The messages exchanged have to be as short as possible, i.e. one bit long.
- Dedicated hardware has to be used to measure the round trip time: no application-level operation can be performed during the rapid exchange.

Based on those assumptions, we proposed a *fast challenge-response protocol* in [BR02] and reached round-trip times of few-nanoseconds with a prototype. Indeed, using fast logical gates, we were able to measure 2 ns RTT that is 30 cm of maximum distance (i.e. 60 cm round trip). However, we discovered shortly thereafter that Brands and Chaum had presented a *distance bounding protocol* with very similar properties in [BC93].

Verifier V wants to check that the prover P (a given artifact) has some properties. P can show a certificate signed by a trusted certification authority (CA) which certifies that the owner of the public key K_{P_P} has some properties. The distance-bounding protocol is used to verify that the entity knowing the private key K_{S_P} is close enough, i.e. this entity is indeed the selected artifact.

One-bit challenges and one-bit responses being used, the probability of a successful attack during a single round is high: 1/2. Using m rounds ensures that the probability of successful attack is 2^{-m}. Table 4.3 shows how artifact V can measure the distance to artifact P.

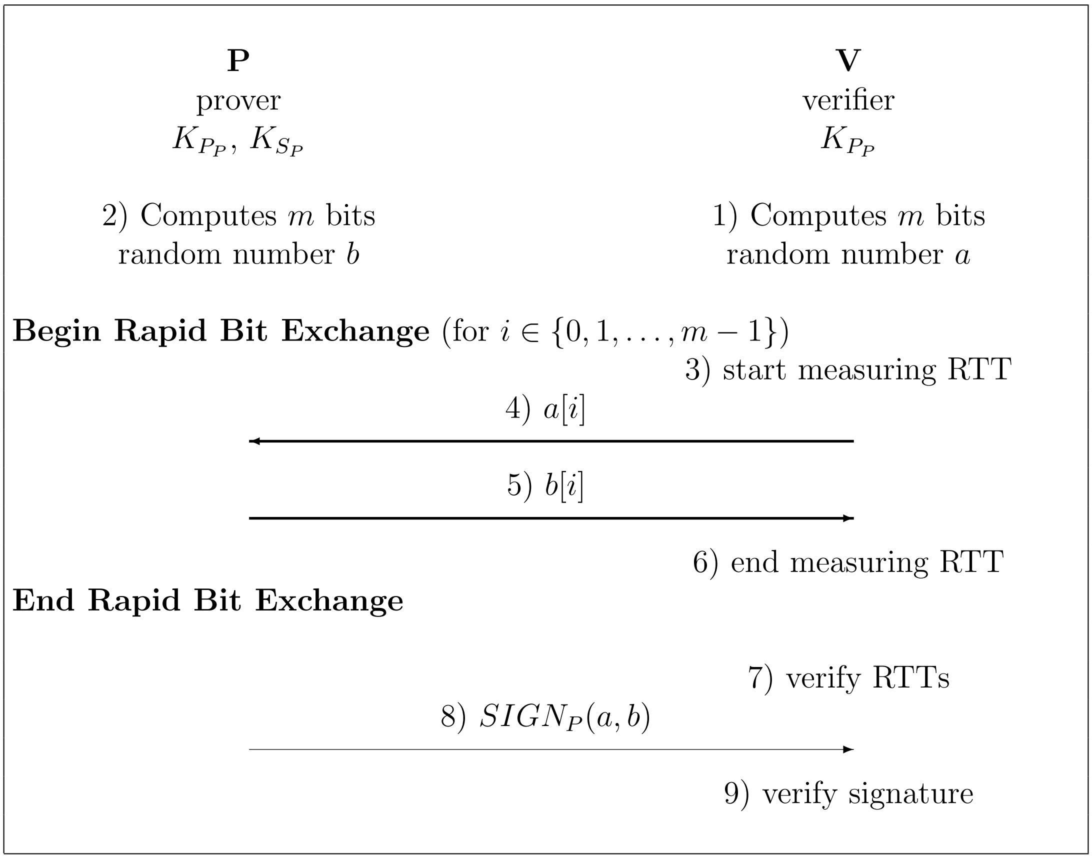

P
prover
K_{P_P}, K_{S_P}
V
verifier
K_{P_P}
2) Computes m bits random number b
1) Computes m bits random number a
Begin Rapid Bit Exchange (for $i \in \{0, 1, \ldots, m-1\}$)
3) start measuring RTT
4) $a[i]$
5) $b[i]$
6) end measuring RTT
End Rapid Bit Exchange
7) verify RTTs
8) $SIGN_P(a, b)$
9) verify signature

Table 4.3: Basic principle of Chaum's distance bounding protocols

In steps 1 and 2, both artifacts V and P generate m random bits (respectively a and b). Next m rapid bit exchanges occur. During rounds $0 \ldots m-1$, the dedicated interface (bold arrows) is used and the response time (few nanoseconds) is measured by V in order to evaluate the distance to P.

The protocol proves that someone knowing a secret b is present. The signature of P on $\{a, b\}$ proves that P knows the secret and that it is present. This assumption is only right when the prover does not misbehave, i.e. does not collude with an intruder. In other words, this protocol prevents mafia frauds and distance frauds when a bit commitment is used [BC93], but does not prevent terrorist frauds.

4.4.2 Implementation Constraints

Figure 4.5 presents the hardware architecture required to deploy distance-bounding protocols. Any involved artifact (i.e. P and V), be it a smart card or a workstation, offer computation (P_C and V_C), communication facilities (label a), and specific interface hardware (P_H and V_H). Interface hardware ensures the fast exchange of one-bit messages (label c). The interface hardware is very simple and based on a few fast logical gates.

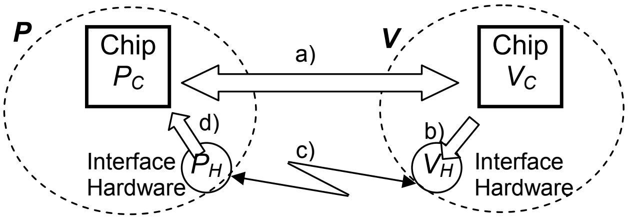


Figure 4.5: Dedicated hardware for distance bounding protocols

Different parameters have to be taken into account when implementing distance-bounding protocols. First, such protocols rely on a large number of simple rounds at each of which the probability of a successful attack is quite high. Such a protocol is thus difficult to be rendered fault-tolerant and has to be implemented on a noiseless channel. Broadcast media such as radio are thus seemingly out of question for implementing these protocols.

In addition to these technical problems, radio communications do not offer a way to precisely select a (physical) artifact, which is essential for the user who wants to authenticate a precise device and not any device that can listen and answer to his requests. Contact based approaches are better in this respect. Alternately, a directional technique like the use of a laser may allow contacting wirelessly a precise artifact. It should be noted however that the laser performs two different tasks at the same time. It first selects a precise artifact and measures its physical distance, like the plugging of a socket in a contact based approach. It is then used to implement a distance-bounding protocol, therefore checking that the logical entity is effectively embodied in that artifact through the measurement of the time taken to transmit a secret. Performing these two tasks suggests a difficult technological integration.

Finally, mutual authentication is often required and, in this case, the selection process has to be bi-directional. Again, defining a wireless approach based on lasers seems difficult in that case. The easiest solution is to use contact based approaches, be they electrical or optical. Moreover, contact-based approaches are less subject to data losses and denial of service (DoS) attacks.

4.5 Tackling Terrorist Frauds

In this section, we present a general scheme that contains the basic building blocks of *distance-bounding proof of knowledge* protocols. The scheme will be denoted DBPK. It is an extension of the protocol of Table 4.3 with specific constraints on challenge and response bits to prevent distance, mafia, and terrorist frauds.

4.5.1 Description

The DBPK protocol is depicted in Table 4.4. This scheme relies on a set of system-wide settings that have to be performed before the execution of any interaction between the prover and the verifier. Besides the cryptosystem's public parameters, these global settings require the prover to have a valuable private key and a certificate on the corresponding public key. That is, before any interaction with the verifier, the prover holds a private key $x \in \{0,1\}^m$ whose importance is so high that the prover cannot reveal it to any other party. In addition, the prover holds a certificate (generated by a globally trusted authority) on its public key $y = \Gamma(x)$.

The first stage of the DBPK protocol is called the *Bit Commitment* stage. During this stage the prover first picks a random symmetric key $k \in_R \{0,1\}^m$ and uses it to encrypt her private key x according to a publicly known symmetric key encryption method $\mathcal{E} : \{0,1\}^m \to \{0,1\}^m$. This leads to $e = \mathcal{E}_k(x) \in \{0,1\}^m$. Note that as in every encryption scheme, only the knowledge of both e and k allows to compute the private key $x = \mathcal{D}_k(e)$. Once the encryption performed, the prover commits to each bit of both k and e according to a secure bit commitment scheme *commit*. For each bit $k[i]$ (resp. $e[i]$), a string v_i (resp. v'_i) is randomly chosen by the prover to construct the commitment blob $c_{(k,i)}$ (resp. $c_{(e,i)}$).

Once the *Bit Commitment* stage is completed, the actual distance-bounding interactions are executed during the *Distance-Bounding* stage. Basically, m interactions are performed between the prover and the verifier. In the i^{th} interaction, the prover releases either $k[i]$ or $e[i]$ depending on whether the challenge bit is equal to 0 or to 1. Note that $k[i]$ (resp. $e[i]$) denotes the i^{th} bit in the binary representation of k (resp. e) where $k[0]$ (resp. $e[0]$) is the least significant bit of k (resp. e).

After the execution of the m challenge-response bit exchanges, the prover opens the commitments on the released bits of k and e. The *Commitment Opening* stage consists

P Prover		**V** Verifier
private key x public key $y = \Gamma(x)$		
Bit Commitments		

secret key $k \in_R \mathcal{K}$
$m = \lceil log_2(\vert\mathcal{K}\vert) \rceil$, $\mathcal{M} = \{0, \ldots, m-1\}$
$e = \mathcal{E}_k(x) \in \{0,1\}^m$
for all $i \in \mathcal{M}$ $\quad v_i, v'_i \in_R \{0,1\}^*$
for all $i \in \mathcal{M}$ $\quad c_{(k,i)} = commit(k[i], v_i)$
for all $i \in \mathcal{M}$ $\quad c_{(e,i)} = commit(e[i], v'_i)$		

	for all $i \in \mathcal{M}$ $\quad c_{(k,i)}, c_{(e,i)}$ $\longrightarrow$	
Distance-Bounding (for all $i \in \mathcal{M}$)		
	$a_i \in_R \{0,1\}$ $\longleftarrow$	
$b_i = k[i]$ if $a_i = 0$ $b_i = e[i]$ if $a_i = 1$		
	$b_i \in \{0,1\}$ $\longrightarrow$	
Commitment Opening		
	for all $i \in \mathcal{M}$ v_i (if $a_i = 0$) $\quad v'_i$ (if $a_i = 1$) $\longrightarrow$	
		$c_{(k,i)} \overset{?}{=} commit(b_i, v_i)$ if $a_i = 0$ $c_{(e,i)} \overset{?}{=} commit(b_i, v'_i)$ if $a_i = 1$
Proof of knowledge		
		$\{c_{(k,i)}, c_{(e,i)}\}_{0 \le i \le m-1} \mapsto z = \Omega(x, v)$
	$PK[(\alpha, \beta) : z = \Omega(\alpha, \beta) \wedge y = \Gamma(\alpha)]$ $\longleftrightarrow$	

Table 4.4: A general scheme for $DBPK[\alpha : y = \Gamma(\alpha)]$

on sending the string v_i if $k[i]$ has been released and v'_i otherwise. Note that only half the bits of k and e are released to the verifier. This should not allow the verifier to get any significant information about the valuable private key x. In the case where the verification of $c_{(k,i)}$ (resp. $c_{(e,i)}$) fails, the verifier sends back an error notification of the form $\text{error}_k(i)$ (resp. $\text{error}_e(i)$).

The last step in the DBPK protocol is the *Proof of Knowledge* stage. During this stage, the prover convinces the verifier in a zero-knowledge interaction that she is the party who performed the three previously described stages. That is, the prover proves that she has generated the different commitments, that the generated commitments correspond to a unique private key, and that this private key corresponds to the public key y that is used by the verifier to authenticate the prover. Note that before the proof of knowledge process can be performed, the verifier must compute $z = \Omega(x, v)$ where v is known only by the prover. As z depends on and only on the commitments on the bits of k and e, it may even be computed just after the *Bit Commitment* stage. The proof of knowledge we use is denoted $PK[(\alpha, \beta) : z = \Omega(\alpha, \beta) \wedge y = \Gamma(\alpha)]$ where the Greek small letters denote the quantity the knowledge of which is being proved, while all other parameters are known to the verifier. The functions Ω, Γ, and *commit* are adequately chosen to meet our security requirements, namely the prevention of the distance, mafia, and terrorist frauds. More details about the needed properties are given in Section 4.7.

4.5.2 Sketch of Security Properties

To sum up, we point out, in the following, the major principles behind the general scheme described above. We define by $DBPK[\alpha : y = \Gamma(\alpha)]$ the distance-bounding proof of knowledge of a secret x so that $y = \Gamma(x)$. The principle of the scheme can be sketched as follows:

(1) Distance bounding phase implies that someone knowing k' and e' is close.

(2) Bit commitments on bits of k and e can be transformed into z that is the result of a collision-free one-way function applied to the decryption of e: $z = \Omega(\mathcal{D}_k(e), v')$.

(3) Opening bit commitments shows that $k' = k$ and $e' = e$.

(4) Proof of knowledge shows that z is the result of a collision-free one-way function applied to x: $z = \Omega(x, v)$

And thus:

(4),(2): Because it is not possible to have the same result z when the collision-free one-way function is applied to two different values, $x = \mathcal{D}_k(e)$.

(4),(2),(3): $x = \mathcal{D}_k(e) = \mathcal{D}_{k'}(e')$.

(4),(2),(3),(1): Someone knowing x is close.

4.6 Distance-bounding Proof of Discrete Log

This section presents specific instantiation of DBPK: distance-bounding proof of knowledge of a discrete logarithm, which consists of exactly the same building blocks of the DBPK protocol. The described protocol will be denoted DBPK-Log $= DBPK[\alpha : y = g^{\alpha}]$.

The two first phases of the DBPK-Log protocol are global settings. In the *Initialization* stage, a certification authority (CA) provides the public parameters of the system.

4.6.1 Initialization

CA sets up the system's global parameters

- CA chooses a large enough strong prime p, i.e. there exists a large enough prime q such that $p = 2q + 1$
- CA chooses a generator g of $\mathcal{Z}_p^*$
- CA chooses an element $h \in_R \mathcal{Z}_p^*$

The randomly chosen element h will be used by the commitment scheme. The only requirement is that neither of the prover and the verifier knows $\log_g(h)$. This can be achieved either by letting the trusted authority generate this element or by making the prover and the verifier jointly generate h. The two alternatives rely on the intractability of the *discrete logarithm* problem [MVO96].

In the *Registration* stage, a user chooses a private key and registers at the trust authority so to get a certificate on the corresponding public key.

4.6.2 Registration

The following steps are taken by P to get a certified public key corresponding to a valuable private key

- P selects an odd secret $x \in_R \mathcal{Z}_{p-1} \setminus \{q\}$, then computes $y = g^x$. The public key of P is y and his private key is x
- P registers his public key with CA so CA publishes a certificate on this public key

Assuming that the global settings have been performed, the actual distance-bounding proof of knowledge protocol can be performed by the prover P and the verifier V. In the *Bit Commitments* stage, P chooses a randomization factor u, generates a random key k, and uses this key to encrypt the randomized private key ux. Then, P performs a secure commitment on each bit of the key k and encryption e.

4.6.3 Bit Commitments

The following steps are performed

- P chooses $u \in_R \{1, \ldots, p-2\}$, then sends u to V
- P chooses $k \in_R \mathcal{Z}_{p-1}$ and let $e = ux - k \mod p-1$
- For all $i \in \{0, \ldots, m-1\}$ where $m = \lceil log_2(p) \rceil$, P chooses $v_{k,i}$, $v_{e,i} \in_R \mathcal{Z}_{p-1}$, computes $c_{k,i} = g^{k[i]} \cdot h^{v_{k,i}}$ and $c_{e,i} = g^{e[i]} \cdot h^{v_{e,i}}$, then sends $c_{k,i}$ and $c_{e,i}$ to V

Once the verifier V receives all the commitment blobs corresponding to the bits of k and e, the *Distance-Bounding* stage can start: a set of fast single bit challenge-response interactions is performed. A challenge corresponds to a bit chosen randomly by V while a response corresponds either to a bit of k or to a bit of e.

4.6.4 Distance-Bounding

The following interactions are performed m times and P reveals half bits of k and e. For all $i \in \{0, \ldots, m-1\}$,

- V sends a challenge bit $a_i \in_R \{0, 1\}$ to P
- P immediately sends the response bit $b_i = \bar{a}_i k[i] + a_i e[i]$ to V

At the end of the *Distance-Bounding* stage, the verifier V is able to compute an upper-bound on the distance to P. In order to be sure that P holds the secrets k and e, the prover P opens, during the *Commitment Opening* stage, the commitments on the bits of k and e that have been released during the *Distance-Bounding* stage.

4.6.5 Commitment Opening

The commitments of the released bits are opened. If all the checks hold, all the bit commitments on k and e are accepted, otherwise they are rejected and an error message is sent back

- For all $i \in \{0, \ldots, m-1\}$, P sends $\bar{a}_i v_{k,i} + a_i v_{e,i}$ to V
- For all $i \in \{0, \ldots, m-1\}$, V performs the following verification:
$\bar{a}_i c_{k,i} + a_i c_{e,i} \overset{?}{=} g^{\bar{a}_i k[i] + a_i e[i]} \cdot h^{\bar{a}_i v_{k,i} + a_i v_{e,i}}$

The *Proof of Knowledge* allows the verifier V to be sure that the sum of the secrets k and e is equal to the randomization of the valuable private key of the prover P. From the bit commitments, V can compute:

$$\begin{aligned}
z &= \textstyle\prod_{i=0}^{m-1} \left(c_{k,i} \cdot c_{e,i}\right)^{2^i} = \prod_{i=0}^{m-1} \left(g^{k[i]} h^{v_{k,i}} \cdot g^{e[i]} h^{v_{e,i}}\right)^{2^i} \\
&= \textstyle\prod_{i=0}^{m-1} \left(g^{k[i]+e[i]}\right)^{2^i} \cdot \prod_{i=0}^{m-1} \left(h^{v_{k,i}+v_{e,i}}\right)^{2^i} \\
&= \textstyle\prod_{i=0}^{m-1} \left(g^{2^i k[i] + 2^i e[i]}\right) \cdot \prod_{i=0}^{m-1} \left(h^{2^i v_{k,i} + 2^i v_{e,i}}\right) \\
&= g^{\sum_{i=0}^{m-1} \left(2^i \cdot k[i] + 2^i \cdot e[i]\right)} \cdot h^{\sum_{i=0}^{m-1} \left(2^i \cdot (v_{k,i} + v_{e,i})\right)} = g^{k+e} \cdot h^v = g^{u \cdot x} \cdot h^v \mod p
\end{aligned}$$

Note that V is able to compute z as soon as all the commitments on the bits of k and e are received and that u is public.

4.6.6 Proof of Knowledge

Given $z = g^{u \cdot x} \cdot h^v$, the following proof of knowledge is performed by P and V: $PK[(\alpha, \beta) : z = g^{u\alpha} h^{\beta} \wedge y = g^{\alpha}]$.

Here is a possible execution of this proof of knowledge: $PK[(\alpha, \beta) : z = g^{u\alpha} h^{\beta} \wedge y = g^{\alpha}]$ Given $z = g^{ux} \cdot h^v$ and $y = g^x$, the interactions below are performed t times as long as the verification step is performed successfully.

1. P $\longrightarrow$ V : $w_1 = g^{ur_1} \cdot h^{r_2}$ and $w_2 = g^{r_1}$ where $r_1, r_2 \in_R \mathcal{Z}_{p-1}$
2. V $\longrightarrow$ P : $c \in_R \{0, 1\}$
3. P $\longrightarrow$ V : $s_1 = r_1 - cx$ and $s_2 = r_2 - cv$

4. Verification :
   if $c = 0$: $w_1 \overset{?}{=} g^{us_1} \cdot h^{s_2}$ and $w_2 \overset{?}{=} g^{s_1}$
   if $c = 1$: $w_1 \overset{?}{=} z \cdot g^{us_1} \cdot h^{s_2}$ and $w_2 \overset{?}{=} y \cdot g^{s_1}$

4.7 Security Analysis

In this section, we discuss the relevant security properties of the DBPK-Log protocol. First, we show that our protocol prevents distance, mafia, and terrorist frauds. Next, the security properties of the encryption scheme that is used to hide the prover's private key are studied.

4.7.1 Preventing Distance, Mafia, and Terrorist Frauds

The first security requirement for our distance-bounding proof of knowledge protocol is a correct computation of an upper-bound of the distance between the prover and the verifier. This requirement is already achieved in the DBPK general scheme:

Proposition 4.1 *If the DBPK protocol is performed correctly, then the distance fraud has a negligible probability of success.*

Proof: Assume that the prover P knows at which times the verifier V will send out bit challenges. In this case, she can convince V that she is nearby by sending out the bit response b_i at the correct time before he receives the bit a_i. The probability that P sends correct responses to V before receiving the challenges is equal to:

$$p = \prod_{i=1}^{m} (P[b_i = k[i] | a_i = 0] + P[b_i = e[i] | a_i = 1]) = 2^{-m}$$

□

In Proposition 4.1, the correct execution of the protocol means that each party performs exactly and correctly the actions specified in the different steps of the protocol. The DBPK-Log protocol is an implementation of the DBPK protocol and thus:

Proposition 4.2 *If the DBPK-Log protocol is performed correctly, then the distance fraud has a negligible probability of success.*

Using the same notations of Section 4.5, we introduce the three following properties.

Property 4.1 *Let $\Gamma : \{0,1\}^* \rightarrow \{0,1\}^*$ be the function such that $y = \Gamma(x)$, then the following holds:*

- *Given y, it is hard to find x such that $y = \Gamma(x)$.*
- *It is hard to find $x \neq x'$ such that $\Gamma(x) = \Gamma(x')$.*

Property 4.2 *Let $\Omega : \{0,1\}^* \times \{0,1\}^* \rightarrow \{0,1\}^*$ be the function such that $z = \Omega(x, v)$, then the following holds:*

- *Knowing z and Ω, it is hard to find (x, v).*
- *It is hard to find $(x, v) \neq (x', v')$ such that $\Omega(x, v) = \Omega(x', v')$.*

Property 4.3 *Let $\mathcal{E} : \{0,1\}^m \times \{0,1\}^m \rightarrow \{0,1\}^m$ be the function such that $e = \mathcal{E}_k(x)$. then the following holds:*

(a) $\mathcal{E}$ is an encryption scheme: knowing e and $\mathcal{E}$, it is hard to find x without knowing k; and given e and k, $x = \mathcal{D}_k(e)$ is efficiently computable.

(b) Given either $k[i]$ or $e[i]$ for all $i \in \{0, \ldots, m-1\}$, it is hard to find x.

(c) It is efficient to compute $z = \Omega(x, v)$ from the commitments on the bits of k and e.

The second security requirement for distance-bounding proof of knowledge protocols consists in preventing terrorist frauds. This requirement can already be achieved in the DBPK general scheme according to the following.

Proposition 4.3 *If Property 4.1, Property 4.2, and Property 4.3 are valid and if the DBPK protocol is performed correctly, then the terrorist fraud has a negligible probability of success.*

Proof: A successful execution of the *Proof of Knowledge* stage proves that the entity knowing the private key corresponding to the public key y has performed the *Bit Commitments* stage. Assume that the latter has been performed using k and e. Then, the probability for an intruder to perform the *Distance-Bounding* stage successfully using $(k', e') \neq (k, e)$ is equal to:

$$p = \prod_{i=1}^{m} (P[k[i] = k'[i]|a_i = 0] + P[e[i] = e'[i]|a_i = 1]) = 2^{-m}$$

This shows that without knowing (k, e), i.e. without knowing $x = \mathcal{D}_k(e)$, the probability of success of a terrorist fraud is negligible. □

The DBPK-Log consists of the same building blocks than those of the DBPK protocol. Moreover, the three following statements hold:

(1) The function $\Gamma : x \mapsto g^x$ respects Property 4.1 thanks to the intractability of the *discrete logarithm* problem.

(2) The function $\Omega : (x, v) \mapsto g^x \cdot h^v$ respects Property 4.2 thanks to the intractability of the *representation* problem (see Section 4.7.2).

(3) Given u, the one-time pad $\mathcal{E}_k(x) \mapsto u \cdot x - k \mod p-1$ satisfies Property 4.3 (see Section 4.7.3).

The properties listed above lead to the following.

Proposition 4.4 *If the DBPK-Log protocol is performed correctly, then the terrorist fraud has a negligible probability of success.*

Recall that the prevention of terrorist frauds makes the prevention of mafia frauds straightforward.

4.7.2 The Representation Problem

This section proves that it is not possible to generate two representations of a value z with respect to a generator tuple (g, h), i.e. two pairs (a_1, a_2) and (a_1', a_2') so that $g^{a_1}h^{a_2} = g^{a_1'}h^{a_2'}$. Generator g and h are chosen randomly so that the logarithm of h to the base g remains unknown. In our scheme, it means that g and h cannot be chosen by the prover. The remaining of this section is a simplification of the complete proof that can be found in [Bra93].

Proposition 4.5 *The following statements are equivalent.*

(1) There exists a polynomial-time algorithm $A_{(1)}$ which, on input a generator tuple (g, h), output a number z and two different representing index tuple of z:
$A_{(1)}(g, h) \rightarrow z, (a_1, a_2), (a_1', a_2')$ with $z = g^{a_1}h^{a_2} = g^{a_1'}h^{a_2'}$.

(2) There exists a polynomial-time algorithm $A_{(2)}$ which, on input a generator tuple (g, h), output a nontrivial representing index tuple of 1: $A_{(2)}(g, h) \rightarrow (a_1, a_2)$ with $1 = g^{a_1} h^{a_2}$.

(3) There exists a polynomial-time algorithm $A_{(3)}$ which solves the Discrete Log problem.

Proof: We only need to show probabilistic polynomial-time transformations from 3) to 1) and 2), since we can come up easily with feasible algorithms $A_{(1)}$ and $A_{(2)}$ if we have $A_{(3)}$.

(1) $\Rightarrow$ **(2)** Algorithm $A_{(2)}$ proceeds as follows:

1. Feed the generator tuple (g, h) into $A_{(1)}$ and receive z and two representing index tuples (a_1, a_2) and (a_1', a_2') of z.
2. Output $(a_1 - a_1', a_2 - a_2')$. Like this, $g^{a_1 - a_1'} \cdot h^{a_2 - a_2'} = z/z = 1$.

(2) $\Rightarrow$ **(3)** Algorithm $A_{(3)}$ proceeds as follows:

1. Generate a 2-tuple (u_1, u_2) at random, and compute the generator-tuple (g, h) according to $g = a^{u_1}, h = b^{u_2}$.
2. Receive an index-tuple (a_1, a_2) from $A_{(2)}$.
3. Compute and output $\log_b(a)$: $g^{a_1} \cdot h^{a_2} = 1$ it means that $a^{u_1 a_1} \cdot b^{u_2 a_2} = b^0$ and thus $\log_b \left(a^{u_1 a_1} \dot{b}^{u_2 a_2} \right) = u_1 a_1 log_b(a) + u_2 a_2 = 0$

$$\log_b(a) = -\frac{u_2 a_2}{u_1 a_1}$$

□

4.7.3 Encryption of the Private Key

To be compliant with Poperty 4.3, we propose a dedicated one-time pad: $e = \mathcal{E}_k(x) = u \cdot x - k \mod p-1$ where $k \in \mathcal{Z}_{p-1}$ and $u \in \{1, \ldots, p-2\}$ are randomly chosen before each encryption and where u is publicly disclosed while k is kept secret. The prime number p is a strong prime, i.e. $p = 2q + 1$ where q is an enough large prime. This scheme is compliant with:

- Poperty 4.3.a: With this encryption scheme, revealing e still ensures perfect secrecy of x:

$$P_{X|E}(X = x \mid E = e) = P_X(X = x) = (p-1)^{-1} \quad \text{for all} \quad x, e$$

- Poperty 4.3.b: We show that revealing b, where $b[i] = b_i$ is either $k[i]$ or $e[i]$ for all $i \in \{0, \ldots, m-1\}$, does not reveal information on x with probability $1 - 2^{m/2}$. Indeed, when e and k are defined in $\mathcal{Z}_{p-1}$, the combination of their bits (denoted b) is in $\mathcal{Z}_{2^m}$. Without a randomization factor u, i.e. when $e = x - k \mod p-1$, the distribution of b reveals information on x. Using a randomization factor u prevents such statistical attack because more than one result based on the same u is required to study the distribution of b_{add} when knowing b.

- Poperty 4.3.c: It is possible to deduce a representation of z depending on x from commitments on bits of k and e (see Section 4.6):

$$z = \prod_{i=0}^{m-1} (c_{k,i} \cdot c_{e,i})^{2^i} = g^{u \cdot x} \cdot h^v \mod p$$

Rationale Behind the Chosen Encryption Scheme

The remainder of this section explains how the encryption scheme was chosen. Property 4.3.b states that during the i^{th} challenge-response bit exchange in the *Distance-Bounding* stage, the prover has to reveal either the i^{th} bit of the encryption of the secret x i.e. $e[i]$ where $e = \mathcal{E}_k(x)$ or the i^{th} bit of the key i.e. $k[i]$. Of course revealing either $e[i]$ or $k[i]$ should not reveal anything about x.

Property 4.3.c implies that the proof of knowledge links the knowledge of k and e to the knowledge of x. Because k and e cannot be revealed, this proof has to be based on the commitment blobs of each bit of k and e, i.e. $c_{(k,i)}, c_{(e,i)}$.

Proposition 4.6 *One-time pad encryption respects Property 4.3.a and 4.3.b*

Proof: One-time pad ensures perfect secrecy: $e = \mathcal{E}_k(x) = x \oplus k$ where k is a m bit random string that is randomly chosen for each encryption.

$$P_{X|E}(X = x \mid E = e) = P_X(X = x) = 2^{-m} \quad \text{for all} \quad x, e$$

Revealing either bit i of k or bit i of e is done by choosing a random m-bits string: $s \in_R \{0, \ldots, 2^m - 1\}$

For all $i \in \{0, \ldots, m-1\}$, $b[i] = \begin{cases} \text{k[i] if s[i]=0} \\ \text{e[i] if s[i]=1} \end{cases} \quad b'[i] = \begin{cases} \text{e[i] if s[i]=0} \\ \text{k[i] if s[i]=1} \end{cases}$

Bits b are revealed and bits b' are kept secret. $x = e \oplus k = b \oplus b'$ and thus, $b = x \oplus b'$ is a new one-time pad that still ensures perfect secrecy of x. □

Unfortunately, exclusive-or does not suit modulo operations that seem mandatory when dealing with string commitments and thus does not respect Property 4.3.c. Indeed, $x = e \oplus k \mod p$ is generally not equal to $(e \mod p) \oplus (k \mod p)$. For instance, $10 \oplus 13 = 7 \mod 11$ but $(10 \mod 11) \oplus (13 \mod 11) = 8$. Another type of encryption is thus necessary.

Proposition 4.7 *One-time pad like encryption based on group addition $p-1$ respects Property 4.3.a and 4.3.c.*

Proposition 4.8 *One-time pad like encryption based on addition modulo 2^m respects Property 4.3.a and 4.3.b.*

Proof: Addition modulo can be used to implement perfect secrecy. For instance, let x be a secret in $\mathcal{Z}_n$ that is encrypted as $e = x - k \mod n$ where $k \in_R \mathcal{Z}_n$. Revealing e achieves the perfect secrecy of x:

$$P_{X|E}(X = x \mid E = e) = P_X(X = x) = \frac{1}{n} \quad \text{for all} \quad x, e$$

When $n = 2^m$, it is possible to reveal b while ensuring perfect secrecy of x. Indeed, in this specific case, e, k, and b are part of the same group $\mathcal{Z}_n$. □

However, when n is not a power of two, perfect secrecy of x is not ensured any more when revealing b. In this case, $e, k \in \mathcal{Z}_n$ and $b, b' \in \{0, \ldots, 2^k - 1\}$. For instance, choosing randomly bits of $k = 0111_b \in Z_{12}$ and $c = 1011_b \in Z_{12}$ can possibly result in $b = 1111_b \notin Z_{12}$. Figure 4.6 shows an example of the distribution of b for different choices of x.

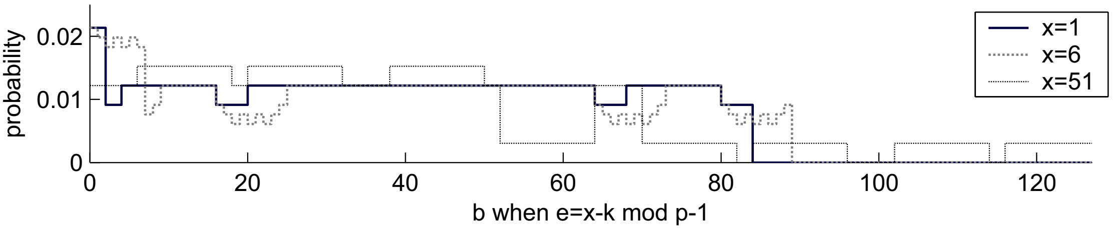


Figure 4.6: Distribution of b when $e = x - k \mod p-1$, $(p = 73, 2^m = 128, x = \{1, 6, 51\})$.

Even when the key is randomly chosen before each encryption, a statistical attack aiming at retrieving x is possible (no perfect secrecy) but remains expensive. However, it is possible to obtain some information on x (e.g. the most significant bit of x). A straightforward approach to make the statistical attack more difficult is to increase the noise, i.e. define k and e in a larger set of values, e.g. $\{0, \dots, n^\epsilon\}$ where $\epsilon \geq 1$. Unfortunately this incurs an important cost in terms of the number of challenge-response bits exchanged during the *Distance-Bounding* stage.

Proposition 4.9 *One-time pad like encryption based on addition modulo a Fermat prime p respects Property 4.3.*

Proof: A *Fermat prime* [MVO96] $p = 2^m + 1$ combines Propositions 4.7 and 4.8 and thus could solve our problem. □

Unfortunately there are less Fermat primes than Mersenne primes (i.e. $p = 2^m - 1$). Fermat primes can always be described as $F_n = 2^{2^n} + 1$ and only five Fermat primes are known: $\{F_0, F_1, F_2, F_3, F_4\}$. It seems unlikely that any more will be discovered soon. Anyway, F_0 to F_4 are too small to be used in the context of this chapter and the next F_k prime, does it exist, would be too large (i.e $F_k >> 2^{1024}$).

Our Encryption Scheme

We did not find a scheme assuring the perfect secrecy of x when b is revealed and assuring Property 4.3.c as well. In our protocol, we adopted an approach that makes the statistical attack more difficult without increasing the size of e and k.

The ciphertext $e = \mathcal{E}_k(x) = u \cdot x - k \mod p - 1$ where $u \in_R \{1, \dots, p-2\}$ and the secret key $k \in_R \mathcal{Z}_{p-1}$ are randomly chosen before each encryption of the secret x. Even though the parameter u is public, it makes statistical analysis more difficult.

In order to ensure that b reveals minimal information on x, it is necessary that the order of subgroups created by different x be the same. In other words, it is better to use specific primes such as strong primes: $p = 2q + 1$ where p and q are primes. It means that $\phi(p) = p - 1 = 2q$ and $\phi(\phi(p)) = q - 1$.

Efficient algorithms for finding strong primes exist. Indeed, the chances that a random integer p is prime are about $1/ln(p)$ and the probability that p be a strong prime is thus about $1/(ln(p) \cdot ln((p-1)/2)) \cong 1/ln(p)^2$. When p is a m bits value, the probability that p be a strong prime is about $1/(ln(2)^2 \cdot m^2)$. In other words, finding a 1024 bits strong prime requires half a million primality verifications.

Using strong primes, the distribution of b does not reveal any information on x (see Figure 4.7). In fact, the distribution depends only on $p - 1$ and 2^m when x is odd and

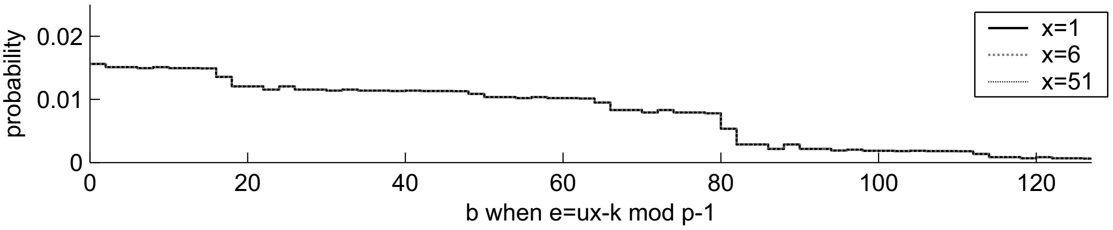


Figure 4.7: Distribution of b when $e = u \cdot x - k \mod p-1$ and p is a strong prime ($p = 83$, $2^m = 128$, $x = \{1, 6, 51\}$).

different from q. In practice, it is only necessary to avoid subgroups with very small cardinality, i.e. when p is a strong prime, $x \neq q$ is the only constraint.

Proposition 4.10 *One-time pad like encryption based on addition modulo $p-1$ and multiplicative factor u respects Property 4.3. when $e = \mathcal{E}_k(x) = u \cdot x - k \mod p-1$ and p is a strong prime, i.e. $p = 2q + 1$ where q is prime.*

Proof: The number of samples necessary to deduce information on x from b when $e = x - k \mod p-1$ is difficult to evaluate because it depends on the effect of the modulo operation and of the random selection of bits (see Figure 4.6). However, we can show that at least two different values of b are necessary to retrieve some information on x.

Thus, when x is encrypted as $e = u \cdot x - k \mod p-1$, it is necessary to collect at least two b corresponding to the same u. The birthday paradox shows that $2^{m/2}$ samples are necessary and the probability of disclosing information on x is thus smaller than the probability of using twice the same key k:

$$P_{\text{info on } x} < 2^{-m/2} \quad \text{where } m \text{ is the size of } b, x, k$$

□

Defeating Intruders with Partial Knowledge

The security of the scheme relies on the fact that a prover that is able to participate to the *Distance-Bounding* stage has to know e and k and thus can retrieve x. However, we can imagine that a malicious prover P could provide all bits of e and k but j random bits. For instance, $\tilde{e} = \{e_0, e_1, \ldots, \bar{e}_i, \ldots, e_{m-1}\}$ and $\tilde{k} = \{k_0, k_1, \ldots, \bar{k}_{i'}, \ldots, k_{m-1}\}$ where two bits e_i and $k_{i'}$ have been changed. Because the verifier selects randomly half bits during the *Distance-Bounding* stage, the probability of undetected use of $\tilde{e}, \tilde{k}$ is:

$$P_{\text{undetected } \tilde{e},\tilde{k}} = 2^{-j}$$

and the probability of I finding x is:

$$P_{I \text{ gets } x} = (2 \cdot m)^{-j}$$

For instance, with $m = 1024$ bits and $j = 6$, 32 tries are necessary to let I impersonate P and the probability that I discovers x is 2^{-66}. Even though this is a marginal attack, we propose to let V return an error message whenever the verification step in the *Commitment Opening* stage fails in order to help a potential intruder find out x. Hence, P cannot provide e and k or even $\tilde{e}$ and $\tilde{k}$ to another party.

4.8 Conclusion

In this chapter, we addressed the problem of terrorist frauds in application scenarios where cryptographic identification requires the physical proximity of the prover. Our solution consists in distance-bounding proof of knowledge protocols that extend Brands' and Chaum's distance-bounding protocols [BC93]. We first presented a general scheme that shows the main components of such protocols. We then presented a possible implementation of such protocols and analyzed its security properties.

The general scheme presented in this chapter (DBPK) could be used with any public key scheme if adequate commitment scheme, encryption method, and representation function exist. We proposed a solution, DBPK-Log = $DBPK[\alpha : y = g^{\alpha}]$, relying on a public key scheme based on the discrete log problem, a bit commitment based on the discrete logarithm, a group addition one-time pad, and the representation problem. This scheme could directly be used with ElGamal's and Schnorr's identification schemes that both rely on the discrete log problem.

Even though the proof of knowledge is zero-knowledge, the whole scheme can only achieve statistical zero-knowledge: when revealing b, the encryption scheme does not assure perfect secrecy of the private key x.

Terrorist frauds could also be prevented by combining the initial distance-bounding protocol with a certified tamper-resistant hardware [BR03b]. In this case, the verifier can check that k and e were generated by a trustworthy (i.e. certified by a TTP) hardware that will not disclose those secrets to an intruder. In comparison, the DBPK protocol is more general and is easier to deploy because it does not require any tamper-resistant hardware or device certification process.

Distance-bounding proofs of knowledge will be used in Chapter 5 to define location-stampers that are part of our trust-establishment scheme.

Chapter 5

History-Based Trust Establishment

"Every kind of peaceful cooperation among men is primarily based on mutual trust and only secondarily on institutions such as courts of justice and police."

– **Albert Einstein**

This chapter describes our solution to establish trust while preserving privacy. In this scheme, contrary to Part I of this dissertation, we assume a minimal infrastructure, i.e. users have a valuable secret and services are deployed. Parties deliver credentials defined in Chapter 3 that can define recommendation, group membership, or contextual information. The latter relies on distance-bounding proofs of knowledge presented in Chapter 4.

5.1 Introduction

This chapter combines results already presented in this dissertation in order to define a framework that enables trust establishment preserving privacy. Users collect evidence of their interactions in order to build their history. This history is subsequently used to assert some interaction during trust establishment. History may contain identity, role, or authorization credentials as well as recommendations, and other evidence like ownership, proof of location, etc. Two mechanisms are necessary to build such a scheme:

- *Unlinkable credentials* for proving history while preserving privacy.
- *Proof of location* for enabling context aware history.

In our solution, we have chosen to combine unlinkable credentials described in Chapter 3 and distance-bounding proofs of knowledge described in Chapter 4. However, a solution based on [CL01] should be studied when non-interactive signatures are not required, i.e. when trust is only built through interactive proofs. And when the threat model does not take mafia and terrorist frauds into account, other approaches could be used, e.g. ultrasound-based [SSW03] or infrared [BSSW02]. Isolation [BBD+91] is also an option to ensure strong guarantees on location.

5.2 State of the Art: How to Establish Trust

Before defining our scheme, we first summarize existing approaches that tackle trust and existing work combining trust and privacy.

When parties are registered and part of a controlled environment some accountability can be assumed. However, pervasive computing implies large population of unknown entities. Because of the infrastructure's dynamism, service providers will be confronted with requests from unknown entities, and users will need to obtain services in unfamiliar, possibly malicious environments. A party facing such a complex world needs a way to respond to new entities and assign meaningful privileges to them. Approaches based on human notions of risk [JP04, Dem04] are subjective and have to tolerate partial information. The ability to reason about trust and risk is thus necessary to let entities accept risk when interacting with other entities [CGS+03, Ing03, SDB03]. Figure 5.1 presents a common principle: the verifier collects evidence on the requester by monitoring this one or by gathering recommendation or reputation on this entity. The request and its context are combined with the evidence in order to evaluate how trustworthy the requester is and the risk related to the request. Once trust and risk are known, a way for reasoning about trust and risk is necessary in order to accept or reject an interaction.

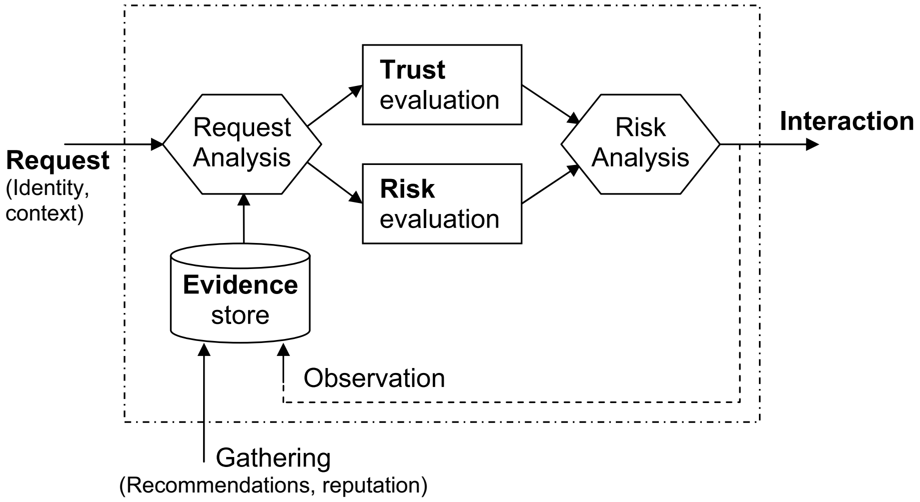
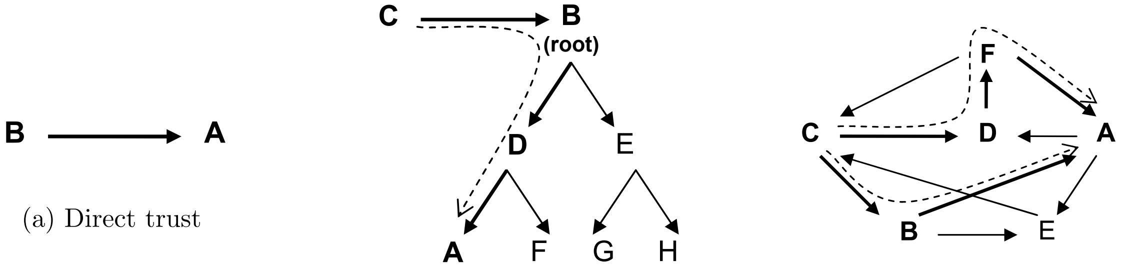


Figure 5.1: Trust and risk evaluations

In this dissertation we mainly focus on evidence gathering and storage. The remaining of this section presents different type of evidence. When some clear hierarchy exists among entities, a *public key infrastructure* [AH00] is sufficient to define trust relationships. A *web of trust* [Gar95] allows non-hierarchical trust relations similar to those formed in human communities. However, using a model based on human notions of trust is not straightforward. Three main sources of information are generally proposed to evaluate trust [ENT+02, EWN+03]: *personal observations* of the entity's behavior, *recommendations* from trusted third parties, and *reputation* of an entity. However, indirect sources of information may be used for evaluating the trustworthiness of a party: some approaches take the *physical context* into account during the trust evaluation [SSW03, KZS02, CMA00]. In a simple example, any person present in a room can be authorized to turn on the light. Surveys on trust establishment can be found in [GS00, KFJ01].

5.2.1 Deriving Trust from *a priori* Relationships

Trust is generally a derivation of some *a priori* trust. For instance, C trusts B and then C trusts A if B says A is trustworthy. In a web of trust, B is a friend of C, and in a public key infrastructure (PKI), B is the hierarchical authority of A.



(a) Direct trust

(b) Hierarchical trust

(c) Web of trust

Figure 5.2: Different ways to define *a priori* trust relationships.

Direct Trust

Direct trust is the simplest case: in Figure 5.2(a), B trusts A and grants A some rights. For instance, Bob trusts software company A and configures his Web browser so that any piece of code signed by A can use local facilities like writing on the hard disk.

In case of direct trust, the authentication is simple and can even rely on symmetric cryptography, i.e. A and B share a secret. Unfortunately, this approach is not scalable and can no more be used when the number of parties increases.

Hierarchical Trust

Hierarchical trust enables C to trust one root authority that manages a whole tree (See Figure 5.2(b)). For instance, company B is a partner of company C and C thus allows any employee of B to access a subset of its Intranet. Likewise, A, an employee of B, can access C's resources.

The rights of A can be derivate from the rights of B by using a delegation mechanism in a chain of certificates, e.g. SPKI [EFL$^+$99] or X.509v3 [IT00]. In other models like Kerberos [MVO96] or Web Services Security [FSLS03], the authority delivers a new security token to A.

Web of Trust

The Web of Trust was initially proposed in PGP [Gar95] to deal with identity based authentication. However, the concept can be extended to define trust as well. The principle is that a friend of a friend is a friend (see Figure 5.2(c)). For instance, when two friends of C say that K_{P_A} is the public key of Alice, C can assume that K_{P_A} is indeed the public key of Alice. In such model, the trust is statistical: rules define a threshold for accepting a statement.

Attributes: from Identity to Authorizations

Different types of attributes can be certified (see Figure 5.3): the identity (i.e. the name) [IT00], the role [SCFY96], authorizations [EFL$^+$99], or any other characteristic. When only the identity of a party is certified, it is necessary to map this identity to some form of trust. For instance, an *access control list* (ACL) can be used. In this case, it is necessary that the verifier has an *a priori* knowledge of the prover, indeed if there is no entry in the ACL corresponding to a given identity, no trust information can be deduced from authentication. A more scalable approach is to use different roles. In *role-based access control* (RBAC) [SCFY96], each party can prove that he has one or more roles and the verifier only has to maintain a mapping between roles and rights. The access control is thus able to deal with unknown parties as long as they can prove having a given role. Finally, when the number of roles becomes too large, it is possible to rely on attributes that directly define rights of credential holders (e.g. SPKI authorization certificate).

Trust Management

Modification of the rights of a party and especially revocation of his rights is a difficult problem in disconnected contexts. *Certificate Revocation Lists* (CRL) and *On-line Cer-*

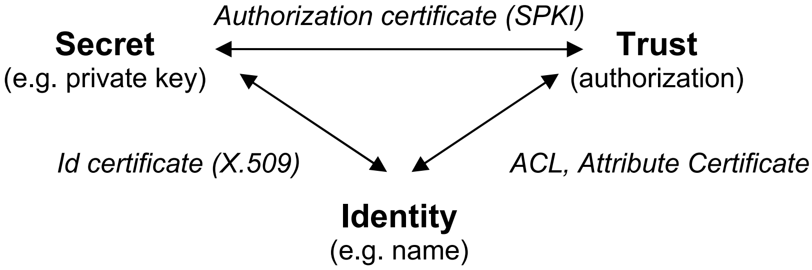


Figure 5.3: *A priori* trust

tificate Status Protocols (OCSP) enable to check which certificates have been invalidated. OCSP assume a way to contact this service: this approach cannot be deployed in off-line context. CRL can be used in off-line context as long as lists can be regularly updated. Unfortunately, the disadvantage of CRL is their potential large size, which makes them inappropriate for mobile devices. Another way to control the rights delegated to a third party is to rely on *short-term* [EFL$^+$99] or even *one-time* certificates [CFN89]. In this case, a mechanism for periodically renewing rights is necessary. It has already been shown [Riv98] that revocation lists are not an optimum solution and thus a hybrid approach is proposed: long-term certificates are used to define the roles, rights, authorizations, etc. and short-term certificates are created when rights are delegated. More sophisticated schemes exist: hash-based validity end [Mic96], certificate revocation trees, etc.

In this dissertation, we only focus on solutions based on credentials in order to support disconnected interactions. Work on security policies like PolicyMaker [BFL96], KeyNote [BFI99], or Ponder [DDLS01] are thus out of the scope of this dissertation because they rely on a permanently connected model.

5.2.2 Trust Establishment without *a priori* Relationships

In emerging environments like peer-to-peer data sharing, pervasive computing, or ad hoc networks, there is a lack of *a priori* trust among parties and new mechanisms are thus required to build trust in an *a posteriori* fashion based on monitored evidence (see Figure 5.4). The three main sources of trust information are defined in the following:

- *Personal observations* of the entity's behavior is ensured by recording the results of interactions. It is essential for the subjective evaluation of direct trust.

- *Recommendations* from trusted third parties provide the possibility for trust regarding unknown entities to be propagated. Recommendations are based purely on the recommender's personal observations and as such it is possible to associate a measure of trust in the opinion of the recommender (this is not the same as trust in the recommender for other actions).

- *Reputation* of an entity can be consulted in the absence of experience or recommendation. Reputation is anonymous in the sense that it is an aggregation of trust information from different sources (including recommendations that are passed to the verifier via intermediate parties) and as such the verifier cannot associate a level of trust with the opinion expressed.

A strong basis for trust is established through an entity's subjective observations. B observes the behavior of A and generates a recommendation, i.e. a credential signed by B that asserts the trustworthiness of A according to B.

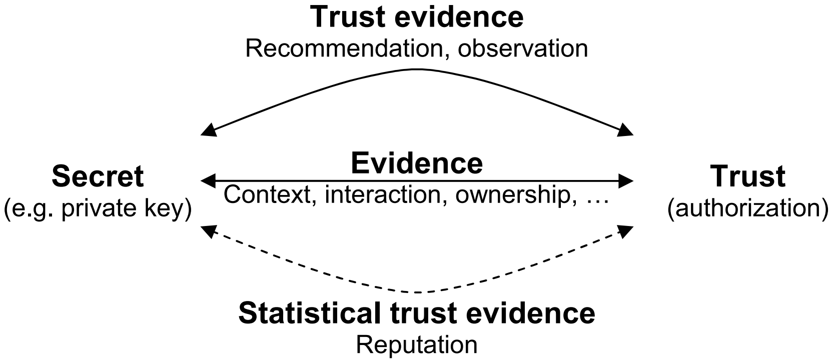

Trust evidence
Recommendation, observation
Secret
(e.g. private key)
Evidence
Context, interaction, ownership, ...
Trust
(authorization)
Statistical trust evidence
Reputation

Figure 5.4: Trust establishment

Evidence-based Trust

Trust can be built without any explicit *a priori* relationship, recommendation, or reputation concerning the trustworthiness of a party: when no information on the trustworthiness of a party is available, it is possible to use evidence that is not related to trust. This fourth source of trust has been less studied. For instance, the fact that Alice can prove she was member of the program committee of some conference on networking, makes her more trustworthy in the telecommunication community. Physical context (mainly location at a given time) can be used to establish trust. For instance, being in a meeting room could give some rights like accessing shared files or using the video projector. The generalized role-based access control (GRBAC) [CMA00] takes the context into account during access control.

In this dissertation we define history as an evidence store that is controlled by the requester. Evidence can be an *a priori* trust relationship, a recommendation, or a proof of context. We do not tackle reputation that is difficult to use when privacy of provers is a concern. In this scheme, Alice has to collect recommendations, as well as contextual evidence in order to be able to prove to another party that she is trustworthy.

5.2.3 Trust and Privacy

It is common to use trust for enabling privacy. For instance, the platform for privacy preferences project (P3P) [P3P] is emerging as an industry standard providing a simple, automated way for users to gain more control over the use of personal information on Web sites they visit. Thus, privacy policies of any visited web site can be checked and, if the site is trusted, private data can be provided. [DFHM01] proposes to use reputation mechanisms for improving the reliability and efficiency of MIX networks. Similarly, [RBM03] propose to use reputation for deciding whether private information can be provided to a given Web Service.

However, few approaches focus on how to establish trust while preserving privacy of users. [KP03] proposes a way to protect the identity of entities that feed a reputation system. Only [CM04] describes an approach related to our scheme: each prover receives recommendations that are linked to one of her pseudonyms and she can subsequently show the recommendation using another pseudonym. It is thus possible for a party to prove that she is trustworthy while preserving privacy. Unfortunately, this work is still at its starting point and thus cannot be compared with our scheme.

5.3 Proving Contextual Information

Having been at a location at a given time is an important part of the history. This section shows how results of Chapters 3 and 4 can be combined to define location stamps, i.e. unlinkable proof that the holder of a secret went to some place.

5.3.1 Location- and Time-Stamper

Let us define a *location- and time-stamper* (LTS) that certifies that some entity has been in a given place at a given time. The proof can be provided by a cell-phone operator that locates subscribers, by a beacon in a building, or even by using distance-bounding proofs of knowledge. A LTS can define logical location (e.g. continent, country, department, town, quarter, building, room) or geographic location (longitude, latitude). We only focus on the latter case because it does not require the definition of a complex data structure.

A location- and time-stamper company can deploy a network of public terminals and sensors. When Alice plugs her smart card in a terminal or when she passes a wireless sensor, she receives a location- and time-stamp with the following attributes: time (UTC, date) and location (latitude, longitude). Table 5.1 shows an example of the attributes that could be delivered by some LTS at Eurecom Institute.

Value	**Meaning**
180432	UTC in hhmmss format (18 hours, 4 minutes and 32 seconds)
24062004	Date in ddmmyyyy format (June 24, 2004)
43.6265	Geographic latitude in dd.dddd format (43.6265 degrees)
N	Direction of latitude (N - North, S - South)
007.0470	Geographic longitude in ddd.dddd format (7.047 degrees)
E	Direction of longitude (E - East, W - West)

Table 5.1: Context data: location and time

It can be represented by four attributes [`180432`, `24062004`, `436265`, `0070470`] that can be divided into frozen blocks (see Section 3.5.2): [`18|04|32`, `24|06|2004`, `43|62|65`, `007|04|70`] the meaning of each block is publicly known: LTS defines his public key as n and a set of e. For instance, e_1 is the least significant bit (LSB) of the time in seconds (0-59 : 6 bits), e_6 is the most significant bit (MSB) of the time in seconds, e_7 is the LSB of checksum of time in seconds, etc. If a location- and time-stamper provides the following credential to Alice:

`[18|04|32, 24|06|2004, 43|62|65, 007|04|70]`
i.e. she was in Eurecom building on 24 June, 2004 at four past six in the evening.

She can disclose a subset of this credential:

`[18|XX|XX, XX|XX|XXXX, 43|62|65, 007|04|70]`
i.e. She proves to be *someone that was in Eurecom building someday around six o'clock.*

Or, she can disclose:

`[XX|XX|XX, 24|06|2004, 43|XX|XX, 007|XX|XX]`
i.e. She proves to be *someone who was in the South of France the 24th of June.*

Note that hidden attributes are different than zero values (`XXX` $\neq$ 000). Indeed, `XXX` is represented as 000|00 and is not equal to 000 that is defined as 000|11. It is thus impossible to convert 09:08:30 into 09:00:30. Checksums of frozen blocks assure that information can be kept secret but cannot be modified. In other words, the value 09:`XX`:`XX` does not mean that some action occurred at nine o'clock but that it occurred between nine and ten o'clock. Another code should be chosen whenever modifications are supported.

5.3.2 Combining DBPK and Unlinkable Credentials

Before delivering a credential to Alice, a location- and time-stamper B has to check whether she is indeed present. For this purpose, we propose to use the scheme described in Chapter 4: distance-bounding proofs of knowledge. Distance-bounding proofs of knowledge can replace any proof of knowledge or signature of knowledge in existing protocols. Combining unlinkable credentials and distance-bounding proofs of knowledge is thus straightforward.

B runs a distance bounding-protocol with A: in the protocol described in Table 4.4, B acts as the verifier and A acts as the prover. The proof of knowledge used during the creation of an unlinkable credential (see Table 3.3):

$$\mathrm{PK}[\alpha \ : \ y_2 = a_b^{\alpha} \wedge \tilde{z} = \tilde{g_{ca}}^{(a_{ca}^{\alpha})}]$$

is replaced by:

$$\mathrm{DBPK}[\alpha \ : \ y_2 = a_b^{\alpha} \wedge \tilde{z} = \tilde{g_{ca}}^{(a_{ca}^{\alpha})}]$$

or:

$$\mathrm{DBPK}[\alpha \ : \ y_2 = a_b^{\alpha}] \qquad \text{and} \qquad \mathrm{PK}[\beta \ : \ y_2 = a_b^{\beta} \wedge \tilde{z} = \tilde{g_{ca}}^{(a_{ca}^{\beta})}]$$

The latter does not require new DBPK because it relies on DBPK-Log that is evaluated in Chapter 4. There are not two different discrete logarithms of y_2 to the base a_b and it is thus obvious that $\alpha = \beta \quad (= x)$. This proves that the entity knowing x is present and that this x is certified by the CA.

Second, B delivers a location stamp to A. Any entity that trusts B can thus assume that A really was at some location at a given time.

DBPK-LogLog

An interesting extension of our scheme could be the ability to prove that soemone with some history is physically present. For instance, Alice gets a credential from B proving that she is member of some group Q. Subsequently, A wants to prove to C that a member of group Q is present. To do this, a new distance-bounding proof of knowledge is required: DBPK-LogLog. Note that the credential provided by C will be linked to A and thus cannot be used by another group member. The signature of knowledge spk_1 of Section 3.4.4 is replaced by a proof of knowledge in interactive proofs:

$$\mathrm{PK}[\alpha \ : \ \hat{z_2} = \hat{g_b}^{(a_b^{\alpha})}]$$

To take proximity into account, the proof is replaced by:

$$\mathrm{DBPK}[\alpha \;:\; \hat{z_2} = \hat{g_b}^{(a_b^\alpha)}]$$

The distance-bounding proof of knowledge of a double discrete log to the bases g_1 and a_1, i.e. DBPK-LogLog=DBPK$[\alpha \;:\; y = g_1^{(a_1^\alpha)}]$, can be implemented as follows. Bit-commitment, distance-bounding, and encryption schemes are the same as in DBPK-Log but the final proof of knowledge changes:

$$PK[(\alpha, \beta) \;:\; z = g^{u\alpha} h^{\beta} \wedge y = g^{\alpha}].$$

is replaced by:

$$PK[(\alpha, \beta) \;:\; z = g^{u\alpha} h^{\beta} \wedge y = g_1^{(a_1^\alpha)}].$$

DBPK-LogLog as well as DBPK-Log relies on bit commitment based on discrete log, group addition one-time pad, and representation problem. Both rely on the same groups and the security properties are thus not modified. The only difference is the use of a statistical zero-knowledge proof of knowledge during the final step of the protocol.

5.4 History

This section studies how users can obtain, store, and disclose their history, which is a collection of unlinkable credentials.

5.4.1 Properties

The building block being defined, it is possible to use them in order to establish trust without *a priori* relationships. The framework relies on the following assumptions:

- Users (e.g. A) keep a history of their interactions.
  They can thus subsequently prove that they are recommended by B or that they indeed went to some location.
- There are third parties (e.g. B) that deliver credentials.
  Those credentials can define recommendations (e.g. A is trustworthy) or statements (e.g. A was at a given location at a given time). Each credential is signed by its issuer.
- Trust establishment can be done in disconnected mode.
  Before granting some rights to A, a service provider C can assert the trustworthiness

of A without relying on a remote TTP. Indeed, A keeps an history, i.e. a collection of credentials, that can be partially disclosed to C. No interaction with any issuer is required during the verification.

- History of A is non-transferable.
  Each user has a valuable secret x. This secret is certified by a CA and cannot be transferred to another party, i.e. it is equivalent to a private key giving access to other services like e-banking. Credentials are associated with this secret and credentials are thus non-transferable as well.

- Privacy of A is protected.
  While using credentials for proving former interactions, the anonymity and unlinkability of A are assured.

5.4.2 History Management

It is important to study who should store evidence. There are generally three different roles in an interaction: a client A that is asking for a service, some credential issuers B that previously interacted with A, and a service provider C that has to evaluate the trustworthiness of A based on credentials. Evidence can be stored in four different ways:

1. *Client* stores credentials and can selectively disclose them.
2. *Credential issuer* keeps a record of observations on clients. Service provider can request information on a given user.
3. *Service provider* stores information on users: recommendation sent by credential issuer, reputation of users, etc.
4. *Distributed* storage could be envisioned for instance by combining 1), 2), and 3).

While the second approach is generally used, the first solution has been chosen because it allows the user to control credentials related to her. Indeed, privacy is easier to achieve when the user can choose which information will be disclosed while establishing trust. It seems impossible to define a reputation or recommendation mechanism that ensures user's untraceability but that does not imply this user when exchanging information. Moreover, like in SPKI, the users can send credentials with request and no connection to trusted third parties is thus required when showing history.

The main drawback of credential schemes is the difficulty to modify or remove credentials once delivered. When revocation lists are not an option, it is possible to define different validity ends according to the type of attribute. Statements like proofs of location could stay indefinitely valid, role or identity credentials should be long-term, and recommendations as well as authorizations may have short-term validity.

5.4.3 Trust Establishment

This dissertation mainly shows how evidence can be proven in an unlinkable way. Building trust, i.e. granting access rights, based on evidence is straightforward in simple cases (e.g. sharing files among people in a same meeting room) but more complex reasoning on evidence could be necessary. For instance, the meaning of a document signed by *"a resident of building y"* or by *"someone that was at location z"* is application dependent. Application-level rules are also necessary to deal with access control or face-to-face interactions relying on such evidence.

5.5 Conclusion

This Chapter shows how unlinkable credentials of Chapter 3 and distance-bounding proofs of knowledge of Chapter 4 can be combined. The resulting framework enables unlinkable evidence on trust, i.e. recommendations. A can thus prove to C that she is trusted by B. The framework also enables unlinkable evidence on location so that A can prove to C that she was in some place at a given time. Finally any kind of evidence can similarly be defined.

The granularity of evidence disclosure can be chosen by the prover. This makes the scheme very flexible and assures that only required information is shown to verifiers in order to enable privacy.

Based on disclosed evidence, trust among parties can be established at application level. When no *a priori* relationship exist, trust evidence like recommendations may help. When no trust evidence is available, trust can still be based on general evidence.

Part III of this dissertation describes the implementation of a subset of the concepts presented in this dissertation. Generic credentials for application-level trust establishment are depicted and prototypes taking contextual information into account are also described.

Part III

Implementing Trust Establishment:

Architecture and Deployment

Chapter 6

Implementing Trust Mechanisms in Federations

”You have zero privacy now. Get over it.”

– Scott McNealy

This chapter describes how we implemented trust establishment in a very specific context: *business-to-employee* and *business-to-business* interactions. This is our main contribution to the WiTness project: the implementation of dynamic trust establishment within a federation of devices and users.

6.1 Introduction

Nomadic computing is becoming mature enough to interest software companies, hardware manufacturers, and operators of mobile network. The *WiTness project* (*Wi*reless *T*rust for Mobile Busi*ness*) sets out to define a framework for the easy development of secure *business to employee* (B2E) and *business to business* (B2B) applications in nomadic environments. Employees remotely access their corporate application server from their personal device, be it a laptop, a personal digital assistant (PDA), or a cell phone. In nomadic context, the operator’s smart card (SIM or USIM card) is a ubiquitous security module. One goal of this project was to use this security module to assure the security of business applications. In WiTness, it is thus assumed that each employee has a modified SIM card that acts as a security module hosting the cryptographic secrets necessary to authenticate this employee, to ensure the integrity and confidentiality of data, and to enable signatures.

This chapter focuses on our contribution to the WiTness project: an implementation of a subset of history-based trust establishment. This work was part of a pre-competitive research project mainly involving industrial partners and focusing on B2E and B2B applications. In such a context, minimal *a priori* trust relationships exist among parties and privacy is not a concern.

The "research work package", which we were leading, focused on the establishment of trust within a *federation* of users and devices. In nomadic computing scenarios, the personal device of a user is mainly used to access corporate services remotely. In pervasive computing scenarios, the personal device is also used to access local services offered by the environment, thereby creating a local federation with devices in its vicinity. Even if federations are an extension of the nomadic access to corporate servers, we assume that federations are self-standing associations where several devices communicate. They also extend the nomadic model of access to fixed corporate servers by enabling temporary access to these servers through members of the federation or to perform *off-line* operations using pre-fetched data.

Federations generally associate devices from different trust domains. For instance, a corporate PDA can be federated with a public terminal in order to get a more convenient display for reading corporate e-mails. To estimate the trustworthiness of surrounding devices, we use device certification (each device holds its own asymmetric key pair) and certificates defining agreements between companies. The WiTness framework being dedicated to mobile and pervasive computing, restrictions on computational power and communication channels are assumed. The creation and validation of credentials thus have to remain simple in terms of cryptography and parsing. Moreover, the scheme cannot rely on remote third parties during access control.

Privacy being an important issue, we first study whether it is affordable in nowadays mobile environments. Next, we describe our contribution to WiTness.

6.2 Cost of Privacy

Unlinkability and anonymity are important to avoid that all daily interactions of users be logged. However, privacy (as well as security) always has a cost in terms of communication and delays (Mix networks, private information retrieval [CGKS95]), computational power (proofs of knowledge), memory (one-time credential, e-cash), etc.

Moreover, as stated in the introduction of this dissertation, even with unlinkable credentials, it is necessary that the communication channels protect the privacy and that the application carefully controls the disclosure of attributes. In the WiTness project we used Bluetooth for personal area networking and GPRS for global communications. None of those communication media ensures privacy.

Finally, our collaboration with industrial partners has shown that the deployment of privacy is very limited in wired applications because there is no clear business model yet. Unlinkable credentials in mobile applications are moreover compromised by the limited computational power of SIM cards and cell-phones. Even a powerful PDA (iPaq, 200MHz) requires 0.2 second for generating a RSA (1024 bits) signature. When adding delays due to communications through Bluetooth and XML parsing, any transaction requiring some delegation of rights takes about one second. It seems difficult to slow down user interaction for privacy purposes. In the remaining of this section, we thus study how signatures and especially signatures based on a proof of knowledge could be speed up.

6.2.1 State of the Art: How to Speed up Signatures

This section presents different approaches that have been proposed to accelerate the computation of a digital signature and shows how interactions requiring a signature based on a proof of knowledge could be accelerated as well.

Server-Aided Signature: a small tamper-resistant trusted module uses the computational power of an untrusted server to compute a signature [BQ95, Bla96]. The server can try to obtain the secret (i.e. private key) of the module or to cheat with a false result. The trusted module must protect its secret and verify the computation received from the server. With enough bandwidth between the server and the trusted module, this scheme can be up to ten times faster than the trusted module alone. The drawback of this approach is that it implies the availability of a powerful server each time a signature has to be computed. Finally, numerous former server-aided signature schemes have been found insecure [Mer00].

Verifiable Server: an approach similar to server-aided signature is proposed in [BB03]. A trusted module asks a trusted server to sign some data. The server is in charge of all asymmetric cryptography operations. However, the signature scheme is modified so that a valid signature cannot be generated without the help of the security module. Non-repudiation is ensured even if the private key is hold by the server. As in the previous scheme this approach suffers from relying on surrounding servers. Moreover, in this case, the servers have to be trusted.

Batch Signatures: the idea of batch signatures [PB99] is to do only one asymmetric operation for a set of signatures. A set of messages are linked and signed together. For instance, a hash tree can be used to ensure that the signature of a message can be verified without knowing the other messages. This approach is useful when a set of messages has to be signed simultaneously but it cannot be used to accelerate the response-time of individual signatures.

Other Public Key Cryptosystems: it is also possible to use special signature schemes. For instance, the McEliece cryptosystem [McE78] can be used to efficiently sign data

[CFS01], the main drawback of this approach being the size of the public key. Elliptic curve cryptography could be investigated too. However, all asymmetric cryptosystem are computationally expensive compared with hash functions [PSW+01].

On-line/Off-line Signatures: the principle of on-line/off-line signature schemes is to enable the pre-computation of the signature in order to fasten the signature when the message to sign is known [EGM96].

Pervasive computing makes it difficult to rely on trusted third party or even untrusted surrounding servers because there could be no available server or the bandwidth could be insufficient for accelerating the signature. On-line/off-line signatures make it possible to define a background process that pre-computes signatures when the system is not overloaded. On-line/off-line concept can be applied to proofs of knowledge and signatures of knowledge that are based on Schnorr's signature scheme. For instance, the signature based on a proof of knowledge of a double discrete log $\mathrm{SPK}_l[\alpha : z = g^{(a^\alpha)}](m)$ (see Section 2.3.3) could be pre-computed as follows.

1. For all $i \in \{1, 2, \ldots, l\}$, generate random r_i and set $P_i = g^{(a^{r_i})}$.

Next, when m is known, the signer computes a hash and returns a set of values:

2. computes $c = \mathcal{H}_l(m \parallel z \parallel g \parallel a \parallel P_1 \parallel \ldots \parallel P_l)$.
3. Set $s_i = \begin{cases} r_i & \text{if } c[i] = 0 \\ r_i - x & \text{otherwise} \end{cases}$

The signature is the $l + 1$ tuple $(c, s_1, \ldots, s_l)$. Of course, in our unlinkable credential scheme, the signature can only be pre-computed when the signer knows in advance which attributes will be disclosed.

6.3 Pervasive B2E

The remaining of this chapter describes the framework that has been implemented to establish trust in federations of users and devices from different trust domains.

6.3.1 General Scenario

Alice comes to visit commercial partners. She only carries a small trusted personal device that could be a personal digital assistant (PDA), a watch, or even some wearable computer. A positioning service provided by the building makes it possible for her PDA to guide her to the meeting room, opening the doors to the only rooms she is authorized to access. While she is waiting for other participants to arrive, she wants to read her e-mail on a workstation in the meeting room. She federates her PDA with the workstation so that this workstation can have access to her corporate mail box and can display the list of e-mails received. Alice reads a few non-classified e-mails, and then selects one e-mail that is tagged as confidential. According to the security policy of her company, confidential data must not be displayed on any device not belonging to the company: the mail does appear on the smaller display of her PDA that is trustworthy. As other meeting participants arrive, their virtual business cards are displayed on each PDA. When the meeting starts, a space for securely sharing data is created between the meeting members. A slide show application in the visitor's PDA can then have all people in the meeting room sign an electronic non-disclosure agreement before it shows the visitor's corporate data on the local video projector. Even after Alice has left the building, she has access to a list of the interactions performed with meeting participants. Back in her office, she can browse classified data and e-mails on any corporate terminal. Because those displays are trusted by her company, she can directly access confidential data. This scenario can be divided into four types of pervasive B2E services:

1. *Ambient services / devices*

   Federative B2E technologies are an ideal tool for deploying ambient services all over corporate buildings. Employees can easily be empowered with devices like mobile phones, PDAs, e-rings, etc. so that they can be granted the right to open an electronically locked door, use a coffee machine, and so on.

2. *Face-to-face liable interactions*

   Federative B2E technologies should be expected to have a very important impact on face-to-face interactions. Federations make it possible to incorporate personal devices that unambiguously identify some individual through a digital signature into B2E and B2B corporate processes. This implies that employees' as well as companies' rights to perform operations may be better enforced. It also means that their liability may be established in some business critical process, like ordering some product, in that sense getting quite close to real-life signature.

3. *Access to corporate data and processes*

   Access to corporate data and processes using federations, i.e. surrounding devices, is essential to most B2E applications. Federative applications in which access control is a very important feature also bring up critical issues. Mechanisms like delegation, and even new types of delegations between users and devices, are required to enable such access control scenarios. Rights need not be given to a particular individual that may in fact be replaced by someone else. In addition, roles make it possible to grant access rights to several persons simultaneously, a very important feature given the fact that federations are dynamic groups.

4. *Access to group data*

Mobile groupware will probably become more ubiquitous with the development of federations. Such scenarios are clearly illustrated by business applications used in meetings. Security-wise, the needs in such scenarios are primarily authentication of the origin of documents or non repudiation of an action performed like modification of a document or approval.

6.3.2 Security Implications

The scenario outlined in Section 6.3.1 shows that B2E applications can be expected to collaborate dynamically with surrounding devices. A federation is defined as *a group of users and devices from different trust domains that collaborate within the same application.* Security is an integral part of federations. By their very nature, federations of communicating devices are more exposed to attacks than plain mobile devices since an application running within a federation spans across devices owned by different entities.

This scenario entails multiple security implications. Users need some authorization to use local facilities. They must also have the right to access corporate data remotely and a way to delegate specific rights to federated devices. Federations have to evaluate whether an execution environment is trustworthy. And last but not least, communication channels have to be protected, i.e. confidentiality, integrity, message authentication, and sometimes non-repudiation are necessary for applications used in a federation.

6.3.3 Trust Model

Corporate security policies define whether a given operation (e.g. dealing with confidential data) can be done by an appliance with an ascertained security level.

Most of federation security thus is about evaluating the rights of each user and the security level of each device that takes part in the federation. This evaluation is not easy to achieve in general. If a person makes use of a terminal in a public place, it is impossible to assume that the terminal is trusted in any way without some additional information that makes up the trust model. In general, there is no relation that can be exploited between the user or her company and the owner of the terminal: this can be called an *open trust model* as opposed to an *a priori trust model*.

The B2E context introduces assumptions that make it simpler to construct a workable trust model based on limited *a priori* trust information. First of all, public key based authentication is possible and meaningful since employees are directly managed by their corporation. In contrast, in an open trust model, there will generally be no authentication (or trust) infrastructure shared by all entities (see Parts I and II of this dissertation). Second, trust may be based on the partnership established between companies. The trust expectations regarding every partner's tasks and behaviors are contractual and can be translated into a security policy. Trust may also be based on the certification of devices, and specifically their level of tamper-resistance. Again compared with the open trust model, this assumption enables the automation of secure data distribution to different devices.

6.4 Security Architecture

This section presents the architecture developed to enable security features as described in Section 6.3.2.

6.4.1 Architecture Overview

Any employee has a trusted personal device that contains some credentials that may be used to prove his rights. Java enabled PDAs and cell-phones have been chosen for this purpose as Java is becoming an ideal platform for developing applications that have to be deployed on heterogeneous appliances. This trusted personal device is part of a federation of surrounding devices that can be managed by other companies. The federation is used by the employee when accessing some resources protected by a corporate server. Access control is based on the user's rights and on the security level of each federated device.

Each device has its own asymmetric key pair and can be certified by its owner. Security level evaluation and access control are based on those certificates. The corporate server and the trusted device are in charge of enforcing the corporate security policy.

Access Control.

Access control in WiTness is very flexible: it is based on generic credentials (attribute certificates) that can take into account different characteristics of the federation (see Section 6.5). Access control is based on users' rights that are defined by authorization or role certificates, devices' security-level (see demonstrator I, Section 6.6.1), or context like being in a same meeting room (see demonstrator II, Section 6.6.2).

Tamper-Resistant Module.

The tamper-resistance of the trusted device is also essential for enforcing the corporate security policy since it can hold private keys as well as the decryption keys for confidential data. A straightforward way to ensure that a device can be trusted is proposed by the Trusted Computing Group (TCG) [TCG]. The hardware is certified and can verify whether a certified kernel is running on top of it. This kernel controls the OS, which can check applications. This architecture makes it possible to prove that a given environment is running. As long as all layers are trusted (i.e. certified) and without implementation errors, it is possible to trust the environment. In other words, confidential data can be provided to any TCG public terminal with the guarantee that those data will not be misused. TCG being not deployed as of now, a pragmatic approach relying on trust-based distribution has been chosen: data are distributed according to the security level of each federated device. Security level evaluation is possible because the trust model proposed in this project is restricted to B2E scenarios. When B2B relationships exist, *a priori* trust is a realistic assumption as well: it is possible to base trust on the knowledge of who manages a device. Expectations can be set regarding partners' behavior or the way partner companies manage their devices (patches, antivirus, etc.) in order to protect themselves. Instead of proving the security of an involved device in the open, data distribution is performed according to the trust that can be derived from established relationships.

Flexibility versus Security

On one hand, employees would like to use transparently any surrounding appliance such as a screen embedded in a plane seat, a printer in an airport lounge, or location services offered by a building. On the other hand, the corporation has to protect its resources and prevent that an employee unintentionally reveal corporate data to untrusted and potentially malicious devices. The tradeoff between flexibility and security is based on the corporate security policy that defines whether a given device certified by an entity can be involved when getting access to non-public data. Such an open federation, in which devices owned by different entities are used, has to be restricted to secure interactions, and yet remain fully usable.

In this architecture, access control must be enforced, devices have to be certified, and resources have to be tagged. Deploying such a solution is possible in a B2E application context in which a local PKI exists [AH00] (without a global CA) and where trust can be specified. This architecture offers a pragmatic approach to secure the use of potentially malicious environments in B2E applications. It ensures that federations be secure without loosing their flexibility and user-friendliness.

Notations

Figure 6.1 shows different parties in a federation: companies, users (employees), devices, and resources. A full representation of relationships should include intermediate levels like departments or teams. Alice (A) is an employee of company C_A. She is visiting company C_B and would like to federate her cell-phone D_{A2} with the public terminal D_{B1}

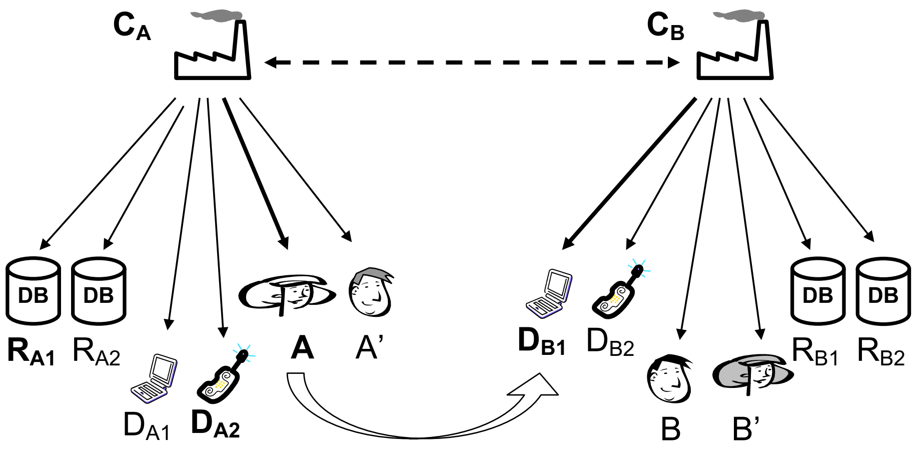


Figure 6.1: General overview of relationships in a federation

in order to access corporate data R_{A1}. A must be granted the right to use D_{B1} and to access R_{A1} (see Section 6.4.2). The security level of D_{B1} has to be evaluated in order to decide whether it can deal with classified data (see Section 6.4.3).

In the remaining of this section, we use a simplified notation for describing attribute certificates: $\mathrm{CERT}_{\mathrm{x}}(y, attr)$ means that party x signs an XML data structure (see Section 6.5) linking the public key of y to some attribute. Details such as validity end or certificate version are omitted in this notation.

6.4.2 User-level Access Control Infrastructure

In order to enforce security, an access control system to corporate resources has to be set up. In this system, each employee receives a security profile (set of rights) to access resources or to use other devices.

Access control is a classical problem that becomes complex when delegation is required. To be compatible with the issuance of local rights, we propose to use authorization certificates (similar to SPKI [EFL$^+$99]). Delegation happens for instance when a secretary gives a visitor access to a meeting room for a few hours. It also takes place when a visitor authorizes a wall display to access some specific corporate data and display them. Short term and application specific rights may also be required.

Accessing Corporate Database

Any employee of company C_A receives authorization certificates. For instance, Alice may receive:

$$\mathrm{CERT}_{\mathrm{C_A}}(A, r_A, \mathrm{deleg} = 1)$$

where $r_A = \{r_1, r_2, \cdots, r_n\}$ is a set of rights and *deleg* is the ability to delegate those rights to another person. Even if it does not appear in this notation, it is possible to choose which attributes can be delegated. It is also necessary to distinguish between delegation to other users (*deleg* tag) and delegation to devices. Indeed, when employee A wants to let a federated device D_{B1} access a resource, she has to delegate some specific short-term rights. For instance, A will browse a file server on her cell-phone and authorize the terminal in front of her to access a given file R_{A1} in order to open it for edition work. A provides the following certificate to D_{B1}:

$$\mathrm{CERT}_{\mathrm{A}}(D_{B1}, r_{D_{B1}}, \mathrm{deleg} = 0)$$

where $r_{D_{B1}} \subseteq r_A$ is a subset of Alice's rights that authorizes accessing R_{A1}. It is important to define $r_{D_{B1}}$ as precisely as possible and to set a short validity in order to avoid unexpected access from federated devices. When D_{B1} accesses the resource R_{A1}, it has to provide the following certificate chain:

$$\left\{ \underbrace{\mathrm{CERT}_{\mathrm{C_A}}(A, r_A, \mathrm{deleg} = 1)}_{\text{authorization}} , \underbrace{\mathrm{CERT}_{\mathrm{A}}(D_{B1}, r_{D_{B1}}, \mathrm{deleg} = 0)}_{\text{authorization (delegation)}} \right\}$$

where $r_{D_{B1}} \subseteq r_A$. During access control, the links between certificates of the chain are verified, and the signature and validity of certificates are controlled. A challenge-response protocol is used to check whether D_{B1} knows the private key $K_{S_{D_{B1}}}$ corresponding to the public key $K_{P_{D_{B1}}}$ embedded in the last certificate of the chain, and the server finally verifies that $r_{D_{B1}}$ indeed gives access to R_{A1}. When the whole verification is successful, access is authorized.

Using Surrounding Devices

Authorization certificates are similarly defined to enable the use of surrounding devices that can be owned by another company. When a visitor A enters the building of the partner company C_B, she needs some rights to benefit from the services offered by the environment (like using wall displays and printers, getting a map of a building and her location, opening a door to access a room, getting a lunch, etc.). Those rights can be based on an agreement between the visitor's company and the visited company, or can be delivered when the visitor enters the building. Depending on the security policy, rights can be provided automatically or require that a human being B be involved. For instance, the security policy may specify that anybody physically in the building can use wall displays but that registration is necessary in order to get access to some restricted areas. Different chains are possible. For instance, company C_B authorizes company C_A to use its wall displays D_{B1} and to delegate this right to its employees:

$$\left\{ \underbrace{\mathrm{CERT}_{\mathrm{C_B}}(C_A, r_{C_A}, \mathrm{deleg} = 1)}_{\text{authorization}} , \underbrace{\mathrm{CERT}_{\mathrm{C_A}}(A, r_A, \mathrm{deleg} = 0)}_{\text{authorization (delegation)}} \right\}$$

where $r_A \subseteq r_{C_A}$ and r_A is the authorization to use device D_{B1}. Similarly, company C_B could authorize a secretary B to delegate rights to any visitor. When A enters the building and registers, B grants her some rights:

$$\left\{ \underbrace{\mathrm{CERT}_{\mathrm{C_B}}(B, r_B, \mathrm{deleg} = 1)}_{\text{authorization}} , \underbrace{\mathrm{CERT}_{\mathrm{B}}(A, r_A, \mathrm{deleg} = 0)}_{\text{authorization (delegation)}} \right\}$$

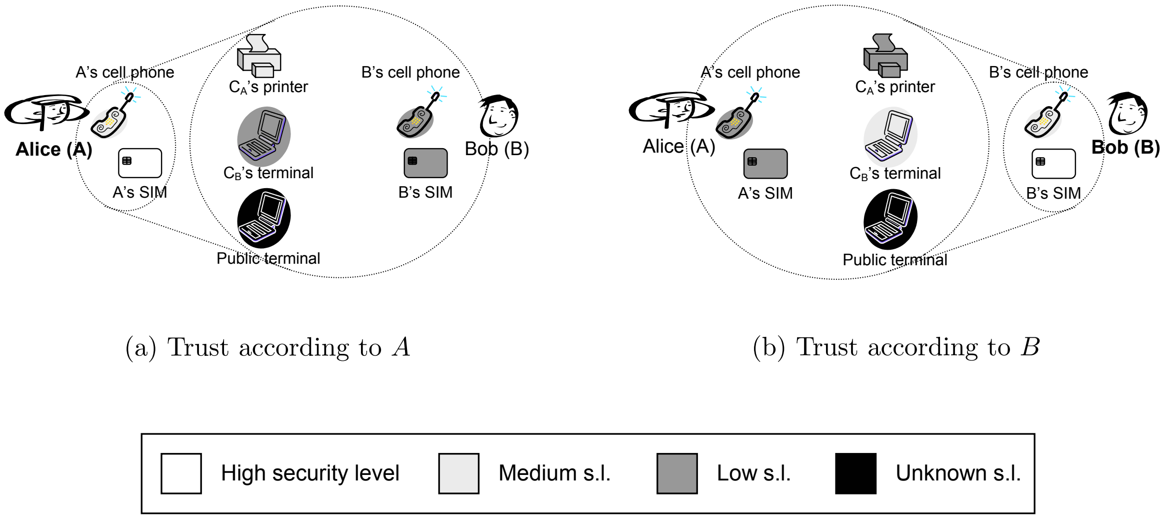
where $r_A \subseteq r_B$ and r_A is the authorization to use device D_{B1}. The certificate chain can be longer but it can be assumed in B2E environments that certificate chains remain short enough. Their verification can thus be simplified by providing well-formed chains instead of individual certificates that would have to be assembled by the verifier on the corporate side.

6.4.3 Device-Level Access Control Infrastructure

It is not sufficient to base access control on users' rights: characteristics of federated devices have to be taken into account as well.

Security Level Verification

The security level of a device depends on features of this device (owner, tamper-resistance, etc.) but depends on the observer as well. Figure 6.2 presents the trustworthiness of a whole federation according to two parties A and B.



(a) Trust according to A

(b) Trust according to B

(c) Keys: security-levels

Figure 6.2: Different views on the trustworthiness of a federation.

New types of credentials have been defined to evaluate the security level of federated devices. Device D_{B1} is owned and managed by company C_B (say for instance a wall-display in a meeting room that is physically protected). Company C_B provides to device D_{B1} an ownership certificate

$$\mathrm{CERT}_{\mathrm{C_B}}(D_{B1}, \mathrm{sl}_{D_{B1}})$$

where $sl_{D_{B1}} \in \{high, medium, low, unknown\}$ is the security level of D_{B1} according to C_B. For instance, $sl_{D_{B1}} = high$ could mean that D_{B1} is a computer managed and trusted by C_B (i.e. antivirus, regularly patched, physically protected, etc.) that can deal with any kind of classified data. Agreements between companies are necessary to formally define trust relationships. Such agreements are also defined by certificates. Company C_A is a partner of C_B. Employees of C_A frequently need to work in C_B's offices and use local facilities, e.g. D_{B1}. Because C_A trusts C_B, they can have the following agreement:

$$\mathrm{CERT}_{\mathrm{C_A}}(C_B, \mathrm{sl}_{C_B})$$

For instance, $\mathrm{sl}_{C_B} = low$ could mean that devices owned by C_B can be used to deal with confidential and unclassified data. The security level of a federated device is evaluated thanks to the chain:

$$\left\{ \underbrace{\mathrm{CERT}_{\mathrm{C_A}}(C_B, \mathrm{sl}_{C_B})}_{\text{agreement}} \;,\; \underbrace{\mathrm{CERT}_{\mathrm{C_B}}(D_{B1}, \mathrm{sl}_{D_{B1}})}_{\text{ownership}} \right\}$$

where $sl_{D_{B1}}$ does not have to be a subset of sl_{C_B}. It is not a delegation chain and the trustworthiness of D_{B1} according to C_A is defined by $sl_{D_{B1}} \cap sl_{C_B}$. The definition of security levels is application dependent and it is thus possible to define more precise trust relationships involving other parameters such as tamper-resistance or location.

6.5 Structure of Certificates

The natural choice for the format of attribute certificates has been the extensible markup language (XML). XML is becoming the standard format for data transactions on the Internet for many reasons: it is text based, human legible, self describing, structured, easy to treat, modular, and object oriented. Associating XML with one of the many existing libraries for parsing, displaying, transforming documents, and standards for signatures (XML-Dsig) [dsi], encryption (XML-encrypt) [XEn], remote procedure call (SOAP), user data exchange (SAML) [SAM], access control (XACML) [XAC] delivers a powerful combination. Using XML as the format for attribute certificates instead of S-expressions (SPKI) or ASN.1 (X.509) is not a novelty, and references can be found in an expired draft [OSM] and some projects: Akenti [ake] and ICare [ICa]. However, the WiTness framework deals with different problems and environments, which cannot be fulfilled with existing standards or applications.

WiTness certificates have been designed as a simple and very flexible data structure for the management of distributed credentials, easy to deploy and use in a off-line scenario. A certificate associates some content to the holder with a signature based tamper-proof guarantee of an issuer. The holder proves the ownership of the certificate through a classic challenge-response protocol demonstrating he knows the private key associated to the public key contained in the certificate. WiTness certificates holder and Issuer can either be a public key or another referenced WiTness certificate. In addition, validity duration, delegation level, and attributes can be evaluated at application level. The whole structure is signed with standard XML-Dsig using the issuer private key, although limitations in the Jeode Java Virtual Machine used on PDAs also required implementing an additional and alternative ad-hoc signature format. Figure 6.3 gives the structure of WiTness attribute certificates. More details on the structure can be found in [BCC$^+$04, BRKC03].

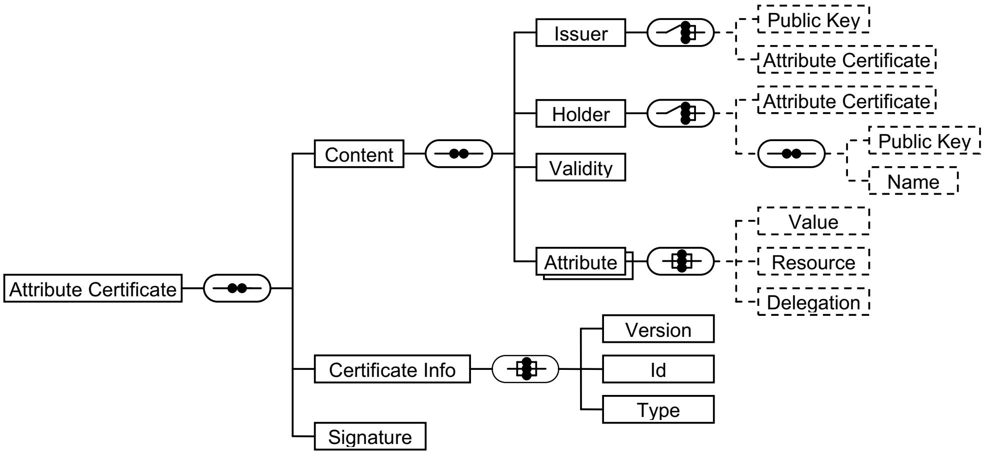


Figure 6.3: XML structure of a WiTness attribute certificate

Attributes

Most of the flexibility of this data structure comes from the very open attribute format. An attribute is defined by its *name*, *value*, *resource*, and *delegation*. The content of the attribute must be evaluated at application level, though some simple rules apply at library level to validate the certificate. The name of the attribute, associated with the application-level certificate type, allows applying semantic aware validation to the value for the attribute, and the optional related resource. A toolkit is provided with the certificate library in order to easily define new attributes with specific validity rules.

Delegation

WiTness certificates are designed to allow dynamic roles and rights distribution and delegation in a mixed on-line/off-line scenario. Any entity can act as a certification authority, creating new attributes or delegating existent attributes, and distributing them to other entities.

Certificate Chains and Validation

The chain from the root to the leaf delegated certificate is called a 'physical chain'. All parent certificates in a physical chain can be included in the final certificate for off-line validation, since they are generated in sequence. A different kind of chain, called 'logical chain' is used to create inter-domain delegation, and corresponds to an agreement between two companies, linking existing physical chains from different authorities.

For a certificate to be validated, the root of the physical chain must be a ‘trusted (root) certificate’ (as in X.509), or it must be trusted through an inter-domain delegation signed by a trusted certificate. Additionally, delegation level must be coherent, and application level validation is enforced on the attributes in the chain.

Off-line Revocation

Revocation of rights is a difficult problem in disconnected contexts. *Certificate Revocation Lists* (CRL) and *On-line Certificate Status Protocols* (OCSP) enable to check which certificates have been invalidated. OCSP needs full connectivity, while CRL can be used in off-line context as long as lists can be updated on a regular basis. The main disadvantage of CRL is their potential large size, which makes them inappropriate for mobile devices. A different way to control the duration of rights is to rely on *short-term* [EFL$^+$99] or even *one-time* certificates. In this case, a mechanism for periodically renewing rights is necessary. A hybrid approach was chosen in WiTness: long-term certificates are used to define the roles, rights, or security levels of employees and devices, and are updated along with revocation information when connectivity is available; while short-term certificates are used for delegating rights. Short-term delegation enables Alice to let a terminal access corporate data as long as she is present. When she leaves, i.e. when the terminal is out of Bluetooth range, authorizations are no more renewed.

6.6 Demonstrators

This section presents the two demonstrators that we implemented at Eurecom to validate the framework presented in this chapter.

6.6.1 Federative Access to Corporate Data

The first demonstrator shows how access control can takes user’s rights as well as security level of devices into account (more details can be found in Appendix C). Alice federates her cell-phone with a public terminal in order to enlarge the display when reading some corporate document (Web mail). Parts of the document are differently classified (e.g. confidential, secret). Depending on the trustworthiness of the terminal, be it a corporate terminal, a partner’s terminal, or even an unknown terminal, only a subset of the document appears. Each piece of data that cannot be decrypted is replaced by an applet displaying a button. When this button is selected, encrypted data is sent to a trusted enough member of the federation (e.g. Alice’s cell-phone) that can decrypt and display this part of the document. This allows flexible yet secure document viewing in federations. A similar concept was presented at PerCom’04 by IBM to protect users’ privacy

[RNP03].

This demonstrator requires at least two workstations (terminals) and one iPaq (trusted personal device). Local communications are based on the serial profile of Bluetooth. The application is written in Java to be portable on both platform (JDK 1.4 on workstations and Jeode Personal Java on iPaqs). On each platform, the application is split between a web server and a set of applets that run within a web browser. It thus is possible to define a federation of interacting browsers.

6.6.2 Pervasive Meeting

The second demonstrator tackles face-to-face interactions (more details are available in Appendix D). In a meeting room, employees of different companies can vote, share files, discuss an agenda, etc. The context, i.e. the fact that parties are present in the same room, is asserted by a simple mechanism: the short range of the communication channel. However, as presented in Chapter 4, more sophisticated schemes could be deployed. Federations make it possible to incorporate personal devices that unambiguously identify some individual through a digital signature into B2E and B2B corporate processes. This implies that employees' as well as companies' rights to perform operations may be better enforced. It also means that their liability may be established in some business critical process, like ordering some product, in that sense getting quite close to real-life signature.

The main goal is to demonstrate the implementation of the security features in WiTness libraries. Specific interfaces have been added to show the creation, distribution, and validation of certificates within a federation. This prototype illustrates how federations may rely on roles and track important decisions of employees.

6.7 Conclusion

Our contribution to the WiTness project is briefly described in this chapter. However, in terms of time and effort, the project was an important part of this Ph.D. thesis. The collaboration with R&D teams of industrial partners was fruitful in spite of some disagreements about expected results. Being leader of the research work package, we focused on extensions of the main framework and went towards pervasive computing. We implemented a certificate library and two demonstrators that were successfully presented during the second review meeting (Sophia Antipolis, February 2003) and final review meetings (Prague, April 2004).

Interestingly, the most time-consuming task has not been the implementation of the concept developed in this chapter (i.e. certificate library and demonstrators) but the use on immature technologies: access to Bluetooth and SIM card from Java. The industrial environment imposed the operating system Pocket PC (Windows CE). Pocket Linux being not an option, the only Java virtual machine available at this time was Jeode, an implementation of Personal Java (i.e. Java 1.1.8 + some extensions). Another constraint was the choice of Bluetooth (instead of WiFi) for local area networks. Bluetooth was chosen because it is available on major devices (cell-phones, PDAs, laptops, etc.) and because it only assures short-range communications and thus suits well personal area networks. Due to those constraints, a lot of time was lost to get access to Bluetooth stack and to a SIM card from Personal Java. Using modern XML parsers and Bouncy Castle crypto library as well as running web servers and applets on PDAs have been challenging and very instructive.

We were not involved in the modification of the SIM card for supporting business application security but however proposed to access this crypto-processor through a standard interface (Java crypto extension: JCE). This enables using transparently a SIM card, bouncy castle, or both together for cryptographic operations.

Our main contribution is a library for defining generic XML attribute certificates. For the sake of efficiency, XML digital signature has been replaced by a standard signature on raw XML data. Those generic credentials were used for combining *a priori* trust relationships and context-based trust within the prototypes.

Today, Bluetooth PAN profile, JSR-82 [JSRa], JSR-177 [JSRb], and J2ME personal profile are available and would make the development easier. Combining our approach with Web Services is discussed in a research report [BCC$^+$04]. The use of trust for protecting mobile code against malicious execution environment and protecting environment against malicious pieces of code is discussed in Appendix E and [BR04].

Conclusions

> *"It's only by going too far that you can hope to break the mould and do something new."*
>
> – **Francis Bacon** (the painter)

This dissertation has presented different approaches to deal with the establishment of trust relationships in pervasive computing. We tackled extreme cases where there is neither *a priori* trust nor permanent connectivity and where privacy of users is a concern.

The first part of this thesis presents two specific security protocols to deal with users that do not have valuable secrets, i.e. lacking something similar to a private key. The first scheme is computationally expensive because it relies on one-time credentials that have to be generated and stored beforehand and because electronic checks have to be associated with credentials. However, the principle of attaching attributes to "the purse of a user" instead of relying on the identity of this user seems promising in pervasive computing where identity is often meaningless. The combination of other digital cash and credential schemes could be studied for the same purpose. The second scheme presented in this dissertation enables encrypted attributes and only suits applications where attributes are defined in a small set of values. In other words, this approach cannot be used with complex attributes like location, authorizations, or time. It is however a new way to define unlinkable credentials suitable for specific attributes. Due to the strong limitations of both schemes, in the second part of this dissertation we assume that each user holds a valuable secret.

The second part described the architecture for collecting and showing any kind of evidence implemented as a credential. Credentials are based on an extension of group signatures in order to be unlinkable and anonymous. Credentials are strongly linked to the secret of the holder and thus are non-transferable. The attributes embedded in a credential can be selectively disclosed. Credentials can be used in challenge-response protocols as well as in non-interactive signatures. Finally, a distance-bounding proof of knowledge can be used when generating or presenting a credential. This general scheme enables a prover to collect a set of credentials in order to build a provable history that is subsequently used to establish trust relationships with unknown verifiers. Building trust relationships on evidence that is not directly related to trust is potentially fruitful when no trust evidence like recommendation or reputation is available. Trust-establishment and privacy seems to be two major requirements of pervasive computing security. Our approach fulfill both requirements and only recently other researchers started to work on this topic [CM04].

The last part of this thesis presented some implementation results. For practical reasons, privacy was not taken into account in this work. However, due to the increasing computational power of PDAs, it seems feasible to deploy unlinkable credentials in mobile environments. Unfortunately, as long as communication channels do not protect the privacy of users, it is useless to use any unlinkable credentials.

Perspectives

A possible continuation of this work would be to remove constraints on communications. In fact, assuming an on-line security model is realistic in numerous cases because a large part of business activities that require security require connectivity as well (access to corporate data, workflows, etc.). Permitting devices to have a permanent access to trusted third parties would simplify the scheme (see our research report [BCC$^+$04]). Another advantage of a connected model would be the possibility of using Idemix [CL01] instead of our unlinkable credential scheme. Idemix has two major advantages over our scheme (see Table 3.1): First it ensures unlinkability from any party including the credential issuer itself. Second it is becoming a standard and is already part of the trusted computing group's platform. Whether it is possible to combine distance-bounding proofs of knowledge with this scheme is still an open issue.

We think that distance-bounding proofs of knowledge and other distance-bounding protocols will be more and more important because physical artifacts are already ubiquitous in our daily interactions: plug a smart card in a point of sale terminal, auto-toll systems to pay for highway tolls, electronic keys for starting cars or accessing offices, etc. Thus mafia and terrorist frauds are more and more likely to occur. We would like to study how distance-bounding protocols could be integrated in trusted computing platforms and smart cards in order to avoid mafia fraud attacks. Indeed, even with tamper-resistant devices and sound security protocols, it is possible to mount a mafia fraud attack against, for instance, *what you see is what you sign*: the smart card protects the signer's private key and runs a challenge-response protocol to check that the terminal displaying the document to sign is indeed a certified trusted computing platform. However, similarly to Figure 4.3(b), the verifier could be in front of a dummy terminal that forward challenges to an authentic one. Such a fraud would enable the attacker to get a signature on a chosen document.

This dissertation mainly focused on security protocols but did not define application layers. Thus, it is still necessary to define verifier-side mechanisms to compute the trust-level of a prover according to disclosed evidence. Moreover, we have to define prover-side techniques to decide which part of a history can be shown without threatening the privacy. Statistical disclosure control seems to be an interesting approach to assure user's privacy at application layer.

Prototypes have shown that there is a lack of high-level tools for describing trust in terms of *a priori* relationships, recommendations, and context. Security policies are envisioned for defining general rules and we also proposed to use a meta object protocol for enforcing policies related to the protection of the execution environment [BR04].

Finally, we will be involved in *Mosquito*, another European project (IST-FP6) that will start in September 2004 and mainly focus on the security of pervasive business applications. Mosquito will exploit results of WiTness, which ended in April 2004.

Appendix A

Using Quantum Cryptography to Build Unforgeable Channels

This appendix describes a new idea: using quantum cryptography for proximity verification. It is still impossible to integrate this technology in mobile devices but it is a powerful solution that could enable secure device pairing.

Quantum cryptography [Sin00] is a technique for transmitting data over optical fiber or even through the open air using quantum state to code binary values. Quantum cryptography is already used to protect some communication channels. The first deployment outside of a physics laboratory was done using an optical fiber part of an installed cable used by the telecommunication company Swisscom for carrying phone conversations in 1995. It runs between the Swiss cities of Geneva and Nyon (my birth place), under Lake Geneva. Quantum cryptography does not rely on *Quantum computing*, which aims at using the quantum state to define bits (or qubits) for computation. If such a computer can be built, algorithms exist to tackle difficult problems, e.g. factoring very large composite numbers into their primes. This is, however, out of the scope of this dissertation.

A.1 Principle

Quantum cryptography offers a similar service than the Diffie-Hellman protocol without relying on complexity theory assumptions: it enables the exchange of a secret that cannot be known by eavesdroppers but it does not ensure authentication. The main application of quantum cryptography is the distribution of secret keys to secure point to point communications. A direct link, generally an optical fiber, is used to exchange the secret keys encoded as quantum states of photons. This approach does not rely on cryptography and is provably secure. Once the secret key is shared, common secret-key cryptography can be used to achieve the integrity and confidentiality of exchanged data.

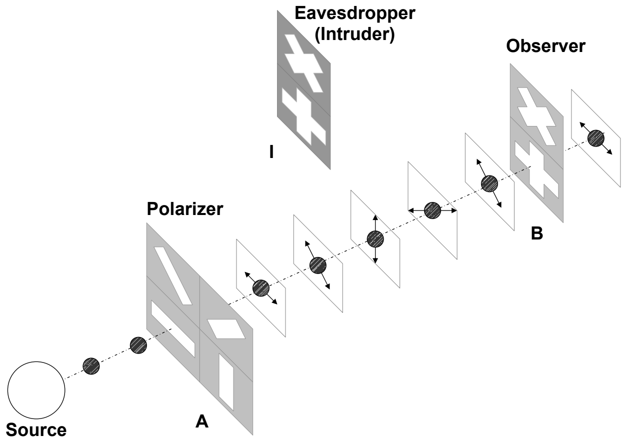


Figure A.1: Principle of quantum cryptography

Each bit of the secret key is encoded as a quantum state generally referred as a *qubit*. Typically photons are put into a particular state by the sender and then observed by the recipient (see Figure A.1). Thanks to the uncertainty principle, some quantum information occurs as conjugates that cannot be measured simultaneously. Depending on how the observation is carried out, different aspects of the system can be measured but observing one aspect randomizes the conjugates. Thus, if the receiver and sender do not agree on what basis of a quantum system they are using, the receiver may destroy the sender's information (see Table A.1).

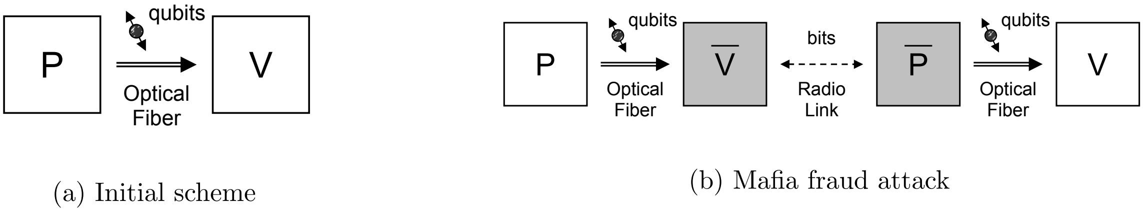
A's scheme	A's bit	A sends	B's detector	B detects	B's bit
Rectilinear	1	↕	+	↕	1
			×	⤡ or ⤢	0 or 1
	0	↔	+	↔	0
			×	⤡ or ⤢	0 or 1

A's scheme	A's bit	A sends	B's detector	B detects	B's bit
Diagonal	1	⤢	+	↔ or ↕	0 or 1
			×	⤢	1
	0	⤡	+	↔ or ↕	0 or 1
			×	⤡	0

Table A.1: Eavesdropping quantum cryptography

A.2 Unforgeable Channels

The main properties of quantum cryptography is that an eavesdropper cannot get the secret exchanged by two entities and when the channel is noiseless, eavesdroppers can be detected. A resulting property can help to build unforgeable channels: it is impossible to forward data without being detected. The intruder can redirect the photon through an optical fiber but cannot read the data and transmit them through another media like radio (see Figure A.2. Note that this technique only prevents mafia frauds attacks (prover and verifier behave fairly).



(a) Initial scheme

(b) Mafia fraud attack

Figure A.2: Quantum cryptography in order to disable mafia frauds

Quantum cryptography is an interesting approach but current technology neither enables its implementation in small mobile devices nor allows cheap deployment. Moreover, it does not fit big appliances like printers or ATM that could forward photons thanks to an optical fiber. This method could fit very well scenarios in which a user holds the artifact he wants to authenticate because it is easy to see that this artifact is not connected to something else. This method can prevent distance as well as mafia frauds but cannot prevent terrorist frauds.

Theoretically, quantum teleportation could be used to attack this scheme. Quantum teleportation enables the exchange of one quantum state through a classical channel when the sender and the receiver share an entangled pair of particles. Thus, if $\bar{P}$ and $\bar{V}$ could store entangled pairs, they could threaten this scheme. However, entangled pairs are not stable and the probability of successful state transfer is bounded. Thus, such an attack does not seem realistic.

Appendix B

Drag-and-Drop: User-friendly Distance-Bounding Protocols

It is well known that security can have a strong impact on usability. Security leads to more complex protocols, requires users' action such as entering a password, and sometimes relies on additional tokens such as smart cards. In pervasive computing, users should transparently interact with computers without caring about security. This appendix shows how distance-bounding protocols and authentication of artifacts can be deployed in a user-friendly way.

B.1 Usability of the Environment

Interactions between users and artifacts can be complex but have to stay as transparent as possible. If some service requires the explicit interaction of users with real-world objects, this interaction should be rendered as intuitive as possible.

Discovery and advertisement [Ric00] approaches impose the selection of virtual representations of surrounding devices: it is obviously neither transparent nor intuitive to select a printer in a list on one's PDA when it stands in front of the user and could be directly selected by touching it.

Physical contact with the device whose service is required may be constraining in the sense that it has to be within physical reach, but is extremely intuitive for the user; rooms containing devices out of reach might even be equipped with authentication switches alike light switches. However, plain physical contact lacks the details provided by a computer interface, which suggests that combining both approaches might be relevant.

Multi-party interactions involving multiple devices and/or multiple users should be possible in pervasive environments. Most paradigms essentially focus on two party interactions scenarios, but scenarios of sales typically involve two people and one device for instance.

B.2 Security Requirements

Securing interactions between users and artifacts is mandatory as soon as resources have to be protected. Access Control is necessary to verify the rights of users interacting with the environment. Pervasive computing environments are shared by numerous users. It is the reason why it is necessary to account for their respective actions and to keep a proof that some critical interaction happened. In this context, non-repudiation can be required for establishing user liability.

Artifacts have an important role in pervasive computing and it is thus necessary to have a way to verify their characteristics. For instance, an interaction may require that the rights of an artifact or who its owner is be verified. The establishment of trust relationships has to be investigated as well.

Last but not least, finding a way to ensure security without bothering users is not trivial. Prompting the user for passwords each time he interacts with his environment is not a credible solution and does not fit with the expected transparency or "pervasiveness". We propose a new user-centric approach based on personal tokens with contact interface that are used to touch devices in order to dynamically and securely create relationships. In this solution, the user is authenticated and his rights are used during the operation.

B.2.1 Strong Authentication in Pervasive Environments

Pervasive computing environments seem to head for totally wireless interactions, supposedly simplifying the interactions between artifacts. Unfortunately, wireless networks make it easier to snoop on some protocol or to attack it by inserting malicious traffic. *Artifact authentication* is thus an essential security feature.

Former work is mainly concerned with providing a unified and informative interface to all services and artifacts. It relies on a purely virtual representation, accessible on a PDA for instance, in which the user chooses the service or specific artifact that he wants to make use of. In such an approach, traditional techniques of authentication and key distribution apply quite straightforwardly. This approach however does not bridge the gap between the theoretical representation of the environment and the reality of it: no proof is given that the printer in front of the user is the one that the PDA says should be in this room.

Depending on the criticality of the resources accessed or of the goods exchanged, it can be sufficient to base the access control or the transaction on weak authentication. However, pervasive computing will lead to numerous payment or micro-payment schemes and access to critical resources will take place in some applications. This appendix aims at answering the following question: *Is it possible to define user-friendly interactions in pervasive computing environments requiring strong authentication?*

B.2.2 Presence of User

Distance-bounding protocols can be used to verify that an artifact is physically present. When it is necessary to check if a human being is present during a transaction or to deliver a proof of location, it is in fact the presence of his token that is verified, the token being for instance an electronic ring [Cur98] that can be worn by the user. Tokens may potentially carry numerous rights such as accessing office, house, car, and paid services and may be used to sign documents, for payment, or to delegate rights. However, even such tokens can be stolen: directly authenticating the user of a token is thus critical. PIN codes are generally used to unlock tokens such as SIM cards. However, in pervasive computing, it is not possible to rely on passwords for each interaction because it suppresses intuitiveness. Two mechanisms can be proposed to diminish the threats on the personal token: when the interactions are done on-line, it is possible to create a token revocation list; when the interactions are done off-line it is necessary to lock the token periodically. Both rely on surrounding artifacts: the former needs a terminal to add the token to the revocation list; the latter requires a way to delegate the right of unlocking the token to another artifact. For instance, an e-ring could be unlocked by entering a PIN code on a cell-phone or by touching a finger print reader integrated in the user's watch.

B.3 Solution: Drag-and-Drop Metaphor

Pervasive computing requires the most transparent interaction semantics in order to remain intuitive: touching a device holds such a promise. For instance, Alice may plug her ring into an ATM in order to be authenticated and retrieve money. In order to prevent mafia frauds, the setup will be slightly more complex so that, for instance, the user be warned with a blinking led on her ring that the device she believes to be an ATM is potentially a fake because it cannot associate a certificate issued by a bank with a distance-bounding protocol. In addition, interesting interactions often involve three or more artifacts. For instance, Alice may have to connect two artifacts using her ring.

This section describes how the *drag-and-drop* metaphor can be recycled to implement several security mechanisms that often come up in pervasive scenarios: how to be sure that an artifact conforms to some specification? How to enable an artifact to perform some access on behalf of a user? How to provide some proof about an operation? Finally, how to be sure of the ownership of an artifact and how to transfer it?

In the following, we will assume that each user (e.g. Alice A) carries a personal token D_A that identifies her and that is used to interact with surrounding artifacts. We propose to implement tokens as tamper-resistant electronic rings with dedicated distance-bounding interface. They can be used for protecting the private key K_{S_A} of their owner and to select other artifacts by touching them.

Each artifact D_i has its own private key $K_{S_{Di}}$. It is possible to provide rights to an artifact by defining an authorization certificate. The features and the owner of an artifact may also be defined by attribute certificates.

B.3.1 Verifying Attributes of Artifacts

Data are often associated with a physical object, be it a comment associated to paintings in a museum or the expiring date of some food. Protecting those data against forgery requires certifying and linking them to the artifact thanks to a distance-bounding protocol. For instance, when someone buys an artifact, let's say a pervasive Swiss watch or a pervasive box of pills, it is necessary to verify that the manufacturer did actually certify this artifact. Mechanisms ensuring that artifacts (or at least the chips associated to these artifacts [KZS02]) cannot be cloned and that they are physically present are required. This approach relies on the following hypotheses:

- Tamper-resistant hardware are available
- Hardware interfaces dedicated to distance-bounding protocols are deployed.
- Each user has a unique personal token (e-ring) that is used for authentication and interactions.

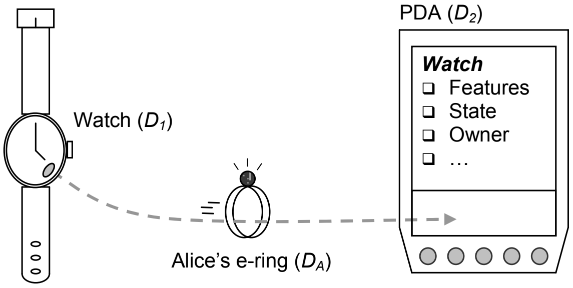


Figure B.1: Drag-and-drop to show attributes of an artifact

Figure B.1 shows how a drag-and-drop mechanism can be used to intuitively display the characteristics of an artifact. This protocol ensures that the displayed data correspond to the artifact that has been touched (e.g. the watch). The user trusts his token to perform a distance-bounding protocol with the artifact. The artifact can be verified anonymously using the token, hence protecting the user privacy. Alternately, the artifact can require the user identity or an authorization to control access to its services or resources.

Protocol Description

Table B.1 describes how a drag-and-drop protocol can be used between two artifacts in order to verify attributes of the first one.

1)	**Drag**			
1.1)	D_A	$\rightarrow$	D_1	<Get description>
1.2)	D_A	$\leftarrow$	D_1	$\mathit{CERT}\text{-}D_1 = \mathit{CERT}_{CA}(K_{P_{D1}}, \mathit{attributes})$
1.3)	D_A	$\Rightarrow$	D_1	Distance-bounding protocol
2)	**Drop**			(before timeout)
2.1)	D_A	$\rightarrow$	D_2	<Put description>:
				$\mathit{CERT}_{DA}(D_A \text{ touched } D_1, \mathit{CERT}\text{-}D_1)$
2.2)	D_2			Display description of D_1

Table B.1: Basic drag-and-drop protocol between two artifacts

The user touches the two artifacts in succession with his token. The protocol is described in two parts: drag, which is the interaction between the token D_A and the artifact D_1, and drop, which is the interaction between the token and another artifact D_2. In 1.2, D_A receives a certificate describing D_1 signed by a Certification Authority (CA). In 1.3, D_A verifies that the touched artifact knows the private key $K_{S_{D1}}$ corresponding to the certified public key $K_{P_{D1}}$. As a result of the drag operation, the token has the proof that *it touched an artifact knowing* $K_{S_{D1}}$ and that *CA certifies that the entity knowing* $K_{S_{D1}}$ *has some attributes.* In 2.1, D_A drops the certificate of D_1 to D_2. It is certified by D_A.

On a side note, the distance-bounding protocol, for it involves nonces, provides a one-time proof that the token is in contact with the owner of the secret key each time it is performed. No additional nonce or timestamp is required in the derived protocol. In any case, a timer should be available in the token to cancel operations if the drag is not performed after some timeout.

This scheme can be extended. For instance, mutual authentication could be required in order to provide access control so that D_1 only delivers description to authorized tokens. In this case, there is an obvious tradeoff between access control and privacy, that is, can anonymous drag be used or not. Another important feature is non-repudiation of interaction that aims at proving that an interaction occurred between users and/or artifacts. Distance-bounding protocols are not sufficient to ensure non-repudiation. Indeed, after a distance-bounding protocol $D_A \Rightarrow D_1$, the artifact D_A has the proof that D_1 is within some distance. However this information must be associated with a certain context: providing a non-repudiation service to a pervasive interaction may not only require certifying the identity of the involved parties, but also the time at which it occurred and the history of previous interactions, for instance. Different types of non-repudiation (e.g. order of interaction, initiator, or target) can be integrated within the drag-and-drop protocol.

Appendix C

Demonstrator 1: Device Certification

A prototype showing in what way secure federations are required in B2E was developed at Eurecom and presented during the second review meeting of the WiTness project.

C.1 Prototype Scenario

A salesman travels with his trusted device. He can use this device to access corporate e-mails but reading them on a small display is not always user-friendly or even realistic. He therefore uses surrounding public terminals, laptops, or video projectors to enlarge his display. The salesman selects an e-mail on his trusted device and delegates it to a discovered device. If the security level of the latter device permits so, this device can be allowed to retrieve the e-mail and to display it.

The prototype focuses on a specific service: displaying Web pages on surrounding devices. It is of course possible to extend this concept to other distributed applications like edition or signature of documents. In our prototype, the access control protecting Web pages takes into account the authorizations of users as well as contextual information: the security level of involved devices.

The corporate security policy defines which data can be accessed by a given user, e.g. e-mails, and the classification of the data that can be handled by federated devices. For instance, some e-mails or attachments can be tagged as confidential. The access control leads to the following cases:

- **Case 1: User not authorized**. Access control ensures that the salesman can only access the documents that he is authorized to.

- **Case 2: User authorized but untrusted terminal**. Using a public terminal, the e-mails (or parts of e-mails) that are confidential will not appear but will be replaced by a hypertext link. Upon selecting this link, the e-mail is displayed by the salesman's personal device that is a trusted enough member of the federation.

- **Case 3: User authorized and trusted terminal**. When the salesman uses a terminal of a partner company, the confidential document is displayed.

Federations allow user-friendly interactions with surrounding artifacts and security is ensured by combining authorization (i.e. what a user is authorized to do) and security levels (i.e. what kind of data can be securely handled by a federated device).

C.2 Principle

Access to resources is protected at two levels (see Figure C.1). The first access control layer (label 1) verifies whether a given user is authorized to access the required resource. Each employee has his own asymmetric key pair, the private key being kept in the user's PDA or physically protected by a dedicated tamper-resistant module. SIM cards [sim] are specifically envisioned because of their ubiquity in mobile devices. The user's rights are described in authorization certificates referencing the user's public key as an identifier. The second access control layer (label 2) distributes data according to the security level of each federated device. Each device has its own asymmetric key pair that may also be physically protected. Manufacturers, owners, or administrators of devices can install certificates providing useful information about the device security level. When the security level of a federated device is known, it becomes possible to decide whether it can access some confidential data or not. Layer 1 resolves a chain of certificates defining the relationships between the user and a know root key and layer 2 resolves a set of chains corresponding to the relationships between each federated device and a root key. Data is distributed according to the employee and the devices he is using.

C.2.1 Data and Key Distribution

Data distribution is based on the rights of the requestor and on the security level of the devices involved. Suppose that a set of resources $R_{req} = \{R_{A1}, R_{A2}, \ldots\}$ are requested from a server of company C_A by an employee A using a federation $F = \{D_{A2}, D_{B1}, \ldots\}$. The first access control layer requires the user authorization chain and delivers authorized resources $R = r_A \cap R_{req}$ where r_A are the rights of A.

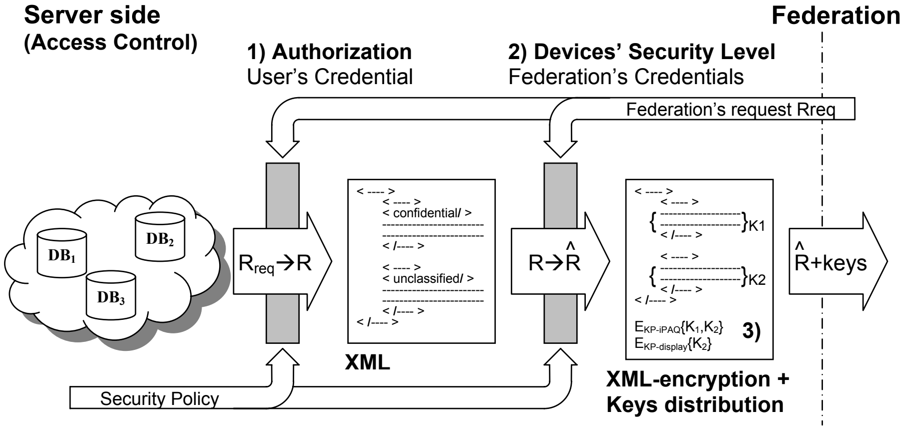


Figure C.1: Server-side two stage access control based on the employee's rights and on the security level of each federated device

Each device can receive any data in an encrypted form but can only retrieve keys corresponding to its security level. Resources R are encrypted according to their classification $cl \in CL$. For instance, $CL = \{\text{unclassified} = 0, \text{confidential} = 1, \text{secret} = 2\}$. The classification of a resource R_i is defined as $cl(R_i)$. Tags are associated with resources in order to specify their classification. The security level of a device D_j is defined as $cl(D_j)$ (see Section 6.3.3). A chain of certificates has to be resolved for each device in order to know whether it is trustworthy enough. When $cl(D_j) \geq cl(R_i)$, device D_j is enabled to deal with resource R_i. A symmetric key K_{cl} has to be defined for each classification cl:

$$\text{For all } cl \in CL \quad : \text{Server generates a symmetric key } K_{cl}$$

Each resource R_i has to be encrypted with the symmetric key K_{cl} corresponding to its classification $cl(R_i)$:

$$\text{For all } R_i \in R \quad : \text{Server computes the encrypted resource } \widehat{R_i} = E_{K_{cl(R_i)}}(R_i)$$

The set of encrypted resources is $\widehat{R} = \{\widehat{R_i} \mid R_i \in R\}$. Before sending it to the federation, the devices trust chains are necessary for discovering security levels in order to distribute the necessary keys:

$$\text{For all } D_j \in F \quad \text{and for all } cl \in CL \quad : \text{if } cl(D_j) \geq cl : \text{Server computes } E_{K_{P_{Dj}}}(K_{cl(D_j)})$$

Key distribution (see label 3 of Figure C.1) ensures that federated devices can only receive keys for decrypting data they are authorized to deal with. For instance, a terminal that is trusted enough to deal with confidential data will receive $K_{confidential}$ and $K_{unclassified}$ but will not receive K_{secret}.

C.2.2 XML Documents

XML has been chosen for defining certificates and also as the format for document storage. XML is actually becoming the natural format for many kinds of documents, be they in pure text, vector graphics, or more complex data formats; XML databases are spreading. Adding trust and security information to an XML document is quite straightforward, because XML has been conceived to be naturally extensible. In this work, there are two different kinds of XML documents. One is the *stored* document, which can be seen in the middle of Figure C.1. The other one is the *transmitted* document, on the right part of Figure C.1.

Upon a user request to the server for a resource stored in the corporate database, an XML Parser generates a document depending on security policies, user credentials, and federated devices credentials. This document contains some cleartext parts, some encrypted parts, and an additional section to distribute symmetric keys for the encrypted sections. The receiving device will be able to show the encrypted parts if it can retrieve the corresponding key.

Using XSLT, XML documents could be directly transformed into a proper format for visualization, depending on the device.

C.3 Platform

In the time frame of this project, no device offering all required features was available. The following hardware was expected:

1. Short range point to point wireless communication (preferably Bluetooth).
2. Global communication (cellular network: GSM, GPRS, or UMTS; or wireless LAN).
3. Smart card reader (preferably integrated SIM card reader).
4. Sufficient computational power and memory for applications and cryptography.

The following software requirements were also expected:

5. Support for high-level and full featured programming language.
6. Access to the Bluetooth stack from this programming environment.
7. Full access to the SIM card from this programming environment.

Both prototypes developed at Eurecom have been defined for iPaqs that were the most appropriated devices at this time. Other prototypes and student projects requiring less cryptography were implemented on standard cell-phones supporting J2ME.

C.3.1 Software Environment.

The Pocket PC operating system was chosen because of its wide use in corporate environments. This choice had a strong impact on the Java virtual machines supported and Bluetooth profiles available. Indeed, only Personal Java (Jeode) offers Java Native Interfaces for the Pocket PC at the time of this development. JNI is a mandatory feature to be able to access Bluetooth from Java without a JSR-82 implementation [JSRa], none of which existed when the development was started. Personal Java is a lightweight Java for PDAs that offers an extension of JDK1.1.8 with Java 2 security. Those restrictions limit the libraries available to parse and transform XML data. For application level security, the Bouncy Castle cryptographic [BC] library was chosen. It provides a Java Cryptographic Extension (JCE) that runs in JDK1.1.8 and offers all the required cryptographic tools.

C.3.2 PAN Communications.

Federations require local communications and discovery mechanisms to work with surrounding devices. Bluetooth is appropriate to implement those concepts and widely available on mobile devices (including cell-phones). The prototype is based on Bluetooth but the architecture is flexible enough to easily replace this local communication media by another one (e.g. WLAN). Bluetooth stack access from Java has been specified [JSRa] but there is no implementation for Pocket PC available at the time of this writing. A dynamic link library (DLL) has thus been implemented to connect devices through the Bluetooth serial profile. Bluetooth connections can be driven from applications thanks to the Java Native Interface (JNI).

C.4 Result: Secure Federation

Figure C.2 gives an overview of the architecture. A Web-mail interface was adopted that makes it easy to integrate XML transformations. On each device, a local HTTP server acts as a proxy when accessing the corporate intranet. This proxy is in charge of transforming XML into HTML, decrypting encrypted parts when the key is available or encapsulating them in applets when the key is unknown, and pushing data to the browser. It has been chosen to use a browser for the graphical user interface because such tools are standardized and available on all considered platforms. The browser loads some applets

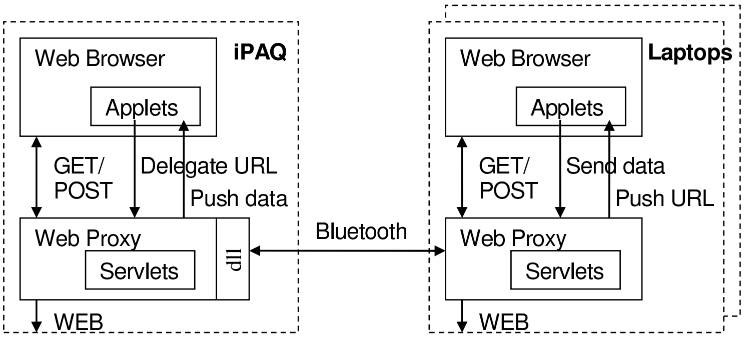
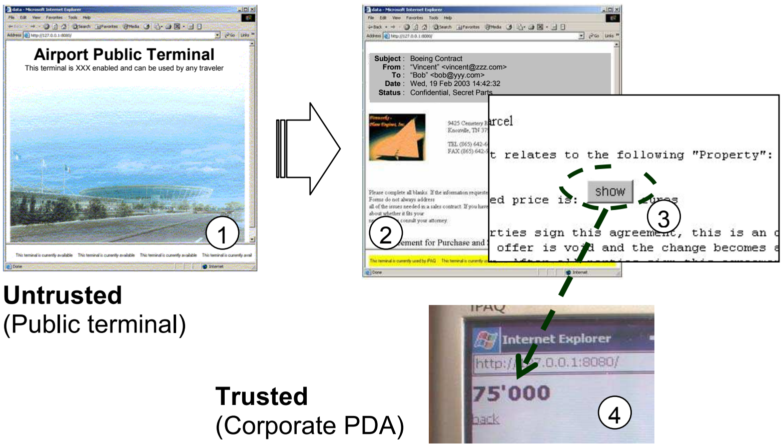


Figure C.2: Prototype architecture

that communicate with the proxy so that pages can be pushed from browsers of other devices. Moreover, a toolbar allows to select discovered devices and to push pages or URL to them (i.e. delegate access right).



Subject : Boeing Contract
From : "Vincent" <vincent@zzz.com>
To : "Bob" <bob@yyy.com>
Date : Wed, 19 Feb 2003 14:42:32
Status : Confidential, Secret Parts

t relates to the following "Property":

ed price is: show euros

rties sign this agreement, this is an
offer is void and the change becomes

75'000

Figure C.3: example of trust-based distribution

Figure C.3 shows some snapshots of the prototype. The Web browser, which runs on a public terminal in an airport, displays a welcome message (see Label 1). When the pervasive salesman is close enough, the terminal is discovered and appears in a toolbar of his PDA browser. He can select a link to an e-mail and delegate it to the discovered terminal (see Label 2). When a part of the document cannot be decrypted by a terminal that is not trusted enough, this part is replaced by an applet displaying a button (see Label 3). Here, the negotiated price of a contract is tagged as confidential and cannot appear on a public terminal: security policies enforcement ensures that a public terminal is not able do decrypt confidential data. The user can press the button to find a member of the federation that is trusted enough. For instance, the confidential part of the document will appear on his PDA (see Label 4). When the same operation is done on a corporate terminal, the whole e-mail (including confidential data) appears.

This approach offers a simple mechanism to adapt a content to the security level. User-friendliness is not sacrificed for security: the salesman does not have to care when delegating the presentation of information to another device because the mandatory security model of his corporation ensures that classified data will never be delivered to an entity distrusted by his company.

Appendix D

Demonstrator 2: Context-Based Trust Establishment

This appendix illustrates the interest of a standardized framework and infrastructure as provided by the WiTness project to build critical business software based on federations. A business application based on a wireless collaborative meeting serves as the demonstrator showing how to secure federations with WiTness tools.

D.1 Application Description

A meeting management application is proposed to demonstrate the capabilities of the WiTness framework to secure federation-based business applications. Meetings are events at which users are typically within a short range. This second demonstrator makes it possible for employees to prepare an agenda and collaborate during a meeting using their devices federated over a wireless network.

The application prototyped provides a complete environment for meeting participants to interact together and proceed along a meeting agenda. Each company taking part in the meeting has at least one designated participant. The application starts with the reception of a meeting agenda by each of these participants. At this stage, each of these participants can delegate one or several of his roles to other employees of his company. This part of the application handles management delegation using WiTness certificates.

When the meeting starts, employees dynamically join a federation made up of their WiTness-enabled personal digital assistants that constitute their digital representation, in particular because it is the device that will perform digital signatures on their behalf. In the demonstrator, the federation is dynamically built upon Bluetooth piconets, but it might also be built using other wireless technologies, in particular WiFi; another alternative for users unable to attend the meeting would be to be connected using GPRS or UMTS and still participate to the meeting and take parts to the votes for instance.

The agenda consists of several items that can be simple discussion item or important decisions that must be ratified by all organizations. The demonstrator makes it possible to keep track of these decisions and to prove, in particular, that a participant did vote for one choice as proven by the signature of his personal device that acts as a personal token. The decisions taken during the meeting can thus be reviewed afterwards and may not be legally challenged. For instance, only employees that receive a manager role, either directly from their company or from a manager, can take part in votes for the orientation of a project. Collaboration during the meeting is also enabled by the ability to share documents, or more precisely grant access rights to other employees for the duration of the meeting for instance. Keeping track of these access rights is again enabled by the use of WiTness certificates.

D.2 Security Requirements

The use of federations for business applications makes it necessary to prevent both external parties from attacking the collaborating partners and one of the users from performing illicit operations or deny having performed an operation. Securing federative business applications thus requires the use of several mechanisms powered by the WiTness framework. Encryption mechanisms are helpful for preserving the confidentiality of communications, essentially from devices outside the federation or from users without eligible roles. Encryption here aims at preventing any leakage outside the trusted machines and devices of the federation, as defined by system administrators of the companies involved in the federation set-up. Meetings generate a need for protecting the information exchanged, for instance against intruders that might be standing outside the meeting room or by the meeting building, especially because of the use of wireless technology.

Signature mechanisms are helpful for providing non-repudiation properties to an operation and thus for tracking operations in an unforgeable manner that stand legally speaking. These mechanisms are binding for the end-users and aim at providing an accountable log of the operations critical to the business application.

Displaying vote information is a particularly critical part of the vote process during a meeting. The important feature of the collaborative meeting application demonstrated is that employees only sign what they see on their personal devices. PDAs have been chosen because they are easy to carry and are thus at least easy to protect against any physical attacks since employees can take them everywhere and at any time. Doing so is even quite critical as an employee's PDA is his digital representation as well as his company's, at least within the roles he was granted.

D.3 Demonstrator Federative Network Architecture

The federation architecture is very much dependent on the underlying wireless technologies. The WiTness framework libraries provide an abstract layer enabling an easier development of mobile business applications. These libraries provide Bluetooth support and the demonstrator was thus developed around this technology. However, working with different implementations of Bluetooth is very difficult as outlined below.

D.3.1 Bluetooth implementations issues: lessons learnt

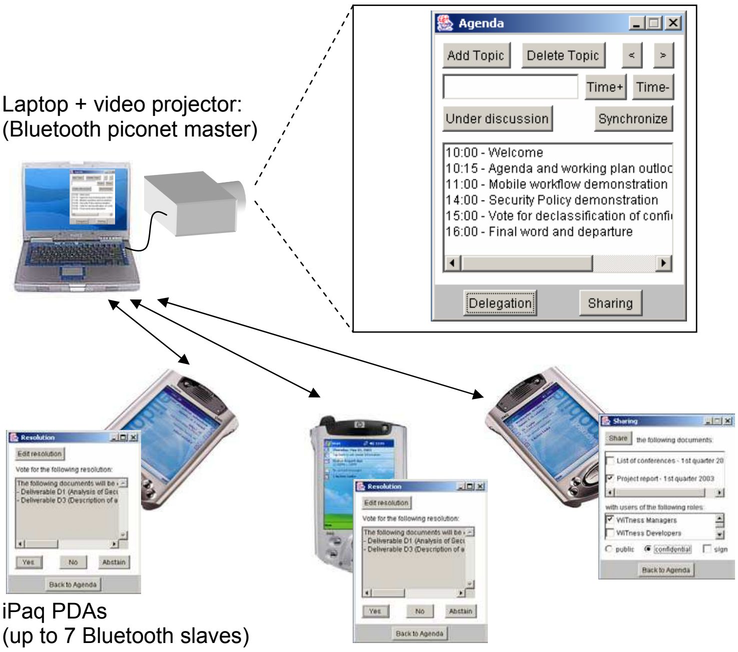
Bluetooth makes it possible to build a network called a piconet in Bluetooth terminology. Communications in such networks are quite different from what is common in 802.11. In particular, communications are deployed in a master/slave fashion, and there is a restriction on the size of the piconet that is a maximum of seven slaves. It is very apparent that these restrictions, which were not very important when Bluetooth was only touted as a replacement technology for cables now represents an important limitation to the construction of wider federations. The concept of scatternet has been developed to alleviate this restriction. The idea to increase the size of a Bluetooth piconet is to chain several piconets, one of the slaves in a piconet being the master of another piconet. However, to date, this concept has received little, if any support.

The Bluetooth technology has also been deployed in a very diverse way among all the devices that claim to be Bluetooth enabled. The specification of the Bluetooth standard defined several layers and associated protocols, as well as several ”profiles” that are made of slices of several of the specified protocols. For instance, the serial profile emulates a physical serial profile as defined in every device it is implemented in; the LAN access profile makes it possible to communicate using TCP/IP on top of Bluetooth. The most promising profile defined for Bluetooth in relation with federations is the PAN profile. Unfortunately, because the specifications for these profiles were elaborated after many Bluetooth enabled devices shipped, most profiles are unavailable or poorly implemented as we found for instance with iPaq Pocket PCs and the LAN access profile. This renders the development of an application, especially if it runs on top of several different devices, quite difficult to achieve.

The serial profile seems to be the most commonly found of these profiles in all devices. However, dealing with serial ports is not simple in every architecture as we experienced with Personal Java on iPaqs. To make matters worse, the decision about who becomes the master is only based on which device speaks first in a piconet. This raises a problem with iPaqs again for these devices may only be connected to one device at the same time and provide no support for scatternet. This means that an iPaq PDA cannot be the master of a full-fledge piconet.

D.3.2 Architecture of the demonstrator

For the implementation of the meeting part of our collaborative agenda demonstrator, we thus decided to use a laptop acting as the master of the piconet federation in order to overcome the limitations of the Bluetooth implementations on PDAs. The role of this laptop (that can be easily taken to a meeting room) is only that of a gateway that relays Bluetooth traffic to and from PDAs. Our demonstrator proves that even with the restrictions of the existent technology, implementing a working federation is possible. Of course, WiFi implementations of federations would be free of such limitations. Finally, the recent adoption of a new standard for Bluetooth, Bluetooth Core specification V1.2, may speed up Bluetooth device replacement and thus address these issues.



10:00 - Welcome
10:15 - Agenda and working plan outloc
11:00 - Mobile workflow demonstration
14:00 - Security Policy demonstration
15:00 - Vote for declassification of confi
16:00 - Final word and departure

Vote for the following resolution:

The following documents will be
- Deliverable D1 (Analysis of Secı
- Deliverable D3 (Description of a

Share the following documents:

□ List of conferences - 1st quarter 20
☑ Project report - 1st quarter 2003

with users of the following roles:

☑ WiTness Managers
□ WiTness Developers

○ public ● confidential □ sign

Figure D.1: Federative groupware

Figure D.1 presents the demonstrator: the meeting leader uses a video projector to show the agenda and proposes to vote an item. Each person present can vote thanks to his/her laptops or PDAs. A secure file sharing is also established. Trust is based on the context, i.e. Bluetooth ensures that members of the group are present in the meeting room, and attribute certificates that define roles and rights.

Appendix E

Mobile Code Protection Can Rely on Trust

This appendix shows how trust can be a pragmatic way to implement security features that are difficult to achieve. For instance, the Platform for Privacy Preferences Project (P3P) offers a pragmatic way to protects users' privacy that does not rely on cryptographic assumptions but on whether the server is trusted. We propose a similar pragmatic approach to deal with two well-known security problems: how to protect the integrity and confidentiality of execution of a piece of code executed by a potentially malicious host and how to protect hosts' integrity from potentially malicious pieces of code.

E.1 Problem Statement

In chapter 6 we have shown that trust evaluation is necessary for defining access control in a federation. In this appendix we extend the notion of trust to control the execution of applications in federations in term of protection of the application and protection of the environment.

Computer users are becoming more and more mobile thanks to the deployment of wireless technologies and to the increasing availability of mobile personal devices. Nomadic computing makes it possible for users to take advantage not only of his handheld or even wearable devices, but also of the appliances in his immediate vicinity, even if they do not belong to him. Enabling an application in such a system means accessing global and local communication infrastructures. For instance, UMTS can be used for communications with remote servers while Bluetooth will enable a pocket device to access surrounding appliances (e.g. printers, screens, sensors).

Nomadic application thus range from over the air access to a classical distributed service provided by a remote server to a set of mobile codes dispersed over close communicating devices, which is generally called a federation of devices. The latter organization helps alleviate the limitations of on-site available communication channels (i.e. restricted bandwidth, long round-trip time, or expensive cost) or the limitations of mobile devices (i.e. lack of computational power, screen size). For instance, a user traveling with a cell-phone will much more efficiently edit a document with a local public terminal than on the keyboard and screen of his phone.

Nomadic or pervasive computing is especially interesting for a mobile corporate workforce, like salesmen visiting their customers. In this context, security becomes a major concern. First, access to the corporate resources and data must be controlled. Second, the safety of the operations performed by a user depends in fact directly on the integrity of execution of a program on devices that will not, for most of them, be owned by the employee or his company, and that may potentially be malicious. This is for instance what happens when a public terminal is used to edit a document that is subsequently signed with the employee's cell-phone (assuming the employee's private key is held by his SIM card). To ensure the *what you see is what you sign* principle, it is necessary to verify the integrity of execution of the editor. Finally, it is necessary to protect public appliances offering some service from hostile users uploading some malicious mobile code in order to attack the environment hosting it. If not enforced, such appliances might be good candidates as Trojan horses of a new kind, unbeknownst to their owner.

Application protection and devices protection have often been discussed in the literature about mobile code security and have proven quite difficult to tackle [ST98, BV99, LBR02, NL98]. In contrast with these works, this appendix suggests that both issues be seen in terms of trust relationships:

- Can the terminal trust this piece of code and give it access to resources?
- Can the user trust this terminal to run some part of an application?

We propose a pragmatic way to evaluate the security-level of pieces of code and devices in the very specific context of business-to-employee (B2E) and business-to-business (B2B) nomadic applications. Access control as well as host and code protection can thus be defined jointly.

E.2 Approaches to Protect Environment and Code

Pervasive computing requires distributing data and pieces of code in a federation of devices that are not always controlled by the user. The problem addressed in this appendix is twofold: on one hand, attacks may be performed by mobile programs against the execution environment and its resources; on the other hand, mobile code and data may be subverted by a malicious execution environment. Here we present mechanisms dedicated to the former issue, which has been widely addressed [LMR00], mechanisms to deal with the latter issue, and some more global approaches.

E.2.1 Protecting Execution Environments

Protecting vital resources against potentially malicious pieces of code has been widely addressed in operating systems and virtual machines. This section lists several approaches and their relevance for securing nomadic B2E or B2B applications.

VM approaches

These approaches address the protection of the environment through the isolation of the potentially malicious code.

Sandbox: The sandbox model is the original security model provided by Java. It offers a severely restricted environment (the sandbox) in which untrusted pieces of code are executed. Local code is trusted and has access to resources (e.g. file system, network) while downloaded code is untrusted and cannot leave the sandbox. This initial mechanism is still widely deployed: it is the default behavior of browsers (i.e. without java plug-in), it is also used in lightweight environments such as J2ME and Personal Java that run on cell-phones and PDAs. Finally, the applet firewall mechanism of Java cards has similar properties. This mechanism has now been superseded by the Java 2 security model.

Java 2 Security Model: The sandbox model has been enhanced with new security features [GMPS97]. There is no more built-in concept defining that local code is trusted and remote code untrusted but each piece of code receives different rights depending on its origin (i.e. URL), on the signature, and recently on the entity who runs the code. The access control to resources is fine-grained and easy to configure. Permissions allow the definition of rights and programmer can define application specific permissions (accessing a smart card, etc.). Security Policies are used to associate permissions to pieces of code. The work described in this appendix and in [BR04] uses and extends those mechanisms.

JavaSeal: JavaSeal [BV99] proposes a security framework to ensure strong security between mobile agents. Confinement mechanism avoids covert channels between agents. Mediation ensures that security controls can be added between pieces of code. Finally, local denial of services attacks are avoided by finely controlling the resources (i.e. memory, computational power) used by agents. This offers interesting security properties that are out of our initial scope. However, JavaSeal could be combined with the approach proposed in this paper to offer a full featured platform for securing mobile code in pervasive computing.

Proof-carrying code

An approach to host protection is to statically type-check the mobile code; the code is then run without any expensive runtime checks. Promising results were obtained in this area by the proof-carrying code work [NL98]. In proof-carrying code, the host first asks for proof that the code respects his security policy before he actually agrees to run it. The code owner sends the program and an accompanying proof, using a set of axioms and rewriting rules. After receiving the code, the host can then check the program with the guidance of the proof. This can be seen as a form of type checking of the program, since the proof is directly derived from it. In proof-carrying code, checking the proof is relatively simple compared to constructing it, thus this technique does not impose much computational burden on the execution environment. However, automating the proof generation is still an open problem.

Two security requirements specific to nomadic systems are not fulfilled by those approaches: a way to define rights of a piece of code in a distributed way that should make possible the delegation of rights between entities in charge of certifying pieces of code; and a mechanism to dynamically change the rights of an application is also necessary.

E.2.2 Protecting Mobile Codes

Protecting nomadic applications often requires protecting the mobile code parts that make it up. Protecting a mobile code against the environment that executes it is notoriously difficult. Verifying the environment trustworthiness is possible with some computer architectures. Other architectures in which this verification is impossible make it necessary to resort to techniques that render the understanding of the behavior of a piece of code extremely difficult in order to ensure its integrity or confidentiality of execution.

Protecting code with trusted platforms

When the device that evaluates a piece of code is trustworthy, integrity and confidentiality of execution are ensured. Two approaches have been undertaken.

Neutral Tamper-Resistant Platform: A straightforward way to ensure that a device can be trusted is proposed by the Trusted Computing Group (TCG) [TCG]. The hardware is tamper-resistant and certified. This hardware can verify whether a certified kernel is running on top of it. This kernel controls the OS, which can check applications. This architecture makes it possible to prove that a given environment is running. As long as all layers are trustworthy (i.e. certified and without implementation errors), it is possible to trust the environment. In other words, an application with some integrity or confidentiality requirements can be executed by any TCG public terminal with the guarantee that the host will not misbehave. For instance, it is possible to ensure that some confidential data will be erased when the user leaves the terminal.

Trusted Tamper-Resistant Module: It is also possible to provide a trusted tamper-resistant hardware that will be in charge of executing applications. For instance telecommunication operators provide SIM cards to their customers in order to have a piece of hardware that is totally under control. For obvious cost reasons, this approach suffers from limited performances. Moreover, it is not realistic to embed a personal hardware in all surrounding devices that can be involved. Finally, this approach only protects the execution of some program but does not protect inputs and outputs, e.g. keyboard and display of the cell-phone bearing the SIM card are still used.

Securing functions in malicious environments

Protecting the evaluation of a mathematical function on a potentially malicious host is a first step towards application protection.

Secure function evaluation has been addressed by many researchers. Sander and Tschudin [ST98] defined a function hiding scheme and focused on non-interactive protocols. In their framework, the confidentiality of function $y = f(x)$ is assured by an encrypting transformation. The authors illustrated the concept with a method that allows computing with encrypted polynomials. The potentially malicious host evaluates the encrypted function and returns an encrypted result. [SYY99] and [LBR02] present non-interactive solutions for secure evaluation of Boolean circuits. Securing a program based on secure functions is not straightforward however, and may again require the use of a personal tamper-proof hardware.

Securing applications in malicious environments

Securing the integrity and confidentiality of a whole application is difficult.

Integrity of Software Execution: Integrity of execution is the possibility for the program owner to verify the correctness of the execution. This problem has been extensively studied for achieving reliability (see for example [WB97] for a survey) but security requirements taking into account possible malicious behavior from the execution environment were not considered. Yee [Yee99] suggested the use of proof based techniques, in which the untrusted host has to forward a proof of the correctness of the execution together with the result. It requires checking only a subset of the proofs in order to assure the correctness of a statement.

Confidentiality of Software Execution: Malicious reverse engineering is an important problem. For instance, Java byte code can easily be decompiled because it retains a large part of the original information and because applications based on powerful libraries are small. Obfuscation aims at transforming an application into one that is functionally identical to the original but that is much more difficult to understand. It is an empirical and mathematically unfounded solution (see [CTL96] for a catalogue of obfuscating transformations).

To summarize, on one hand, hardware solutions to protect pieces of code are difficult to deploy and expensive. Tamper-resistant modules are necessary to protect private keys but it is not always affordable to have a secure hardware that protects the execution of a whole application. Moreover, the process for certifying hardware is complex. On the other hand, there is no software solution to fully ensure integrity and/or confidentiality protection of a piece of code running on a malicious host. Indeed, all approaches presented in this section are restricted to a set of functions, are computationally expensive, and/or cannot be proven secure.

E.2.3 Trust-Based Application Protection

Rather than focusing on mechanisms to tackle either the mobile code side or the environment side, this appendix proposes a system wide and pragmatic mechanism common to both the protection of code and environment. Environment and code protection can be based on trust, i.e. authorizations and/or roles of application developers and security-level of runtime environments.

Approaches based on distributed policies for managing trust [KFP01, BFK99] do not take into account the security-level of execution environments. It is assumed that policies are always enforced and it is not possible to recognize an untrusted device from a trusted one. Policies are thus not sufficient for enforcing the protection of applications. We however envision policies to offer a flexible and high level specification of trust management.

In the business context described in this appendix, trust is based on *a priori* knowledge. Recommendations, results of previous interactions, or even contextual information might further be used to extend this knowledge.

E.3 Pragmatic Approach

We propose a framework for protecting the pieces of code, i.e. verifying the security-level of environment before allowing distribution, and protecting the environment, i.e. verifying that pieces of code are authorized to access resources, be they a database or a network connection. More details on this approach, which was developed within student projects, can be found in [BR04].

E.3.1 Nomadic System Organization

Figure E.1 shows how code distribution is done: different parts of an application are tagged according to the security requirements and the security-level (SL) of each device is evaluated. For instance, the signature related operation of an application has to be done in a trusted-enough environment. Each piece of code receives short-term authorization to access resources. For instance, a word processor can call the signature function but a game cannot.

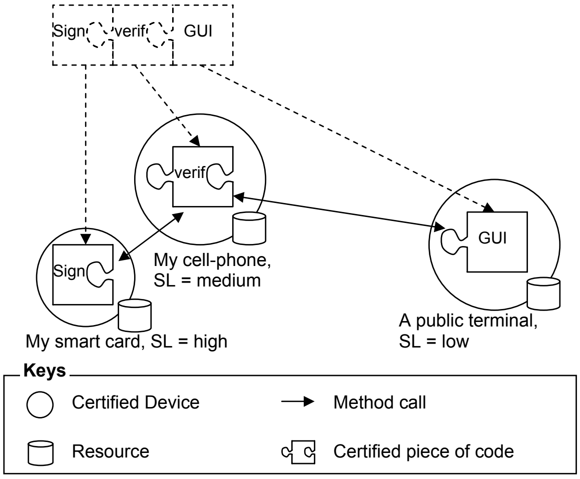


Figure E.1: General overview: certified pieces of code within certified devices

Devices, which are not managed by the user and whose trustworthiness may be questioned, may anyway have to deal with confidential data. Moreover, in order to enable flexible services, it is necessary to let users upload pieces of code (or *applets*) to surrounding devices. Using trust information when deploying the application implies new constraints when distributing data and code. We focus on the implications of this environment for satisfying to *data integrity*, *data confidentiality*, *integrity of execution* and *confidentiality of execution*. In this model, integrity of execution means that servers do not alter the execution of the application and surreptitiously modify its results. Confidentiality of execution aims at preventing the disclosure of program semantics.

E.3.2 Defining Trust Relationships

WiTness attribute certificates have been chosen to formally define relationships and authorizations between the involved parties. Rights can be delegated if the certificate allows so and delegation can be performed in a local way without the need to connect to a centralized authority: each user behaves as a local authority for attribute certificates. Delegated credentials have a short lifetime, thus rendering the use of centralized revocation lists unnecessary, and permitting a local validation of the certificate chain. For long-lasting capabilities, revocation lists are envisaged. Attribute certificates are used to store a different type of information: for an employee, it can consist of his role or personal rights; for a device, information about its security-level and the company it belongs to may be provided.

Environment Protection: The Java 2 security model relies on signed pieces of code and thus is identity based. Only mechanisms similar to access control list are available to protect resources. [MR00] suggests that instead of signing pieces of code and associating permissions with signers, manipulating capabilities such as chain of authorization certificates associated with pieces of code is required to handle multiple domains in a manageable manner. We use WiTness certificates associated to pieces of code. A meta-object protocol (MOP) [KdRB91] is used to intercept all method calls done by this piece of code. Like this, it is easy to dynamically modify the authorizations when a new certificate chain is available.

Code Protection: We propose to distribute data and code according to the security-level of federated devices. Securing federations thus becomes evaluating the security-level of each platform that takes part in the federation. The evaluation is not easy to achieve in general: if a person makes use of a terminal in a public place, it is impossible to assume that the terminal is trusted in any way without some additional information that makes up the trust model. In general, there is no relation that can be exploited between the user or his company and the owner of the terminal. B2E and B2B assumptions provide a clear trust model and allow validating whether a given device is trustworthy (e.g. managed by a partner company, patches are regularly applied). This information is used to distribute code and data according to the security-level of each federated device. Security-levels are defined by a chain of certificates. It is possible to increase the granularity of security-levels by defining new semantics taking into account project names, groups, etc.

Bibliography

[AH00] T. Austin and D. Huaman. *PKI.* John Wiley and Sons, 2000.

[ake] Akenti: a security model and architecture to provide scalable security services in highly distributed network environments `http://www-itg.lbl.gov/Akenti/`.

[And01] Ross Anderson. *Security Engineering: A Guide to Building Dependable distributed Systems.* John Wiley and Sons, 2001.

[AS02] A. Alkassar and C. Stuble. Towards secure iff: preventing mafia fraud attacks, [pdf]. In *Proceedings of MILCOM 2002*, volume 2, pages 1139–1144, October 2002.

[AT99] G. Ateniese and G. Tsudik. Some open issues and new directions in group signatures, [pdf]. In *Proceedings of Financial Cryptography'99*, volume 1648 of *LNCS*, pages 196–211. Springer-Verlag, 1999.

[BB03] K. Bicakci and N. Baykal. Saots: A new efficient server assisted signature schema for pervasive computing. In *proceedings of Security in Pervasive Computing SPC'03*, March 2003.

[BB04a] A. Beaufour and P. Bonnet. Personal servers as digital keys, [pdf]. In *the Second IEEE Annual Conference on Pervasive Computing and Communications (PerCom'04)*, pages 319–328, 2004.

[BB04b] L. Bussard and W. Bagga. Distance-bounding proof of knowledge protocols to avoid terrorist fraud attacks, [pdf]. Technical Report RR-04-109, May 2004.

[BBD$^+$91] S. Bengio, G. Brassard, Y. Desmedt, C. Goutier, and J.J. Quisquater. Secure implementation of identification systems. *Journal of Cryptology*, 4(3):175–183, 1991.

[BC] *Legion of Bouncy Castle*, Java Crypto APIs (JCA), `http://www.bouncycastle.org/`.

[BC93] S. Brands and D. Chaum. Distance-bounding protocols (extended abstract), [pdf]. In *Proceedings of EUROCRYPT 93*, volume 765 of *LNCS*, pages 23–27. Springer-Verlag, May 1993.

[BCC$^+$04] L. Bussard, J. Claessens, S. Crosta, Y. Roudier, and A. Zugenmaier. Can we take this off-line? how to deal with credentials in federations without global connectivity, [pdf]. Technical Report RR-04-105, May 2004.

[BFI99] M. Blaze, J. Feigenbaum, and J. Ioannidis. The keynote trust-management system version 2, [pdf]. Technical Report Request for Comments: 2704, Network Working Group, 1999.

[BFK99] Matt Blaze, Joan Feigenbaum, and Angelos D. Keromytis. The role of trust management in distributed systems security, [pdf]. In *Secure Internet Programming*, pages 185–210, 1999.

[BFL96] Matt Blaze, Joan Feigenbaum, and Jack Lacy. Decentralized trust management, [pdf]. In *Proceedings 1996 IEEE Symposium on Security and Privacy*, number 164–173, May 1996.

[BHE00] N. Bulusu, J. Heidemann, and D. Estrin. Gps-less low-cost outdoor localization for very small devices, [pdf]. *IEEE Personal Communications*, 7(5):28–34, 2000.

[BHKK$^+$04] L. Bussard, J. Haller, R. Kilian-Kehr, J. Posegga, P. Robinson, Y. Roudier, and T. Walter. Secure mobile business applications – framework, architecture and implementation. Submitted for publication in a journal, 2004.

[BK] J. Barton and T. Kindberg. The challenges and opportunities of integrating the physical world and networked systems, [pdf]. Technical report.

[Bla96] Matt Blaze. High-bandwidth encryption with low-bandwidth smartcards, [pdf]. In *Fast Software Encryption*, volume 1039 of *LNCS*, pages 33–40, 1996.

[BM04a] L. Bussard and R. Molva. Establishing trust with privacy, [pdf], April 2004. To appear in proceedings of the twelve international workshop on security protocols.

[BM04b] L. Bussard and R. Molva. One-time capabilities for authorizations without trust, [pdf]. In *Proceedings of the second IEEE conference on Pervasive Computing and Communications (PerCom'04)*, pages 351–355, March 2004.

[BMR04a] L. Bussard, R. Molva, and Y. Roudier. Combining history-based trust establishment with distance-bounding protocols, [pdf]. Technical Report RR-04-100, April 2004.

[BMR04b] L. Bussard, R. Molva, and Y. Roudier. History-based signature or how to trust anonymous documents, [pdf]. In *Proceedings of the Second Conference on Trust Management (iTrust'2004)*, volume 2995 of *LNCS*, pages 78–92. Springer, March 2004.

[Bor00] Gaetano Borriello. The challenges to invisible computing, [pdf]. *IEEE Computer*, 33(11):123–125, November 2000.

[BP00] P. Bahl and V.N. Padmanabhan. Radar: An in-building rf-based user location and tracking system, [pdf]. In *INFOCOM*, volume 2, pages 775–784, 2000.

[BQ95] P. Bégiun and J.J. Quisquater. Fast server-aided rsa signatures secure against active attacks, [pdf]. In *proceedings of CRYPTO'95*, pages 57–69, 1995.

[BR02] L. Bussard and Y. Roudier. Authentication in ubiquitous computing, [pdf], 2002. Workshop on Security in Ubiquitous Computing at UBICOMP'2002.

[BR03a] L. Bussard and Y. Roudier. Background signature for sensor networks, [pdf]. Technical Report RR-03-076, June 2003.

[BR03b] L. Bussard and Y. Roudier. Embedding distance-bounding protocols within intuitive interactions, [pdf]. In *Proceedings of Conference on Security in Pervasive Computing (SPC'2003)*, volume 2802 of *LNCS*, pages 143–156. Springer, March 2003.

[BR04] L. Bussard and Y. Roudier. Protecting applications and devices in nomadic business environments, [pdf]. In *Proceedings of 3rd Conference on Security and Network Architectures (SAR'04)*, pages 243–252, June 2004.

[Bra93] Stefan Brands. An efficient off-line electronic cash system based on the representation problem., [pdf]. Technical report, 1993.

[Bra00] Stefan A. Brands. *Rethinking public key infrastructures and digital certificates: Building in privacy.* MIT Press, 2000.

[Bra02] Stefan Brands. A technical overview of digital credentials, [pdf]. Technical report, Credentica, 2002.

[Bri03] R. Bridgelall. Enabling mobile commerce through pervasive communications with ubiquitous rf tags, [pdf]. In *Proceedings of IEEE Wireless Communications and Networking Conference (WCNC'03)*, volume 3, pages 2041–2046, 2003.

[BRKC03] L. Bussard, Y. Roudier, R. Kilian Kehr, and S. Crosta. Trust and authorization in pervasive b2e scenarios, [pdf]. In *Proceedings of the 6th Information Security Conference (ISC'03)*, volume 2851 of *LNCS*, pages 295–309. Springer, October 2003.

[BRM04] L. Bussard, Y. Roudier, and R. Molva. Untraceable secret credentials: Trust establishment with privacy, [pdf]. In *Proceedings of the Workshop on Pervasive Computing and Communications Security (PerSec'04) at PerCom'04*, pages 122–126, March 2004.

[BSSW02] D. Balfanz, D.K. Smetters, P. Stewart, and H. Chi Wong. Talking to strangers: Authentication in adhoc wireless networks, [pdf]. In *Symposium on Network and Distributed Systems Security (NDSS '02)*, February 2002.

[BV99] Ciaran Bryce and Jan Vitek. The JavaSeal mobile agent kernel, [pdf]. In *First International Symposium on Agent Systems and Applications (ASA'99)/Third International Symposium on Mobile Agents (MA'99)*, Palm Springs, CA, USA, 1999.

[CD00a] J. Camenisch and I.B. Damgard. Verifiable encryption, group encryption, and their applications to group signatures and signature sharing schemes, [pdf]. In *Advances in Cryptology - Asiacrypt 2000*, volume 1976 of *LNCS*, pages 331–345. Springer-Verlag, 2000.

[CD00b] D. Caswell and P. Debaty. Creating web representations for places, [pdf]. In *Proceedings of Handheld and Ubiquitous Computing: Second International Symposium, HUC 2000*, volume 1927 of *LNCS*, page 114. Springer-Verlag, January 2000.

[CFN89] D. Chaum, A. Fiat, and M. Naor. Untraceable electronic cash (extended abstract), [pdf]. In *Advances in Cryptology – CRYPTO '88 Proceedings*, volume 403 of *LNCS*, pages 319–327. Springer-Verlag, 1989.

[CFS01] N. Courtois, M. Finiasz, and N. Sendrier. How to achieve a mceliece-based digital signature scheme, [pdf]. In *Advances in Cryptology - ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 157–174, 2001.

[CGKS95] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private information retrieval, [pdf]. In *IEEE Symposium on Foundations of Computer Science*, pages 41–50, 1995.

[CGRZ03] S. Creese, M. Goldsmith, B. Roscoe, and I. Zakiuddin. Authentication for pervasive computing, [pdf]. In *Proceedings of the First International Conference on Security in Pervasive Computing*, LNCS. Springer, 2003.

[CGS$^+$03] V. Cahill, E. Gray, J.-M. Seigneur, C.D. Jensen, Yong Chen, B. Shand, N. Dimmock, A. Twigg, J. Bacon, C. English, W. Wagealla, S. Terzis, P. Nixon, G. Di Marzo Serugendo, C. Bryce, M. Carbone, K. Krukow, and M. Nielson. Using trust for secure collaboration in uncertain environments, [pdf]. *IEEE Pervasive Computing*, 2(3):52–61, July 2003.

[CH02] J. Camenisch and E. V. Herreweghen. Design and implementation of the idemix anonymous credential system, [pdf]. In *Proc. 9th ACM conference on Computer and Communications Security.* ACM Press, 2002.

[Cha81] David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms, [pdf]. *Communications of the ACM*, 4(2), February 1981.

[CL01] J. Camenisch and A. Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation, [pdf]. *Lecture Notes in Computer Science*, 2045, 2001.

[CLS$^+$01] M.J. Covington, W. Long, S. Srinivasan, A.K. Dev, M. Ahamad, and G.D. Abowd. Securing context-aware applications using environment roles, [pdf]. In *Proceedings of the Sixth ACM Symposium on Access control models and technologies*, May 2001.

[CM98] J. Camenisch and M. Michels. A group signature scheme based on an rsa-variant, [pdf]. Technical Report RS-98-27, BRICS, University of Aarhus, 1998. Preliminary version in ASIACRYPT'98, volume 1514 of LNCS, pages 160–174, Springer Verlag.

[CM04] D. Cvrček and V. Matyáš. Pseudonymity in the light of evidence-based trust, [pdf], April 2004. To appear in proceedings of the twelve international workshop on security protocols.

[CMA00] M.J. Covington, M.J. Moyer, and M. Ahamad. Generalized role-based access control for securing future applications, [pdf]. In *23rd National Information Systems Security Conference*, 2000.

[CN02] M. Corner and B. Noble. Zero-interaction authentication, [pdf]. In *Proceedings of Conference on Mobile Computing and Networking (MobiCom)*, September 2002.

[CR82] D. Chaum and R.L. Rivest. Blind signatures for untraceable payments, [pdf]. In *Advances in Cryptology, Proceedings of Crypto 82*, LNCS, pages 199–203, 1982.

[CS97] J. L. Camenisch and M. A. Stadler. Efficient group signature schemes for large groups, [pdf]. In *Advances in Cryptology – CRYPTO '97 Proceedings*, volume 1294 of *LNCS*, pages 410–424. Springer-Verlag, 1997.

[CTL96] C. Collberg, C. Thomborson, and D. Low. A taxonomy of obfuscating transformations, [pdf]. Technical Report Technical Report 148, Department of Computer Science, University of Auckland, 1996.

[Cur98] Stephen M. Curry. An introduction to the java ring, [pdf]. *Java World*, 1998.

[DDLS01] N. Damianou, N. Dulay, E. Lupu, and M Sloman. The ponder specification language, [pdf]. In *Workshop on Policies for Distributed Systems and Networks (Policy2001)*, 2001.

[Dem04] Robert Demolombe. Reasoning about trust: A formal logical framework. In *Proceedings of Second International Conference on Trust Management (iTrust'04)*, volume 2995, pages 291–303. LNCS, 2004.

[Des88] Yvo Desmedt. Major security problems with the 'unforgeable' (feige)- at-shamir proofs of identity and how to overcome them. In *Proceedings of SecuriCom '88*, 1988.

[DFHM01] R. Dingledine, M.J. Freedman, D. Hopwood, and D. Molnar. A reputation system to increase mix-net reliability, [pdf]. In *Proceedings of the 4th International Workshop on Information Hiding*, volume 2137, pages 126–141. LNCS, 2001.

[DM96] D.E. Denning and P. F. MacDoran. Location-based authentication: Grounding cyberspace for better security, [pdf]. *Computer Fraud and Security*, February 1996.

[dsi] *XML Digital Signature.* W3C Recommendation, 12 February 2002, `http://www.w3.org/Signature/`.

[EFL+99] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen. Rfc 2693 – spki certificate theory, [pdf], 1999.

[EGM96] Shimon Even, Oded Goldreich, and Silvio Micali. On-line/off-line digital signatures, [pdf]. *Journal of Cryptology*, 9(1):35–67, 1996.

[ENT+02] C. English, P. Nixon, S. Terzis, A. McGettrick, and H. Lowe. Dynamic trust models for ubiquitous computing environments, [pdf]. In *Workshop on Security in Ubiquitous Computing at UBICOMP'2002*, 2002.

[EWN+03] C. English, W. Wagealla, P. Nixon, S.Terzis, A. McGettrick, and H. Lowe. Trusting collaboration in global computing, [pdf]. In *Proceedings of the First International Conference on trust management*, volume 2692 of *LNCS*, May 2003.

[Fer94] N. Ferguson. Single term off-line coins, [pdf]. In *Advances in Cryptology—EUROCRYPT '93*, volume 765 of *LNCS*, pages 318–328. Springer-Verlag, 1994.

[FSLS03] D.F. Ferguson, T. Storey, B. Lovering, and J. Shewchuk. Secure, reliable, transacted web services: Architecture and composition., [pdf]. Technical report, IBM and Microsoft, September 2003.

[Gar95] Simson Garfinkel. *PGP : Pretty Good Privacy.* International Thomson Publishing, 1995.

[GBEE02] L. Girod, V. Bychkobskiy, J. Elson, and D. Estrin. Locating tiny sensors in time and space: A case study, [pdf]. In *Proceedings of ICCD'02*, 2002.

[Ger04] Jon Gertner. The very, very personal is the political, [pdf]. *The New York Times Magazine*, February 2004.

[GLHB03] D. Graumann, W. Lara, J. Hightower, and G. Borriello. Real-world implementation of the location stack: The universal location framework, [pdf]. In *Proceedings of the 5th IEEE Workshop on Mobile Computing Systems & Applications (WMCSA 2003)*, pages 122–128. IEEE Computer Society Press, October 2003.

[GMPS97] L. Gong, M. Mueller, H. Prafullchandra, and R. Schemers. Going beyond the sandbox: An overview of the new security architecture in the Java Development Kit 1.2, [pdf]. In *USENIX Symposium on Internet Technologies and Systems*, pages 103–112, Monterey, CA, 1997.

[GS00] Tyrone Grandison and Morris Sloman. A survey of trust in internet applications, [pdf], 2000.

[GW98] E. Gabber and A. Wool. How to prove where you are: Tracking the location of customer equipment, [pdf]. In *Proceedings of the 5th ACM Conference on Computer and Communications Security*, pages 142–149, November 1998.

[HB01] Jeffrey Hightower and Gaetano Borriello. Location systems for ubiquitous computing, [pdf]. *IEEE Computer*, 2001.

[HHS$^+$99] A. Harter, A. Hopper, P. Steggles, A. Ward, and P. Webster. The anatomy of a context-aware application, [pdf]. In *Mobile Computing and Networking*, pages 59–68, 1999.

[HMS$^+$01] L.E. Holmquist, F. Mattern, B. Schiele, P. Alahuhta, M. Beigl, and H-W. Gellersen. Smart-its friends: A technique for users to easily establish connections between smart artefacts, [pdf]. In *Proceedings of UbiComp 2001*, 2001.

[HPJ03] Yih-Chun Hu, A. Perrig, and D.B. Johnson. Packet leashes: a defense against wormhole attacks in wireless networks, [pdf]. In *Proceedings of INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 3, pages 1976–1986, March 2003.

[ICa] *ICare*, Trust Infrastructure over Internet and Mobile Networks, `http://www.cert-i-care.org`.

[Ing03] D. Ingram. Trust-based filtering for augmented reality, [pdf]. In *Proceedings of the First International Conference on Trust Management*, volume 2692. LNCS, May 2003.

[IT00] ITU-T. Recommendation x.509: The directory - public-key and attribute certificate frameworks, [pdf]. Technical Report X.509, ITU-T, 2000.

[IU97] Hiroshi Ishii and Brygg Ullmer. Tangible bits: Towards seamless interfaces between people, bits and atoms, [pdf]. In *CHI*, pages 234–241, 1997.

[JP04] A. Josang and S. Lo Presti. Analysing the relationship between risk and trust, [pdf]. In *Proceedings of Second International Conference on Trust Management (iTrust'04)*, volume 2995, pages 135–145. LNCS, 2004.

[JSRa] *JSR 82* Java APIs for Bluetooth, `http://www.jcp.org/en/jsr/detail?id=82`.

[JSRb] *JSR 177* Security and Trust Services API for J2ME, `http://www.jcp.org/en/jsr/detail?id=177`.

[KdRB91] Kiczales, des Rivieres, and Bobrow. *The Art of the Metaobject Protocol.* MIT Press, 1991.

[KFJ01] Lalana Kagal, Tim Finin, and Anupam Joshi. Trust-based security in pervasive computing environments, [pdf]. *IEEE Computer*, pages 154–157, December 2001.

[KFP01] L. Kagal, T. Finin, and Y. Peng. A framework for distributed trust management, [pdf]. In *Workshop on Autonomy, Delegation and Control*, 2001.

[KH00] Hiroaki Koshima and Joseph Hoshen. Personal locator services emerge, [pdf]. *IEEE Spectrum*, 2000.

[KHM$^+$00] J. Krumm, S. Harris, B. Meyers, B. Brumitt, M. Hale, and S. Shafer. Multicamera multi-person tracking for easyliving, [pdf]. In *IEEE Workshop on Visual Surveillance*, 2000.

[KKP99] J. M. Kahn, R. H. Katz, and K. S. J. Pister. Next century challenges: Mobile networking for 'smart dust', [pdf]. In *MOBICOM*, pages 271–278, 1999.

[Kob94] Neal I. Koblitz. *A Course in Number Theory and Cryptography.* Springer, 1994.

[KP03] M. Kinateder and S. Pearson. A privacy-enhanced peer-to-peer reputation system, [pdf]. In *Proceedings of the 4th International Conference on Electronic Commerce and Web Technologies (EC-Web'03)*, volume 2738, pages 206–215. LNCS, 2003.

[KZ03] T. Kindberg and K. Zhang. Validating and securing spontaneous associations between wireless devices, [pdf]. In *Proceedings 6th Information Security Conference (ISC03)*, volume 765, pages 44–53, 2003.

[KZS02] T. Kindberg, K. Zhang, and N. Shankar. Context authentication using constrained channels, [pdf]. In *Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)*, pages 14–21, June 2002.

[LBR02] S. Loureiro, L. Bussard, and Y. Roudier. Extending tamper-proof hardware security to untrusted execution environments, [pdf]. In *Proceedings of the Fifth Smart Card Research and Advanced Application Conference (CARDIS'02) - USENIX - IFIP working group 8.8 (smart cards)*, pages 111–124, November 2002.

[LMR00] S. Loureiro, R. Molva, and Y. Roudier. Mobile code security, [pdf]. In *ISYPAR 2000, (4ème Ecole d' Informatique des Systèmes Parallèles et Répartis)*, 2000.

[LR98] A. Lysyanskaya and Z. Ramzan. Group blind digital signatures: A scalable solution to electronic cash, [pdf]. In *Financial Cryptography*, pages 184–197, 1998.

[McC01] Joseph F. McCarthy. The virtual world gets physical: Perspectives on personalization, [pdf]. *IEEE Internet Computing*, pages 48–53, December 2001.

[McE78] R. McEliece. A public-key cryptosystem based on algebraic coding theory. Technical Report DSN Progress Report, DIn Jet Propulsion Lab., 1978.

[Mer00] Johannes Merkle. Multi-round passive attacks on server-aided RSA protocols, [pdf]. In *ACM Conference on Computer and Communications Security*, pages 102–107, 2000.

[Mic96] S. Micali. Efficient certificate revocation, [pdf]. Technical Report MIT/LCS/TM-542b, 1996.

[MM02] P. Michiardi and R. Molva. Core: A collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks, [pdf]. In *IFIP - Communication and Multimedia Security Conference*, 2002.

[MR00] R. Molva and Y. Roudier. A distributed access control model for Java, [pdf]. In *6th European Symposium on Research in Computer Security (ESORICS)*, number 1895, pages 291–308, 2000.

[MRCM02] J. Al Muhtadi, A. Ranganathan, R. Campbell, and D. Mickunas. A flexible, privacy-preserving authentication framework for ubiquitous computing environments, [pdf]. In *International Workshop on Smart Appliances and Wearable Computing (IWSAWC 2002)*, 2002.

[MVO96] Alfred J. Menezes, Scott A. Vanstone, and Paul C. Van Oorschot. *Handbook of Applied Cryptography*, [pdf]. CRC Press, 1996.

[NKR$^+$02] C. Narayanaswami, N. Kamijoh, M. Raghunath, T. Inoue, T. Cipolla, J. Sanford, E. Schlig, S. Venkiteswaran, D. Guniguntala, V. Kulkarni, and K. Yamazaki. Ibm's linux watch, the challenge of miniaturization, [pdf]. *IEEE Computer*, 35(1):33–41, January 2002.

[NL98] George C. Necula and Peter Lee. Safe, untrusted agents using proof-carrying code, [pdf]. *Lecture Notes in Computer Science*, 1419, 1998.

[NMV99] K.Q. Nguyen, Yi Mu, and V.Varadharajan. Divertible zero-knowledge proof of polynomial relations and blind group signature. In *Information Security and Privacy, Proceedings of ACISP'99*, 1999.

[OA00] R.J. Orr and G.D. Abowd. The smart floor: A mechanism for natural user identification and tracking, [pdf]. Technical Report GVU Technical Report GIT-GVU-00-02, 2000.

[OSM] K. Otani, H. Sugano, and M. Mitsuoka. Capability card: An attribute certificate in xml, [pdf]. Expired Internet Draft, 18 Nov. 1998.

[P3P] World Wide Web Consortium, *Platform for Privacy Preferences Project (P3P)*, 2004, `http://www.w3.org/P3P/`.

[PB99] C. Pavlovski and C. Boyd. Efficient batch signature generation using tree structures, [pdf]. In *International Workshop on Cryptographic Techniques and E-Commerce (CrypTEC'99)*, pages 70–77, 1999.

[PCB00] N.B. Priyantha, A. Chakraborty, and H. Balakrishnan. The cricket location-support system, [pdf]. In *Mobile Computing and Networking*, pages 32–43, 2000.

[PHS03] J. Pieprzyk, T. Hardjono, and J. Seberry. *Fundamentals of Computer Security*. Springer, 2003.

[PK01a] Andreas Pfitzmann and Marit Köhntopp. Anonymity, unobservability, and pseudonymity – a proposal for terminology, [pdf]. In *Designing Privacy Enhancing Technologies – International Workshop on Design Issues in Anonymity and Unobservability 2000*, volume 2009 of *LNCS*, pages 1–9. Springer-Verlag, 2001. updated version in [PK01b].

[PK01b] Andreas Pfitzmann and Marit Köhntopp. Anonymity, unobservability, and pseudonymity – a proposal for terminology v.12, [pdf]. Technical report, 2001.

[PSW$^+$01] A. Perrig, R. Szewczyk, V. Wen, D.E. Culler, and J.D. Tygar. SPINS: security protocols for sensor netowrks, [pdf]. In *Mobile Computing and Networking*, pages 189–199, 2001.

[Ram99] Zulfikar Amin Ramzan. Group blind digital signatures: Theory and applications, [pdf], 1999.

[RBM03] A. Rezgui, A. Bouguettaya, and Z. Malik. A reputation-based approach to preserving privacy in web services, [pdf]. In *4th VLDB Workshop on Technologies for E-Services (TES'03)*, pages 91–103. LNCS, 2003.

[Ric00] G.G. Richard. Service advertisement and discovery: enabling universal device cooperation, [pdf]. *IEEE Internet Computing*, 4(5):18–26, 2000.

[Riv98] Ronald L. Rivest. Can we eliminate certificate revocations lists?, [pdf]. In *proceedings of the Conference on Financial Cryptography*, pages 178–183, 1998.

[RNP03] M. Raghunath, C. Narayanaswami, and C. Pinhanez. Fostering a symbiotic handheld environment, [pdf]. *IEEE Computer*, 36(9):56–65, September 2003.

[SA99] Frank Stajano and Ross J. Anderson. The resurrecting duckling: Security issues for ad-hoc wireless networks, [pdf]. In *Security Protocols Workshop*, pages 172–194, 1999.

[SAM] *Security Assertion Markup Language (SAML 1.0)*. OASIS standard, 5-Nov-2002, http://www.oasis-open.org/committees/security/.

[Sat01] M. Satyanarayanan. Pervasive computing: Vision and challenges, [pdf]. *IEEE Personal Communications*, 8(4):10–17, August 2001.

[SCFY96] R.S. Sandhu, E.J. Coyne, H.L. Feinstein, and C.E. Youman. Role based access control models, [pdf]. *IEEE Computer*, 2, February 1996.

[Sch89] C. P. Schnorr. Efficient identification and signatures for smart cards, [pdf]. In *Advances in Cryptology - CRYPTO'89 Proceedings*, volume 435 of *LNCS*, pages 239–252. Springer-Verlag, 1989.

[SDA99] Daniel Salber, Anind K. Dey, and Gregory D. Abowd. The context toolkit: Aiding the development of context-enabled applications, [pdf]. In *CHI*, pages 434–441, 1999.

[SDB03] B. Shand, N. Dimmock, and J. Bacon. Trust for ubiquitous, transparent collaboration, [pdf]. In *Proceedings of the First IEEE International Conference on Pervasive Computing and Communications, (PerCom'03)*, pages 153–160, 2003.

[SFJ$^+$03] J.M. Seigneur, S. Farrell, C.D. Jensen, E. Gray, and Y. Chen. End-to-end trust starts with recognition, [pdf]. In *Proceedings of Conference on Security in Pervasive Computing (SPC'2003)*, March 2003.

[She00] Mostafa H. Sherif. *Protocols for Secure Electronic Commerce.* CRC Press, 2000.

[sim] GSM 11.11, *Digital cellular telecommunications system (Phase 2+); Specification of the Subscriber Identity Module - Mobile Equipment (SIM - ME) interface*, 1999.

[Sin00] Simon Singh. *The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography.* Anchor, 2000.

[SSW03] N. Sastry, U. Shankar, and D. Wagner. Secure verification of location claims, [pdf]. In *Proceedings of the 2003 ACM workshop on Wireless security*, 2003.

[ST98] Tomas Sander and Christian F. Tschudin. On software protection via function hiding, [pdf]. *Lecture Notes in Computer Science*, 1525:111–123, 1998.

[Sta00] Frank Stajano. The resurrecting duckling - what next?, [pdf]. In *Security Protocols Workshop*, pages 204–214, 2000.

[Sta02] Frank Stajano. *Security for Ubiquitous Computing.* John Wiley and Sons, 2002.

[Sti02] Douglas Stinson. What the heck is a zero-knowledge proof of knowledge, anyway, and what does it mean?, [pdf]. Technical report, unpublished note, 2002.

[SYY99] T. Sander, A. Young, and M. Yung. Non-interactive cryptocomputing for nc1, [pdf]. In *40th Annual Symposium on Foundations of Computer Science 99*, pages 554–566, 1999.

[TCG] *Trusted Computing Group (TCG)*, `https://www.trustedcomputinggroup.org/home`.

[WB97] H. Wasserman and M. Blum. Software reliability via run-time result-checking, [pdf]. *Journal of the ACM*, 44(6):826–849, 1997.

[WBRR04] T. Walter, L. Bussard, P. Robinson, and Y. Roudier. Security and trust issues in ubiquitous environments - the business-to-employee dimension, [pdf], 2004. Workshop on Ubiquitous Services and Networking in at SAINT'2004.

[WdW00] L. Willenborg and T. de Waal. *Elements of Statistical Disclosure Control*, volume 155 of *Lecture Notes in Statistics*. Springer Verlag, 2000.

[Wei91] Mark Weiser. The computer for the twenty-first century, [pdf]. *Scientific American*, pages 94–100, September 1991.

[WF] B. Waters and E. Felten. Proving the location of tamper-resistant devices, [pdf]. Technical report.

[WHFG92] R. Want, A. Hopper, V. Falcao, and J. Gibbons. The active badge location system, [pdf]. *ACM Transactions on Information Systems*, 10:91–102, 1992.

[XAC] *eXtensible Access Control Markup Language (XACML 1.0)*. OASIS Standard, 6 Feb. 2003, `http://www.oasis-open.org/committees/xacml/`.

[XEn] *XML Encryption*. W3C Recommendation, 10 December 2002, `http://www.w3.org/Encryption/`.

[Yee99] Bennet S. Yee. A sanctuary for mobile agents, [pdf]. In *Secure Internet Programming*, pages 261–273, 1999.

Résumé

"Ma patrie, c'est la langue française."

– Albert Camus

Ce chapitre propose une traduction en français des principaux résultats présentés dans les chapitres précédents de ce mémoire.

Introduction

En 1991, Marc Weiser prévoyait pour le 21$^{\text{ème}}$ siècle la disparition des ordinateurs de bureau. Il décrivait une extinction massive de ces dinosaures qui allaient être remplacés par des processeurs de plus en plus petits intégrés dans la plupart des objets qui nous entourent. Dans sa vision, les utilisateurs ne devraient plus être conscients d'interagir avec des ordinateurs. Cette tendance est communément appelée "informatique diffuse" (*pervasive computing*), "informatique omniprésente" (*ubiquitous computing*) ou "objets communicants" (*communicating devices*).

Treize ans plus tard, les premiers pas ont été franchis. En effet, les téléphones cellulaires sont omniprésents et offrent de nombreux services tels que l'accès permanent à l'Internet, la prise de photos, la localisation géographique, la découverte et l'utilisation d'autres machines se trouvant à proximité (par exemple un distributeur de boissons ou une imprimante supportant Bluetooth). De plus, les véhicules ainsi que les appareils ménagers commencent à être interconnectés et offrent des services de plus en plus sophistiqués. Finalement, de nombreux petits objets commencent à inclure des étiquettes intelligentes (*RFID tags*) qui remplacent progressivement les codes barres et sont sur le point de révolutionner le monde de la distribution. Ce dernier exemple montre clairement que l'informatique diffuse relie directement le monde physique des objets (étiquetés) et leurs représentations virtuelles dans des bases de données.

Ce nouveau paradigme a un impact important sur la sécurité des systèmes informatiques. En effet, en plus des problèmes classiques liés à la sécurité des communications sans fil, de nouveaux défis apparaissent. Les interactions avec les utilisateurs doivent être spontanées et transparentes. En conséquence, il n'est pas envisageable d'exiger un mot de passe avant chaque interaction. De plus, les informations sur le contexte d'un utilisateur ou d'un objet deviennent importantes. Par exemple, le contrôle d'accès peut être basé sur la localisation d'un utilisateur. Une autre limitation est le manque d'infrastructure de communication qui interdit de baser toute la sécurité sur des tiers de confiance distants. Le nombre d'acteurs potentiels dans ce type de système étant très grand, de nombreuses interactions ont lieu entre des entités qui ne se connaissent pas. L'infrastructure de confiance étant insuffisante voire inexistante, de nouveaux mécanismes pour établir des relations de confiance sont nécessaires. Finalement, la protection de la vie privée (*privacy*) est un problème majeur dans ces environnements. En effet, la vie privée des utilisateurs est mise en péril par le nombre croissant d'interactions pouvant potentiellement être enregistrées et corrélées.

L'informatique diffuse pose encore de nombreux défis et une importante communauté de chercheurs est en train d'émerger autour de ce thème pluridisciplinaire. Au moins trois conférences internationales traitent de ce sujet : Ubicomp depuis 1999, Pervasive depuis 2002 et PerCom depuis 2003. De plus, deux journaux couvrent ce domaine : *personal and ubiquitous computing* est édité conjointement par l'ACM et Springer depuis 1997 et *IEEE pervasive computing (mobile and ubiquitous systems)* a été démarré en 2002. L'informatique diffuse entraînant de nouveaux problèmes en sécurité, les workshops sur ce sujet foisonnent dans les conférences susmentionnées et dans les conférences sur la sécurité. En 2003, la première conférence dédiée à la sécurité de l'informatique diffuse (*international conference on security in pervasive computing*) a été organisée. Finalement, un livre traitant de ce sujet a été publié en 2002 [Sta02].

Structure de cette thèse

Ce mémoire de thèse décrit principalement un ensemble de protocoles de sécurité permettant à une entité de prouver qu'elle a pris part à des interactions passées tout en évitant d'être tracée (*untraceability*). Ces preuves sont utilisées dans le but d'établir une relation de confiance avec une autre entité. Ainsi, une entité peut prouver qu'elle fait partie d'un groupe, qu'elle est recommandée par un tiers de confiance ou qu'elle se trouvait en un lieu à un moment donné. Au niveau applicatif, ces informations sont utilisées pour décider si une entité inconnue peut accéder à un service.

La première partie de cette thèse présente deux protocoles dédiés aux environnements sans infrastructure de confiance, c'est-à-dire sans certification des utilisateurs. Le chapitre 1 présente des jetons (*one-time credentials*) permettant d'accéder à un service une seule fois et révélant un chèque électronique en cas de comportement malveillant, c'est-à-dire en cas d'usage multiple. Le chapitre 2 de ce mémoire présente un autre type de jeton permettant à un utilisateur de prouver le résultat d'interactions passées sans que cette preuve puisse être liée à cet utilisateur ou à une interaction.

La deuxième partie de ce mémoire est le cœur de cette thèse. Il présente l'architecture développée pour créer un historique et pour révéler des éléments de cet historique à un tiers. Le chapitre 3 présente un mécanisme de signature anonyme lié à un historique (*unlinkable credential*). Ce nouveau type de signature est une extension des signatures de groupe et permet de signer en tant que "quelqu'un qui est recommandé par Bob", "une personne qui se trouvait à Paris en janvier", ou "un visiteur du musée d'art moderne de Nice" sans révéler d'information sur l'identité du signataire ni permettre de lier deux signatures. Le chapitre 4 présente un mécanisme permettant de prouver la proximité d'une entité connaissant un secret (*distance-bounding proofs of knowledge*). Le chapitre 5 combine ces deux techniques et montre comment un historique d'interactions peut être défini et utilisé lors de l'établissement d'une relation de confiance.

La dernière partie de ce mémoire se concentre sur l'implémentation d'un sous-ensemble des concepts présentés précédemment. Ce travail est notre contribution au projet WiTness. Une bibliothèque Java permettant la création et la vérification de certificats d'attribut génériques en XML ainsi que deux prototypes utilisant cette bibliothèque ont été implémentés et permettent l'établissement de confiance au sein d'une fédération d'assistants numériques (*PDA*) et d'ordinateurs portables reliés par des connexions sans fil Bluetooth. Le premier prototype permet la distribution de contenu en fonction de la classification des données, des droits des utilisateurs et du niveau de confiance des différentes machines utilisées. Le second prototype permet à un groupe de personnes en réunion d'échanger des données de façon sécurisée, de signer un document ou de voter.

1 Motivation : quatre nouvelles contraintes

Il est important de déterminer les besoins en sécurité qui sont spécifiques à l'informatique diffuse. Pour commencer, certains besoins sont hors du sujet de cette thèse : la sécurité des communications sans fil n'est pas traitée ; la fiabilité et la sécurité au niveau réseau des systèmes auto-organisés (réseaux ad hoc) n'est pas prise en compte ; finalement la sécurité des réseaux de capteurs, où les limitations en termes de puissance de calcul et de moyens de communication sont très fortes, n'est pas couverte.

Les services de sécurité nécessaires à l'informatique diffuse sont classiques : contrôle d'accès, authentification, non-répudiation, confidentialité, intégrité, etc. Cependant, de nouvelles contraintes imposent la redéfinition de ces services et la création de nouveaux mécanismes. Nous nous concentrons sur les quatre contraintes principales de l'informatique diffuse : le manque d'infrastructure de confiance, le manque d'infrastructure de communication, le besoin de protéger la vie privée des utilisateurs et le besoin de prendre en compte le contexte. Chacune de ces contraintes va être détaillée dans les paragraphes suivants.

1.1 Manque de relations de confiance

La première contrainte est le manque de relations de confiance. Quand une entité A interagit avec une entité B, A doit pouvoir évaluer le niveau de confiance de B pour décider si B peut être autorisé à accéder à des services offerts par A ou pour déterminer si les services offerts par B sont fiables.

Différentes techniques existent pour déterminer le niveau de confiance d'une autre entité (voir Figure 1). Une méthode classique pour évaluer la confiance est d'authentifier les entités et d'utiliser une liste (par exemple une liste de contrôle d'accès) pour lier chaque identité à une notion de confiance. Dans l'informatique diffuse, le nombre d'objets communicants est potentiellement immense et les interactions avec des inconnus peuvent donc être fréquentes. Dans ce cas, la notion d'identité est inutile car il n'est pas possible de dériver une notion de confiance à partir d'un nom sans le connaître a priori. En utilisant des certificats d'attribut ou d'autorisation (X.509 ou SPKI), il est possible d'obtenir des informations certifiées par une autorité de confiance. Malheureusement, l'informatique diffuse ne permet généralement pas de trouver un lien hiérarchique entre deux entités. Finalement les approches basées sur l'observation des autres entités semblent appropriées. Les recommandations permettent la distribution de ses propres observations à d'autres parties et les systèmes de réputation se basent sur une mesure statistique des observations de nombreuses entités.

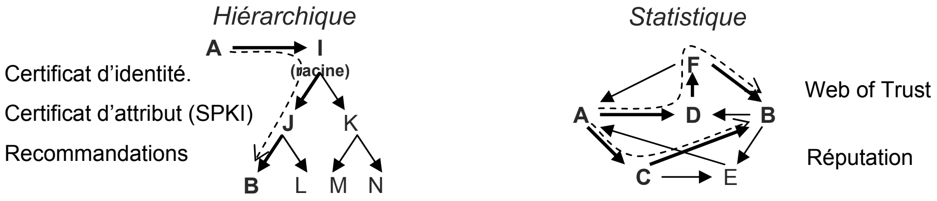


FIG. 1 – Différentes approches pour définir une relation de confiance

Nous proposons une extension des systèmes de recommandation en définissant la notion d'historique qui permet à chaque entité de stocker l'ensemble des interactions passées qui peuvent être prouvées. Ces données peuvent être directement liées à une notion de confiance ou pas : un historique peut ainsi contenir une preuve de localisation, une recommandation ou une carte d'identité numérique. Chacun de ces éléments peut être prouvé lors de l'établissement d'une relation de confiance avec un inconnu.

1.2 Manque d'infrastructure de communication

Une autre contrainte importante est le manque d'infrastructure de communication. Lorsque deux ou plusieurs objets communiquent localement au sein d'une fédération (*per-*

sonal area network), il n'est pas toujours possible d'avoir accès à un tiers de confiance (*trusted third party*) distant que ce soit pour des raisons de coût, de temps de réponse ou de manque d'infrastructure (voir figure 2).

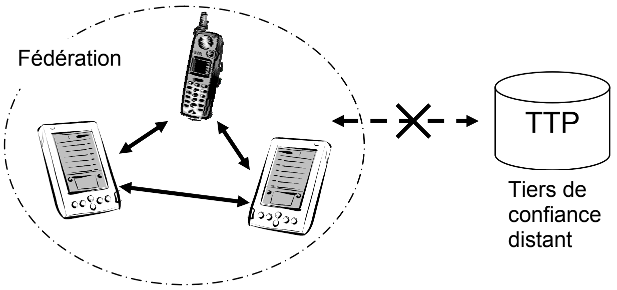


Fig. 2 – Interactions déconnectées : impossibilité de joindre un tiers de confiance distant

Pour permettre à une entité de prouver son historique sans interaction avec un tiers distant, nous proposons que chaque entité maintienne et transporte son propre historique sous la forme d'une base de donnée locale contenant des certificats pouvant être vérifiés sans nécessiter un tiers de confiance.

1.3 Besoin de protéger la vie privée des utilisateurs

Une autre contrainte de l'informatique diffuse est liée à son fort pouvoir d'observation des utilisateurs. Aujourd'hui les habitudes d'achat des utilisateurs sont observées grâce aux cartes de paiement et leur localisation est possible grâce aux téléphones cellulaires. Sans prendre de précaution pour protéger la vie privée des utilisateurs de l'informatique diffuse, toutes leurs interactions, de l'ouverture d'une porte à la rencontre d'une autre personne, pourront être tracées.

Nous proposons donc un historique qui puisse être prouvé tout en choisissant de ne révéler que les informations pertinentes. Un mécanisme de certificat non traçable est proposé dans ce but.

1.4 Besoin de prendre en compte le contexte

Finalement, la quatrième contrainte est la prise en compte du contexte. L'informatique diffuse propose d'associer un microprocesseur avec une puissance de calcul et des moyens de communication à tous les objets qui nous entourent. Le résultat est un lien fort entre une identité virtuelle (par exemple une clé publique) et un objet physique (dans lequel est encapsulé le microprocesseur qui connaît la clé privée correspondante). Ce lien doit pouvoir être prouvé dans de nombreux cas :

- Il peut être nécessaire de prouver le contexte physique dans lequel se trouve un objet, par exemple la localisation de cet objet à un instant donné. Ces informations contextuelles font partie de l'historique.
- Un autre besoin est de lier un objet physique et une donnée (voir figure 3). Par exemple, lier une montre et un certificat signé par le fabriquant de cette montre.
- Finalement associer deux objets (*device pairing*) est souvent nécessaire à l'établissement d'un canal sécurisé.

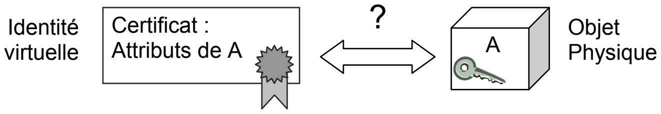


Fig. 3 – Relations entre un objet physique et son identité virtuelle

Pour lier les données aux objets, nous proposons un mécanisme de preuves de proximité qui permet de prouver qu'un secret (une clé privée) est connu localement.

1.5 Notre approche

Pour répondre aux trois premières contraintes (manque d'infrastructure de confiance, interactions déconnectées et protection de la vie privée), un mécanisme permettant de prouver un historique en étant déconnecté et en restant anonyme est nécessaire. Dans ce but, nous proposons un mécanisme de certificats non traçables. Pour traiter la quatrième contrainte (prise en compte du contexte physique), nous proposons un autre mécanisme : les preuves de proximité. La suite de ce résumé décrit en détail ces deux mécanismes.

2 Certificats non-traçables

Le premier mécanisme est le certificat non traçable. Son principe est décrit dans la figure 4. Alice (A) possède différents certificats dans son historique :

- Une preuve de localisation (géodatage) prouve qu'elle était à Sophia-Antipolis le 15 octobre 2004. Ce certificat lui a été délivré par une borne interactive.
- Un certificat d'attribut prouve qu'elle est professeur à l'ENST. Ce certificat est renouvelé chaque année par son employeur.
- Une recommandation prouve que Bob lui fait confiance. Ce certificat lui a été remis après une interaction avec Bob et a une durée de vie limitée.

Lorsque Alice prouve son historique, elle choisit les certificats qu'elle veut révéler et la granularité des informations présentées. Par exemple, avec un codage approprié, elle peut choisir de cacher les détails concernant sa localisation et prouver qu'elle était en France le 15 octobre. Cette preuve peut se faire interactivement lors d'un échange face à face ou sans interaction dans le cas d'une signature.

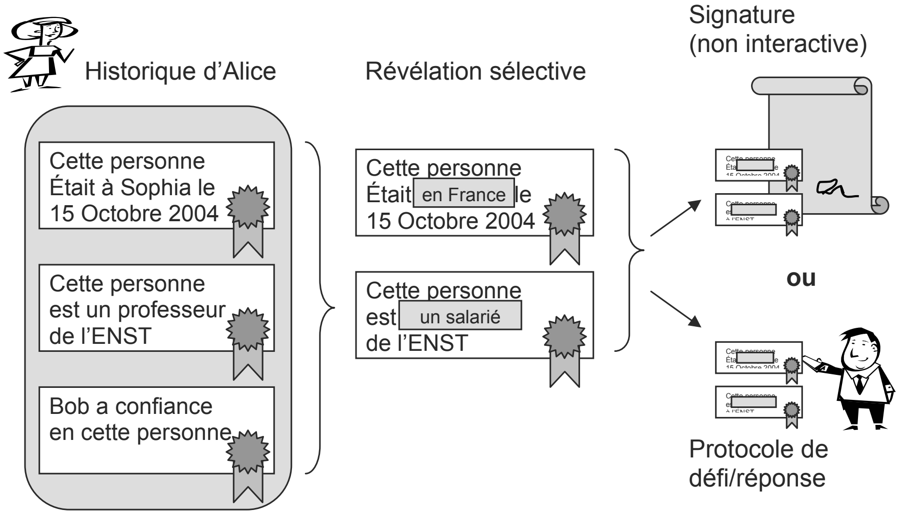

Historique d'Alice
Révélation sélective
Signature
(non interactive)
Cette personne Était à Sophia le 15 Octobre 2004
Cette personne est un professeur de l'ENST
Bob a confiance en cette personne
Cette personne Était en France le 15 Octobre 2004
Cette personne est un salarié de l'ENST
ou
Protocole de
défi/réponse

Fig. 4 – Certificats non-traçables utilisés de manière interactive ou non.

Ce mécanisme permet de signer un document en tant que "un salarié de l'ENST qui était en France le 15 octobre 2004" ou "un journaliste qui était sur les lieux des faits". La signature ne peut être générée que par une entité ayant l'historique adéquat, c'est-à-dire, ayant reçu les certificats nécessaires. La signature peut être vérifiée de manière déconnectée à condition de connaître les clés publiques des entités ayant fourni les certificats. La signature ne révèle aucune information sur l'identité du signataire et il n'est pas possible de savoir si deux signatures ont été générées par la même personne.

2.1 Solutions existantes

Pour implémenter les certificats non traçables, différentes technologies peuvent être envisagées :

- *Les certificats d'attribut classiques* : ils ne sont pas adaptés car la présentation ne peut pas être sélective et ils sont traçables car la clé publique du possesseur est visible.
- *Les jetons non traçables* : par exemple l'argent électronique ou les certificats proposés par Brands [Bra02] ne sont pas traçables mais ne sont utilisables qu'une seule fois. Il est évident qu'un historique doit pouvoir être conservé et réutilisé.
- *les pseudonymes* : l'approche proposée dans Idemix [CL01] correspond mieux à nos besoins. Malheureusement les pseudonymes ne peuvent pas être utilisés pour des

schémas de signature classique c'est-à-dire vérifiable par des entités qui ne sont pas connues à l'avance.

Nous proposons donc une nouvelle approche qui est une généralisation des signatures de groupe. Dans notre cas la signature ne se fait pas en tant qu'un membre anonyme d'un groupe mais en tant qu'une entité anonyme avec un historique donné.

2.2 Notre solution : extension des signatures de groupe

Une version simplifiée du protocole est donnée ici. Pour plus de détails, le lecteur se réfèrera au chapitre 3.

Chaque autorité délivrant des certificats non traçables a un modulo RSA n tel que $n = p \cdot q$ où p et q sont deux grands nombres premiers. Un ensemble de petit nombres premiers $e_1, \ldots, e_m$ est choisi tel que pour tout $i \in \{1, \ldots, m\}$, $\gcd(e_i, \phi(n)) = 1$. Chaque e_i correspond à un attribut et sa signification est publique. Chaque autorité calcule l'ensemble $\{d_1, \ldots, d_m\}$ tel que pour tout $i \in \{1, \ldots, m\}$, $e_i \cdot d_i = 1 \mod \phi(n)$. $\mathcal{Z}_n = \{0, 1, 2, \ldots, n-1\}$ est l'ensemble des entiers relatifs modulo n, $\mathcal{Z}_n^* = \{i \in \mathcal{Z}_n \mid \gcd(i, n) = 1\}$ est un groupe multiplicatif et $G = \{1, g, g^2, \ldots, g^{n-1}\}$ est un groupe cyclique d'ordre n dont g est un générateur. La clé publique d'une autorité est $(n, e_1, \ldots, e_m, G, g, a)$ où $a \in \mathcal{Z}_n^*$. La clé privée de cette autorité est $(p, q, d_1, \ldots, d_m)$.

Alice possède un secret x qu'elle ne veut pas (ou ne peut pas) révéler. Un certificat délivré par B à A a la forme suivante : $(a^x + 1)^D \mod n$ où D définit les attributs de A. Ce certificat peut être obtenu par A sans révéler son secret x.

Révélation sélective

Une extension des signatures RSA est proposée pour signer un message avec un attribut. La signature du message m par le signataire B avec les attributs définis par l'ensemble S est : $SIGN_{(B,S)}(m) = m^D \mod n$ où $D = \prod_{i \in S} d_i$. Cette signature peut être transformée en une signature du message m par B avec les attributs définis par le sous-ensemble S' à condition que $S' \subseteq S$. En effet :

$$\begin{aligned} SIGN_{(B,S')}(m) &= \left(SIGN_{(B,S)}(m)\right)^{\left(\prod_{j \in \{S \setminus S'\}} e_j\right)} \\ &= m^{\left(\prod_{i \in S} d_i \cdot \prod_{j \in \{S \setminus S'\}} e_j\right)} = m^{\left(\prod_{i \in S'} d_i\right)} = m^{D'} \mod n \end{aligned}$$

En d'autres termes, A peut transformer un certificat $(a^x + 1)^D$ stocké dans son historique en $(a^x + 1)^{D'}$ avant de prouver son historique.

Il est nécessaire de définir un codage des attributs qui donne du sens aux transformations rendues possibles par le mécanisme de signature. Par exemple, un attribut ayant la valeur décimale âge= 31_d est représenté en binaire par 011111_b. Si le code consiste à lister les bits égaux à 1, $S = \{4, 3, 2, 1, 0\}$, les seules transformations possibles consistent à enlever des éléments de S et donc de changer certains bits égaux à 1 en 0. Ainsi, il est uniquement possible de réduire la valeur de l'attribut. Par exemple, $D = d_4d_3d_2d_1d_0$ peut être transformé en $D' = d_4d_1$ soit âge'= 18_d en utilisant e_0, e_2 et e_3 qui sont publiques. Ainsi, ayant reçu un certificat indiquant qu'elle a trente et un ans, Alice peut choisir de prouver qu'elle est majeure (âge=31 ans ⇒ âge≥18 ans). Des codages plus subtils peuvent être proposés. Par exemple, Alice peut recevoir le certificat suivant :

> `[14|04|32, 15|10|2004, 43|62|65, 007|04|70]`
> prouvant qu'elle se trouvait dans les bâtiments de l'institut Eurécom à deux heures de l'après midi le 15 octobre 2004. Ce certificat est stocké dans l'historique d'Alice.

En signant ou lors d'une preuve interactive, elle peut choisir de prouver qu'elle possède le certificat suivant :

> `[14|XX|XX, XX|XX|XXXX, 43|62|65, 007|04|70]`
> Elle est *une personne qui se trouvait à l'institut Eurécom un après-midi.*

Ou, elle peut choisir de révéler :

> `[XX|XX|XX, 15|10|2004, 43|XX|XX, 007|XX|XX]`
> Elle est *une personne qui était dans le sud de la France le 15 octobre 2004.*

Ici le codage utilisé contient un "checksum" par bloc qui permet de le montrer ou de le cacher mais n'autorise pas sa modification. Le codage de la localisation peut être plus structuré en utilisant une hiérarchie du type pays, ville, quartier, bâtiment, salle.

Preuve de la connaissance d'un certificat

Le second besoin concernant les certificats est d'éviter leur traçabilité tout en assurant qu'ils ne puissent pas être transférés d'une personne à une autre. Pour prouver la possession d'un certificat sans le montrer, nous utilisons les "preuves de connaissance" (*proof of knowledge*) et les "signatures basées sur des preuves de connaissance" (*signature based on a proof of knowledge*).

Une preuve de connaissance (PK) est un protocole entre un vérifié (P) et un vérificateur (V). A la fin de ce protocole, P a prouvé à V qu'il connaît un secret x sans avoir révélé d'information sur ce secret. Par exemple, $\mathrm{PK}[\alpha : y = g^\alpha]$ est la preuve de la connaissance du logarithme discret de y en base g.

Une signature basée sur une preuve de connaissance est une version non interactive des preuves de connaissances. Par exemple $sig = \mathrm{SPK}[\alpha : y = g^{\alpha}](m)$ est une signature basée sur la preuve de la connaissance du logarithme discret de y en base g. Le message m est signé par une entité ayant cette connaissance.

Pour prouver la connaissance d'un certificat, nous utilisons une extension des signatures de groupe proposées dans [CS97]. Une signature basée sur la connaissance d'un double logarithme discret est combinée avec une signature basée sur la connaissance de la racine $E'^{\text{ième}}$ d'un logarithme discret :

$$\begin{aligned} sig_1 &= \mathrm{SPK}[\alpha \mid \tilde{z} = \tilde{g}^{(a^{\alpha})}](m) \\ sig_2 &= \mathrm{SPK}[\beta \mid \tilde{z}\tilde{g} = \tilde{g}^{(\beta^{E'})}](m) \end{aligned}$$

Dans les signatures ci-dessus, $\tilde{g} = g^r$ et $\tilde{z} = z^r$, où $r \in_R \mathcal{Z}_n$ est choisi aléatoirement avant chaque preuve et où $z = g^{(a^x)}$. La signature sig_1 prouve la connaissance d'un secret α. La signature sig_2 prouve la connaissance de β. En combinant les deux signatures, il est possible de montrer que β est un certificat sur α avec l'attribut D' : $\tilde{g}^{(\beta^{E'})} = \tilde{z}\tilde{g} = \tilde{g}^{(a^{\alpha}+1)}$ et donc $\beta = (a^{\alpha}+1)^{D'} \mod n$. Le vérificateur est convaincu que le signataire connaît un secret et que ce secret est certifié avec l'attribut défini par D'. Cependant le secret x et le certificat $(a^x+1)^{D'} \mod n$ ne sont pas montrés. Il est par conséquence impossible de lier plusieurs utilisations d'un même certificat.

3 Preuves de proximité

Le deuxième mécanisme nécessaire à la sécurité de l'informatique diffuse est la preuve de proximité. Il est parfois nécessaire de prouver sa localisation, de vérifier qu'un certificat est associé à un objet ou d'associer deux objets. Dans le but de résoudre l'ensemble de ces problématiques, nous proposons un mécanisme permettant de prouver qu'un secret tel qu'une clé privée est connu localement. Nous nommons ce nouveau mécanisme "preuves de connaissance et de proximité" ou *distance bounding proof of knowledge* (DBPK).

3.1 Nouvelles attaques et solutions existantes

Les mécanismes utilisés pour prouver la connaissance d'une clé privée ou d'un autre secret sont généralement basés sur le principe des protocoles de défi/réponse (*challenge-response*) : un défi (c) est envoyé par le vérificateur au vérifié qui retourne une réponse (s) dépendant du défi et du secret. Ces protocoles permettent de vérifier qu'un canal de communication est établi avec une entité logique (par exemple le serveur d'une banque). Cependant, dans le cadre de l'informatique diffuse, il est non seulement nécessaire de vérifier qu'une entité est impliquée mais il faut aussi vérifier qu'elle est physiquement présente.

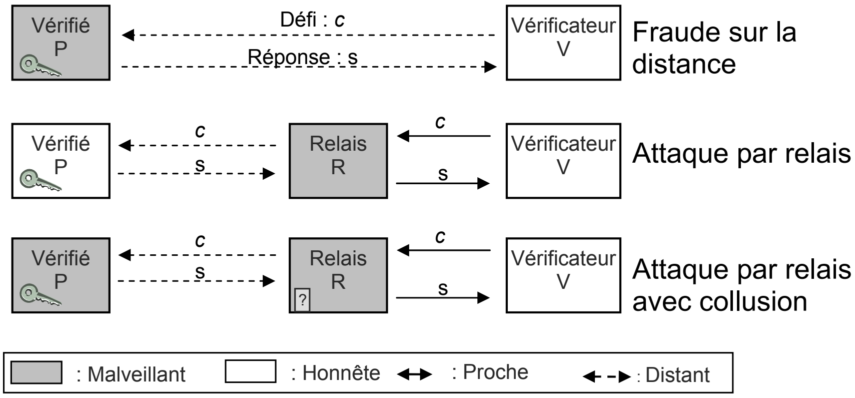


Fig. 5 – Trois nouvelles attaques.

Trois nouveaux types d'attaques peuvent être montés contre un protocole de défi/réponse utilisé en informatique diffuse (voir figure 5) :

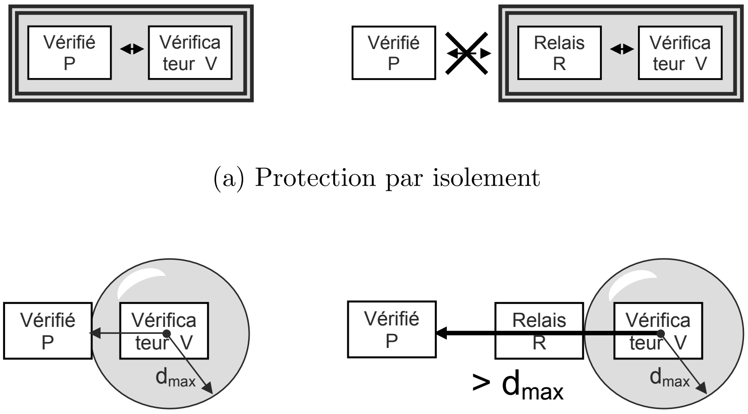
- *Fraudes sur la distance* : un vérifié prétend être proche d'un vérificateur alors qu'il est distant. Ce type d'attaque peut être déjoué à condition de prendre en compte des contraintes physiques limitant la propagation du défi. Par exemple, le défi peut être restreint à une salle de réunion en utilisant une émission infrarouge [BSSW02] ou une onde sonore [SSW03].
- *Attaques par relais* : un relais malveillant est physiquement présent en face du vérificateur. Il relaie les défis et les réponses vers un vérifié en utilisant un autre moyen de communication. Contrairement à une attaque du type "man in the middle", le relais n'agit pas au niveau du protocole cryptographique.
- *Attaques par relais avec collusion* : ce troisième type d'attaque implique un vérifié malveillant, distant, collaborant avec un relais physiquement présent en face du vérificateur. C'est une combinaison des deux attaques précédentes.

Nous nous intéressons aux attaques du troisième type qui englobent les deux autres cas. Pour éviter les attaques par relais, il est nécessaire de prouver qu'un secret est connu localement. Deux approches existent.

La première solution est l'isolement (voir figure 6-a) : le vérifié et le vérificateur sont mis en relation sans moyen de communiquer avec le reste du monde pendant l'exécution d'un protocole de défi/réponse (par exemple en utilisant une cage de Faraday). L'entité vérifiée n'ayant aucun moyen de communication avec l'extérieur, le vérificateur a la preuve que le vérifié est présent.

La deuxième approche est d'utiliser un mécanisme permettant de mesurer la distance entre le vérificateur et le détenteur du secret (voir figure 6-b). Si la distance du vérifié augmente, par exemple en intercalant un relais, la vérification n'est plus valide.

Dans l'annexe A, nous proposons une autre approche basée sur la cryptographie quantique qui empêche un relais de relayer les défis et les réponses en utilisant un autre média de communication.



(a) Protection par isolement

(b) Protection par mesure de distance

FIG. 6 – Deux principaux types de protection contre les fraudes par relais.

Dans ce travail, nous avons choisi la deuxième approche qui est beaucoup plus souple pour répondre aux besoins de l'informatique diffuse et qui peut être implémentée simplement.

3.2 Notre solution : preuves de connaissance et de proximité

La mesure de distance peut être directement liée au temps d'aller-retour d'une information à condition de minimiser les temps de calcul et le protocole de communication. Dans ce cas il peut être possible de détecter l'effet d'un relais qui va forcément allonger le temps de réponse en augmentant le chemin ou en traitant les messages.

Nous proposons d'utiliser un protocole de défi/réponse minimal : un bit de défi, une opération logique pour calculer la réponse et un bit de réponse. L'avantage de ce protocole est que chaque exécution peut se faire en quelques nanosecondes et permet ainsi une mesure précise de la distance. La difficulté consiste à avoir des garanties cryptographiques tout en respectant ces contraintes fortes.

Brands et Chaum ont proposé une solution pour éviter les attaques par relais en partant des mêmes contraintes [BC93]. Leur protocole (voir table 1) est partagé en deux phases : premièrement, une série de défis et de réponses de un bit sont échangés rapidement et le temps d'aller-retour est mesuré ; deuxièmement, le vérifié signe les bits échangés. La première partie permet de vérifier qu'une entité recevant a et connaissant b est proche. La seconde partie permet de vérifier que P a bien reçu les bits de a et retourné les bits de b. Cette approche fonctionne tant que P se comporte correctement. En d'autres termes, ce protocole ne permet pas d'éviter les attaques par relais avec collusion.

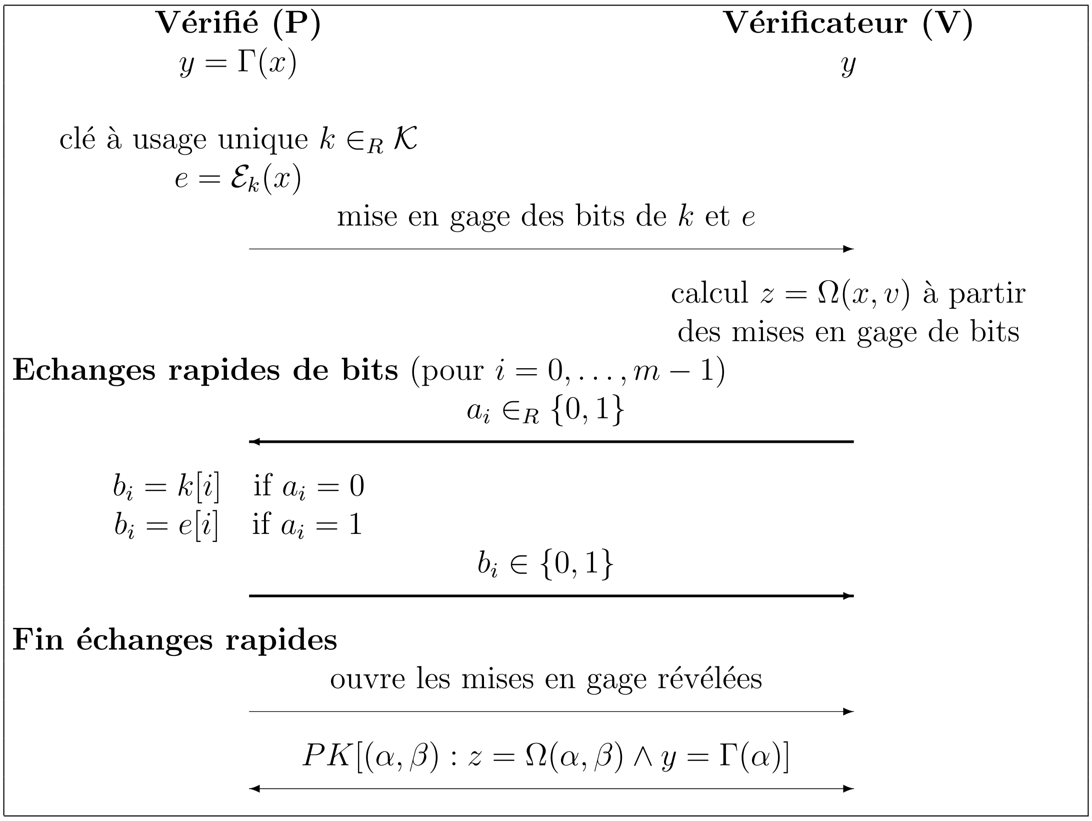
vérifié (P)		**vérificateur (V)**
K_{P_P}, K_{S_P}		K_{P_P}
Génère $b \in_R \{0,1\}^m$		Génère $a \in_R \{0,1\}^m$
Echanges rapides de bits (pour $i = 0, \ldots, m-1$)		
		commence mesure du RTT
	$a[i]$ ←	
	$b[i]$ →	
		arrête mesure du RTT
Fin de l'échange rapide		
		vérifie les RTTs
	$SIGN_P(a,b)$ →	
		vérifie la signature

Tab. 1 – Principe de base du protocole proposé par Brands et Chaum

Pour éviter les attaques par relais avec collusion nous modifions le schéma pour que les bits de réponse dépendent de la clé privée du vérifié. Notre protocole est décrit dans la table 2. Les bits de réponse sont liés à la clé privée x du vérifié : si le $i^{\text{ème}}$ défi $a[i]$ est un zéro, alors la réponse est le $i^{\text{ème}}$ bit d'une clé à usage unique k ; sinon, la réponse est le $i^{\text{ème}}$ bit du chiffrement e de la clé privée x en utilisant la clé à usage unique k. Le schéma évite la présence d'un relais car l'échange rapide nécessite la connaissance de k et de e et donc de x. Cependant, il faut que le schéma assure que le vérificateur qui obtient la moitié des bits de (k, e) ne puisse pas en déduire d'information sur x tout en étant capable de vérifier que e est réellement le chiffrement de x avec la clé k.

Le mécanisme de mise en gage des bits de k et e (*bit commitment*) est choisi de telle façon qu'il est possible d'en déduire une représentation de z. En d'autres termes z est liée au déchiffrement de e avec k, c'est-à-dire x. L'échange rapide permet de vérifier qu'une entité proche connaît un k et un e. L'ouverture des mises en gage correspondant aux bits révélés pendant l'échange rapide lie cet échange rapide au chiffrement de la clé privé : une entité proche connaît un secret lié à z. Finalement, la preuve de connaissance prouve que ce secret est bien la clé privée correspondant à y.

Dans le chapitre 4, nous proposons une implémentation de ce concept basée sur le logarithme discret $y = g^x \mod p$. Pour le chiffrement de la clé privée x, nous utilisons



Vérifié (P) $y = \Gamma(x)$

Vérificateur (V) y

clé à usage unique $k \in_R \mathcal{K}$

$e = \mathcal{E}_k(x)$

$$\xrightarrow{\text{mise en gage des bits de } k \text{ et } e}$$

calcul $z = \Omega(x, v)$ à partir des mises en gage de bits

Echanges rapides de bits (pour $i = 0, \ldots, m-1$)

$$\xleftarrow{a_i \in_R \{0,1\}}$$

$b_i = k[i]$ if $a_i = 0$

$b_i = e[i]$ if $a_i = 1$

$$\xrightarrow{b_i \in \{0,1\}}$$

Fin échanges rapides

$$\xrightarrow{\text{ouvre les mises en gage révélées}}$$

$$\xleftrightarrow{PK[(\alpha, \beta) : z = \Omega(\alpha, \beta) \wedge y = \Gamma(\alpha)]}$$

Tab. 2 – Vue générale des preuves de connaissance et de proximité

le schéma suivant : $e = ux - k \mod p-1$ où u est choisi aléatoirement et est public ($u \in_R \{1, \ldots, p-2\}$) et où la clé à usage unique k est choisie aléatoirement $k \in_R \mathcal{Z}_{p-1}$.

Les mises en gage de bit de la clé k sont définies comme suit : $c_{(k,i)} = g^{k[i]} \cdot h^{v_{k,i}} \mod p$ et les mises en gage des bit de chiffrement e, sont $c_{(e,i)} = g^{k[i]} \cdot h^{v_{e,i}} \mod p$. A partir là, une représentation de z est obtenue :

$$\begin{aligned}
z &= \prod_{i=0}^{m-1} \left(c_{k,i} \cdot c_{e,i}\right)^{2^i} = \prod_{i=0}^{m-1} \left(g^{k[i]} h^{v_{k,i}} \cdot g^{e[i]} h^{v_{e,i}}\right)^{2^i} \\
&= \prod_{i=0}^{m-1} \left(g^{k[i]+e[i]}\right)^{2^i} \cdot \prod_{i=0}^{m-1} \left(h^{v_{k,i}+v_{e,i}}\right)^{2^i} \\
&= \prod_{i=0}^{m-1} \left(g^{2^i k[i] + 2^i e[i]}\right) \cdot \prod_{i=0}^{m-1} \left(h^{2^i v_{k,i} + 2^i v_{e,i}}\right) \\
&= g^{\sum_{i=0}^{m-1}\left(2^i \cdot k[i] + 2^i \cdot e[i]\right)} \cdot h^{\sum_{i=0}^{m-1}\left(2^i \cdot (v_{k,i} + v_{e,i})\right)} = g^{k+e} \cdot h^v = g^{u \cdot x} \cdot h^v \mod p
\end{aligned}$$

Finalement, une preuve de connaissance lie cette représentation de z à la clé publique y : $PK[(\alpha, \beta) : z = g^{u\alpha} h^{\beta} \wedge y = g^{\alpha}]$

4 Historique : prouver sans être tracé

Dans les sections précédentes, nous avons proposé deux nouveaux mécanismes : les certificats non traçables et les preuves de proximité. Pour combiner ces deux mécanismes, il est important de noter que les preuves de connaissance (PK), les signatures basées sur des preuves de connaissance (SPK) et les preuves de connaissance et de proximité (DBPK) sont interchangeables.

Il est donc possible de remplacer la PK utilisée lors de l'obtention d'un certificat par une DBPK. Ainsi, Alice peut prouver sa proximité lorsqu'elle demande une preuve de localisation (géodatage). Dans un autre scénario, Alice pourrait prouver qu'elle fréquente régulièrement un magasin et obtenir ainsi un rabais.

Lors de l'utilisation d'un certificat, il est aussi possible de remplacer l'une des SPK par une DBPK. Alice peut donc prouver qu'une personne anonyme avec un certain historique (par exemple, salarié de l'ENST) est présente. Ce mécanisme peut aussi assurer l'authentification d'objets communicants.

Quand les certificats non traçables sont combinés avec les preuves de connaissance et de proximité, les quatre contraintes décrites dans la section 1 sont satisfaites : les relations de confiance peuvent être établies à partir d'un historique, cet historique peut être prouvé sans interaction avec un tiers de confiance et n'est pas traçable. Enfin, l'historique peut contenir des informations contextuelles. En effet, il est non seulement possible de stocker des recommandations, des certificats d'attributs et des relations hiérarchiques mais aussi des preuves d'interaction et des preuves de localisation. Un document peut donc être signé par "un journaliste qui était sur les lieux des faits" où la notion de journaliste est un certificat d'attribut (une carte de presse numérique) et les lieux des faits sont associés à une preuve de localisation.

4.1 Protection de la vie privée

Nous avons proposé une solution pour définir des certificats non traçables et pour permettre la révélation sélective des attributs. Cependant, pour avoir une protection globale de la vie privée, il est nécessaire de tenir compte de la non traçabilité au niveau réseau et au niveau applicatif (voir figure 7). Au niveau réseau, il est indispensable d'avoir des adresses MAC changeantes ou d'utiliser un réseau de "mixes". Au niveau applicatif, il est nécessaire de mettre en place un système permettant de contrôler les informations révélées.

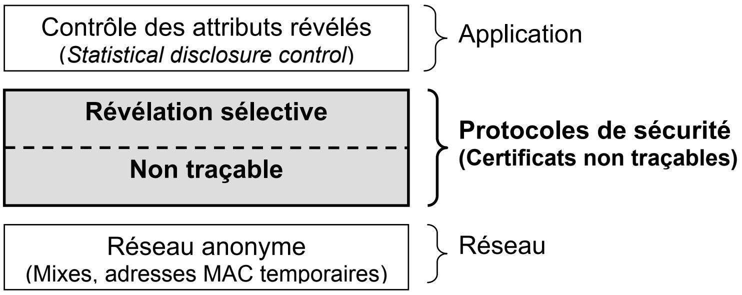

Contrôle des attributs révélés
(*Statistical disclosure control*)
Application
Révélation sélective
Non traçable
Protocoles de sécurité
(Certificats non traçables)
Réseau anonyme
(Mixes, adresses MAC temporaires)
Réseau

FIG. 7 – La non traçabilité est nécessaire sur trois plans

4.2 Implémentation

Dans le cadre du projet de recherche européen WiTness, nous avons pu implémenter une première version de notre notion d'historique. Malheureusement, pour des raisons techniques, la protection de la vie privée n'a pas pu être abordée.

Nous avons donc développé une approche pragmatique pour établir une relation de confiance au sein d'une fédération d'objets communicants. Une bibliothèque Java permettant la création, la délégation et la vérification de certificats XML a été proposée. Le mécanisme de découverte de Bluetooth a été utilisé pour vérifier la proximité des entités. Deux prototypes ont été implémentés pour des applications de type "entreprise-employé" (B2E). La carte SIM étant omniprésente, elle a été choisie comme module sécurisé pour protéger la clé privée de chaque employé.

Ce projet nous a permis d'utiliser de nouvelles technologies : les différentes versions de Java pour les environnements mobiles (J2ME, Personal Java), les réseaux personnels (Bluetooth, 802.11), les cartes SIM et les environnements de développement sur assistants numériques (iPaq). Beaucoup d'énergie a été consacrée à la maîtrise des interfaces de programmation qui sont en constante évolution (JSR-82, JSR-177).

Conclusions et perspectives

Nous avons défini ce concept d'historique en tant qu'extension des systèmes de recommandation pour l'établissement de relations de confiance. Un historique permet de gérer un ensemble d'assertions pouvant être prouvées. Nous avons ajouté deux contraintes : les preuves doivent pouvoir se faire sans connexion avec un tiers de confiance et elles ne doivent pas menacer la vie privée des utilisateurs.

Nos contributions principales sont la définition de deux types de mécanismes : les certificats non traçables et les preuves de proximité. Premièrement, nous avons proposé trois mécanismes de certificats non traçables (chapitres 1, 2 et 3) en partant d'hypothèses différentes concernant l'infrastructure de confiance. Deuxièmement, nous avons défini le mécanisme de preuve de connaissance et de proximité (chapitre 4) qui est la première parade aux attaques par relais avec collusion pour l'informatique diffuse. En effet, seule l'isolement permet une sécurité équivalente mais cette approche n'est pas suffisamment flexible pour être employée dans l'informatique diffuse.

Dans le futur, nous envisageons de fusionner les trois types de certificats non traçables en utilisant une technique unique, que ce soit les preuves de connaissance ou les techniques liées au "chiffrement basé sur l'identité". Nous espérons ainsi pouvoir combiner les différentes caractéristiques de ces approches.

Dans cette thèse, nous avons proposé des mécanismes pour construire et prouver un historique. Cependant, il est encore nécessaire de formaliser les méthodes permettant de sélectionner les informations à révéler et les techniques permettant d'estimer un niveau de confiance à partir des informations prouvées.

Lexique Anglais-Français

Cette liste rappelle certaines terminologies anglaises employées dans ce manuscrit et les termes techniques correspondant que nous avons adoptés dans ce résumé en Français.

Anglais	**Français**
Artifact / Communicating device	Objet / objet communicant
Access Control List	Liste de contrôle d'accès
Bit commitment	Mise en gage de bit
Challenge-response protocol	Protocole de défi/réponse
Context awareness	Prise en compte du contexte
Distance-bounding proof of knowledge	Preuve de connaissance et de proximité
Electronic cash	Argent électronique
Group signature	Signature de groupe
History	Historique (des interactions)
Location-stamping	"Géodatage"
Mafia fraud	Attaque par relais
Off-line, disconnected	Déconnecté
Pervasive / ubiquitous computing	Informatique diffuse / omniprésente
Privacy	Protection de la vie privée
Proof of knowledge	Preuve de connaissance (interactive)
Prover	Vérifié
Sensor Network	Réseau de capteurs
Signature based on a proof of knowledge	Signature basée sur une preuve de connaissance
Terrorist Fraud	Attaque par relais avec collusion
Time-stamping	Horodatage
Trust	Confiance
Trusted third party	Tiers de confiance
Unlinkable credential	Certificat non traçable
Verifier	Vérificateur

CV and Publications

Laurent Bussard is a PhD candidate at ENST and is working in the Network Security Team at Eurecom Institute, Sophia-Antipolis, France. He is interested in security of pervasive computing environments in terms of access control, trust establishment, and privacy. He received his MS in networks and distributed systems from the ESSI in 2000. From 1995 to 1999, he worked as an engineer in software development at Siemens where he was involved in projects related to the telecommunication management network (TMN).

The remainder of this section lists papers related to security that have been written since the beginning of this Ph.D. thesis. Project reports and extended versions of papers have been omitted.

International Conferences and Workshops

[BR02] **Authentication in ubiquitous computing**
L. Bussard and Y. Roudier. Workshop on Security in Ubiquitous Computing at UBICOMP'02. 2002

Preliminary work related to distance-bounding (Chapter 4).

[LBR02] **Extending tamper-proof hardware security to untrusted execution environments**
S. Loureiro, L. Bussard, and Y. Roudier. In Proceedings of the Fifth Smart Card Research and Advanced Application Conference (CARDIS'02) - USENIX, pages 111–124. November 2002.

Work on mobile code protection (out of the scope of this dissertation).

[BR03b] **Embedding distance-bounding protocols within intuitive interactions**
L. Bussard and Y. Roudier. In Proceedings of Conference on Security in Pervasive Computing (SPC'2003), volume 2802 of LNCS, pages 143–156. Springer, March 2003.

Work on user-friendly distance-bounding (Chapter 4 and Appendix B).

[BRKC03] **Trust and authorization in pervasive B2E scenarios**
L. Bussard, Y. Roudier, R. Kilian Kehr, and S. Crosta. In Proceedings of the 6th Information Security Conference (ISC'03), volume 2851 of LNCS, pages 295–309. Springer, October 2003.

Our contribution to the access control of WiTness project (Chapter 6).

[WBRR04] **Security and trust issues in ubiquitous environments - the business-to-employee dimension**
T. Walter, L. Bussard, P. Robinson, and Y. Roudier. Workshop on Ubiquitous Services and Networking in at SAINT'04. 2004.

Framework developped in WiTness project (Out of the scope of this dissertation).

[BM04b] **One-time capabilities for authorizations without trust**
L. Bussard and R. Molva. In Proceedings of the second IEEE conference on Pervasive Computing and Communications (PerCom'04), pages 351–355, March 2004.

One-time credentials with embedded e-check (Chapter 1).

[BRM04] **Untraceable secret credentials: Trust establishment with privacy**
L. Bussard, Y. Roudier, and R. Molva. In Proceedings of the Workshop on Pervasive Computing and Communications Security (PerSec'04) at PerCom'04, pages 122–126, March 2004.

Credentials with secret attributes (Chapter 2).

[BMR04b] **History-based signature or how to trust anonymous documents**
L. Bussard, R. Molva, and Y. Roudier. In Proceedings of the Second Conference on Trust Management (iTrust'2004), volume 2995 of LNCS, pages 78–92. Springer, March 2004.

Unlinkable credential scheme (Chapter 3).

[BM04a] **Establishing trust with privacy**
L. Bussard and R. Molva. To appear in proceedings of the twelve international workshop on security protocols, April 2004.

Unlinkable credential scheme (Chapters 3 and 5).

[BR04] **Protecting applications and devices in nomadic business environments**
L. Bussard and Y. Roudier. In Proceedings of 3rd Conference on Security and Network Architectures (SAR'04), pages 243–252, June 2004.

WiTness and student projects: pragmatic way to protect pieces of code and environmnents (Appendix E).

Pending Submissions and Unpublished Reports

[BR03a] **Background signature for sensor networks**
L. Bussard and Y. Roudier. Technical Report RR-03-076, June 2003.

Not published, preliminary work on signature schemes (out of the scope of this dissertation).

[BMR04a] **Combining history-based trust establishment with distance-bounding protocols**
L. Bussard, R. Molva, and Y. Roudier. Technical Report RR-04-100, April 2004.

Not published, how to merge distance-bounding protocols and unlinkable credentials (Chapter 5).

[BCC^{+}04] **Can we take this off-line? how to deal with credentials in federations without global connectivity**
L. Bussard, J. Claessens, S. Crosta, Y. Roudier, and A. Zugenmaier. Technical Report RR-04-105, May 2004.

A shorter version has been submitted for publication in 2004, joint work with Microsoft on Web Services Security (out of the scope of this dissertation).

[BHKK^{+}04] **Secure Mobile Business Applications – Framework, Architecture and Implementation**
L. Bussard, J. Haller, R. Kilian-Kehr, J. Posegga, P. Robinson, Y. Roudier, and T. Walter. Submitted for publication in a journal.

Describes the framework developped in WiTness (out of the scope of this dissertation).

[BB04b] **Distance-bounding proof of knowledge protocols to avoid terrorist fraud attacks**
L. Bussard and W. Bagga. Technical Report RR-04-109, May 2004.

A shorter version has been submitted for publication in 2004, describes the distance-bounding proof of knowledge scheme (Chapter 4).