Establishing Trust with Privacy

Laurent Bussard, Refik Molva

Institut Eurécom
Corporate Communications
2229, route des Crêtes BP 193
06904 Sophia Antipolis (France)
{bussard,molva}@eurecom.fr

1 Introduction

With the advent of self-organizing systems such as ad hoc networks or ubiquitous computing, security protocols have to meet a new requirement for establishing trust among parties that have no a priori relationship like a shared naming structure or a common organization. Trust establishment in this context calls for a brand new paradigm with respect to classical scenarios whereby entities build trust based on some existing security association. We suggest in this paper a cryptographic protocol through which parties can build trust based on the history of their interactions with other parties. This protocol allows a prover to get a proof of history or 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 proof of history.

Privacy is an essential requirement for such a protocol since providing proof of history to several parties without privacy would severely expose the behavior of the prover. The history-based trust establishment protocol thus assures the anonymity of the prover and the unlinkability of interactions using the proof of history. Moreover, the prover can choose to show only parts of its history. The proof of history is based on a signature mechanism and the trust establishment protocol is a challenge-response protocol based on this mechanism. The signature mechanism is an extension of group signatures.

2 History-Based Signature

This section shows the interactions necessary to build a provable history and to use this for history-based trust establishment. Users collect evidence of their activity and store it as a provable history. To ensure non-transferability of evidences, they are implemented as credentials attached to a valuable secret. Credentials can define group membership, location-and-time stamps, recommendations, etc.

When signing a challenge or a document, the user chooses some credentials in his history, modifies them, and signs with those credentials. Credentials have to fulfill the following requirements to build a provable yet anonymous history: Non-transferability, i.e. credentials can only be used by the owner of some valuable secret (equivalent to the private key in public key infrastructures); Anonymity, i.e. use of history-based credentials should not reveal the identity of the author; and untraceability, i.e. it is not possible to link different signatures based on the same credential.

History-based signature is an extension of the group signature scheme described in [2]. Alice (A) is the signer. She collects some credentials to subsequently prove some history. For instance, A holds a credential 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 some membership, e.g. employee of a company, member of ieee computer society, partner of some project, member of a golf club, citizen of some state, client of some bank, customer of some airline. A can show some recommendations: when she collaborates with other entities, she receives credentials. 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.

2.1 Certification by a CA or Group Manager

To initiate the system, each entity has to get some certificate proving that he/she has a valid secret, i.e. a secret linked to his/her identity. This part is similar to the join protocol of the Camenisch's scheme. However, we use a modified version because a coalition attack exists against the initial scheme [1, 3]. We define the following elements: $n = pq$ 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; $a, b \in \mathcal{Z}_n^*$ are elements of the multiplicative group; and λ is a security parameter (see [2] for more details).

In Table 1, A generates some secret x with the help of a CA or group manager B. Moreover, A receives a certificate on this secret x: $\text{cert}_{1b} = (a_b^x + 1)^{d_b} \bmod n_b$.

2.2 Obtaining Context Proofs or Recommendations

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 2 shows how A can get a credential from C. The identity of A is not known but C 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, C has to trust B otherwise the previous protocol has to be done once more. However, when an entity D needs to verify the signature of A on some document, D only has to know C.

Two interactive proofs of knowledge (PK) are done in step 2.3). The first one proves that y_2 is based on some secret. The second shows that this secret has been certified by B. Indeed, $\tilde{z}\tilde{g}_b = \tilde{g}_b{}^{(\beta^{e_b})} = \tilde{g}_b{}^{(a_b^\alpha)}\tilde{g}_b = \tilde{g}_b{}^{(1+a_b^\alpha)}$ and thus $1 + a_b^\alpha = \beta^{e_b}$. It means that A knows $\beta = (1 + a_b^\alpha)^{d_b}$ that is a certification of α,

A		**B**
		private: p_b, q_b, d_b public: $n_b, e_b, G_b, g_b, a_b, \lambda_b$
1.1) chooses random secret x' $x' \in_R \{0, 1, ..2^{\lambda_b} - 1\}$		
	1.2) $y' = a_b^{x'} \bmod n_b$ $\longrightarrow$	
	1.3) $\xi \in_R \{0, 1, .., 2^{\lambda_b} - 1\}$ $\longleftarrow$	
1.4) computes $x = x' + \xi$ $y = a_b^x \bmod n_b$ commits to $z = g_b^y$		
	1.5) y, z $\longrightarrow$	
	1.6) $\mathrm{PK}[\alpha \mid y = g_b^\alpha]$ $\rightleftarrows$	
		1.7) verifies $y \overset{?}{=} y' \cdot a_b^\xi$
	1.8) $\mathrm{cert}_{1b} = (y + 1)^{d_b} \bmod n_b$ $\longleftarrow$	

Table 1. Creation and first certification of A's secret x

A		**C**
private: $x, (a_b^x + 1)^{d_b}$		private: $p_c, q_c, d_c, d_{c_1}, \dots d_{c_k}$ public: $n_c, e_c, e_{c_1}, \dots e_{c_k}$, $G_c, g_c, a_c, b_c, \lambda_c$
2.1) $y_2 = a_c^x \bmod n_c$ $\tilde{g_b} = g_b^r$ for $r \in_R \mathcal{Z}_{n_b}$ $\tilde{z} = \tilde{g_b}^y$ (i.e. $\tilde{z} = z^r$)		
	2.2) y_2 $\longrightarrow$	
	2.3) pk_2: $\mathrm{PK}[\alpha \mid y_2 = a_c^\alpha \wedge \tilde{z} = \tilde{g_b}^{(a_b^\alpha)}]$ pk_3: $\mathrm{PK}[\beta \mid \tilde{z}\tilde{g_b} = \tilde{g_b}^{(\beta^{e_b})}]$ $\rightleftarrows$	
		2.4) $t \in_R \{0, 1, \dots, 2^\lambda - 1\}$ $\mathrm{cert}_{1c} = (a_c^x + 1)^{d_c}$ $\mathrm{cert}_{2c} = (a_c^x + b_c^t)^{d_h}$ $\mathrm{cert}_{3c} = (b_c^t + 1)^{d_c}$ where $d_h = \prod_{i \in S} d_i$
	2.5) t, cert_{1c}, cert_{2c}, cert_{3c}, S $\longleftarrow$	

Table 2. Obtaining some credential to build history

which is also the discrete logarithm of y_2 to the base a_c. In other words, y_2 has been computed from the same secret x.

In step 2.4) A receives a new credential $\mathrm{cert_{2c}} = (a_c^x + b_c^t)^{d_h} \bmod n_c$ from C that will be used to prove some history. b_c as well as a_c are elements of $\mathcal{Z}_{n_c}^*$, x prevents the transferability of credentials, and t is different for each credential to forbid a user from combining multiple credentials (see Section 4). The attribute value, be it a location or a recommendation, is defined using a technique that comes from electronic cash: $d_h = \prod_{i \in S} d_{c_i}$ where S is a set that defines the amount or any attribute. Construction of d_h is given in Section 3. Two other credentials can be provided: $\mathrm{cert_{1c}} = (a_c^x + 1)^{d_c} \bmod n_c$ is a certification of the secret that can replace $\mathrm{cert_{1b}}$. To avoid a potential attack (see Section 4), we add $\mathrm{cert_{3c}} = (b_c^t + 1)^{d_c} \bmod n_c$.

2.3 Using History for Signing

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

$$\begin{aligned}
&\hat{g}_c = g_c^{r_1},\ \hat{z}_2 = \hat{g}_c^{\,y_2},\ \text{and}\ \hat{z}_3 = \hat{g}_c^{\,(b_c^t)} \\
&spk_1 = \mathrm{SPK}[\alpha \mid \hat{z}_2 = \hat{g}_c^{\,(a_c^\alpha)}](m) \\
&spk_2 = \mathrm{SPK}[\beta \mid \hat{z}_2\hat{g}_c = \hat{g}_c^{\,(\beta^{e_c})}](m) \\
&spk_3 = \mathrm{SPK}[\delta \mid \hat{z}_3 = \hat{g}_c^{\,(b_c^\delta)}](m) \\
&spk_4 = \mathrm{SPK}[\gamma \mid \hat{z}_2\hat{z}_3 = \hat{g}_c^{\,(\gamma^{e_{h'}})}](m) \quad \text{where} \quad e_{h'} = \textstyle\prod_{i \in S'} e_i \text{ and } S' \subseteq S \\
&spk_5 = \mathrm{SPK}[\epsilon \mid \hat{z}_3\hat{g}_c = \hat{g}_c^{\,(\epsilon^{e_c})}](m)
\end{aligned}$$

The signature of message m is $\{spk_1, spk_2, spk_3, spk_4, spk_5, \hat{g}_c, \hat{z}_2, \hat{z}_3, S'\}$. The signatures of knowledge spk_1 and spk_2 prove that the signer knows $\mathrm{cert_{1c}}$: $\beta = (1 + a_c^\alpha)^{d_c} \bmod n_c$. The signatures of knowledge spk_1, spk_3 and spk_4 prove that the signer knows cert'_{2c}: $\gamma = (a_c^\alpha + b_c^\delta)^{d_{h'}} \bmod n_c$. To avoid some potential attack (see Section 4), we added spk_5 to prove the knowledge of $\mathrm{cert_{3c}}$. spk_3 and spk_5 prove that t was generated by C: $\epsilon = (1 + b_c^\delta)^{d_c} \bmod n_c$.

3 Encoding Attribute Values

In the protocol, Alice receives $\mathrm{cert_{2c}}$ and signs with cert'_{2c} to hide part of the attributes. A flexible mechanism is necessary for displaying part of the attributes.

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

A signature $SIGN_{(S,n)}(m) = m^{d_h} \bmod n$, where S is a set and $d_h = \prod_{i \in S} d_i$, can then be transformed into a signature $SIGN_{(S',n)}(m) = m^{d_{h'}} \bmod n$, where

S' is a subset of S and $d_{h'} = \prod_{i \in S'} d_i$. The the attribute value is coded as a set S corresponding to its bits equal to one. This signature based on set S can be reduced to any subset $S' \subseteq S$:

$$SIGN_{(S',n)}(m) = \left(SIGN_{(S,n)}(m)\right)^{\left(\prod_{i \in \{S \setminus S'\}} e_i\right)} = m^{\left(\prod_{i \in S'} d_i \bmod \phi(n)\right)} \bmod n$$

Thus, an entity that received some credential cert_{2c} is able to compute cert'_{2c} and to sign a document with this new credential.

$$\text{cert}'_{2c} = (\text{cert}_{2c})^{\prod_{j \in \{S \setminus S'\}} e_j} = \left(\left(a_c^x + b_c^t\right)^{\prod_{i \in S} d_i}\right)^{\prod_{j \in \{S \setminus S'\}} e_j} = \left(a_c^x + b_c^t\right)^{\prod_{i \in S'} d_i}$$

This technique ensures that part of the signed attributes can be modified. A location and time stamper (LTS) can certifies that some entity has been at a given place at a given time: `[time: 180432, date: 30012004, latitude: 436265, longitude: -0070470]`. If a location and time stamper provides the following credential to Alice:
`[18|04|32, 30|01|2004, 43|62|65, -007|04|70]`, she can sign a document with a subset of this credential.
`[18|XX|XX, XX|XX|XXXX, 43|62|65, -007|04|70]`, i.e. the document is signed by *someone that was in the building someday around six o'clock*. Or
`[XX|XX|XX, 30|01|2004 43|XX|XX, -007|XX|XX]`, i.e. *someone who was in the South of France the 30th of January.*

Similarly, a company can qualify customers as *Platinum, Gold, or Silver*; a state can provide digital Id cards to citizen to certify gender, name; a company can provide credentials that define role, access rights; and a partner can define recommendations. In all those cases, the ability of selecting which attribute is displayed is very important to protect privacy when enabling trust evaluation.

4 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. In addition it is based on the security of Schnorr and RSA signature schemes and on the additional assumption of [2] that computing membership certificates is hard. The signature produced by the above protocol is not forgeable. Specifically, only an entity having received a given credential could have issued this signature.

Secret t has been added to that attributes of multiple credentials could be combined. 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 $a^x + b^t = a^{x'} + b^{t'}$. In order to prevent transferability based on such generation of equivalent pairs, cert_{3c} 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. A proof that the generation of pairs is equivalent to a difficult problem would allow for important simplifications of the history-based signature scheme.

5 Conclusions

This paper introduces a *history-based signature* scheme that enables to build trust based on the evidence of previous interactions without revealing them. This scheme can be useful in two ways: first it allows anonymous signatures, e.g. some article can be signed by an *art critic that visited some museum one week ago*. Next it can be used in a challenge-response protocol to prove one's history. For instance, previous successful collaborations can be asserted. User privacy is protected by displaying only necessary attributes and by avoiding traceability.

There are two main limitations to this scheme: it is well-known that signatures based on the proof of knowledge of a double discrete logarithm are not efficient in terms of computational complexity and the deployment is dedicated to client-server model but does not allow peer-to-peer collaborations where each entity acts as a signer and as a credential provider.

References

1. G. Ateniese and G. Tsudik. *Some open issues and new directions in group signatures.* In Proceedings of Financial Cryptography99, volume 1648 of LNCS, pages 196211. Springer-Verlag, 1999.
2. J. Camenisch and M. Stadler. *Efficient group signature schemes for large groups.* In Advances in Cryptology, CRYPTO '97 Proceedings, LLNCS 1294, pages 410–424, Santa Barbara, CA, August 1997.
3. Zulfikar Amin Ramzan. *Group blind digital signatures: Theory and applications*, Master Thesis, MIT, 1999.