History-Based Signature or How to Trust Anonymous Documents

Laurent Bussard, Refik Molva, and Yves Roudier

Institut Eurécom[1]
Corporate Communications
2229, route des Crêtes BP 193
06904 Sophia Antipolis (France)
{bussard,molva,roudier}@eurecom.fr

Abstract. This paper tackles the following problem: how to decide whether data are trustworthy when their originator wants to remain anonymous? More and more documents are available digitally and it is necessary to have information about their author in order to evaluate the accuracy of those data. Digital signatures and identity certificates are generally used for this purpose. However, trust is not always about identity. In addition authors often want to remain anonymous in order to protect their privacy. This makes common signature schemes unsuitable. We suggest an extension of group signatures where some anonymous person can sign a document as a friend of Alice, as a French citizen, or as someone that was in Paris in December, without revealing any identity. We refer to such scheme as *history-based signatures*.

1 Introduction

Verifying the reliability of a piece of information without revealing the identity of its source is becoming an important privacy requirement. Anybody can easily broadcast inaccurate or even deliberately deceptive information like in the case of what is referred as urban legends or hoaxes. Author authentication thanks to the signature of that very document seems a natural way to check whether the author can be trusted and thus to determine whether the document is accurate or misleading. Furthermore, 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 not to be traced based on their opinions. When users have a way to attach comments to surrounding physical objects [10] (e.g. painting in a museum) the chance that statistics be made on their interests might simply refrain them from commenting at all.

[1] Institut Eurécom's research is partially supported by its members: Bouygues Télécom, Cegetel, France Télécom, Hasler Foundation, Hitachi, STMicroelectronics, Swisscom, Texas Instruments, and Thales.

There are number of cases like pervasive computing or ad-hoc networks in which infrastructure is lacking: neither a public key infrastructure nor a web of trust is available which renders identity-based authentication impossible [13]. Even with an infrastructure, authenticating the author is often not sufficient and more information on the *context*, in which the document was created, is required. For instance, beginning of this year the mass media announced that a senior radio reporter in Swaziland pretending to be reporting live from the war front in Iraq had never left his country and was broadcasting from a broom closet. This case shows that the context (*being in some place*) is sometimes more important than the role or the identity of the author (*being who he pretends to be*). Group signature schemes [5] make one step forward towards such new requirements by assuring the anonymity of the signer when revealing some information on his relationships, i.e. group membership. This paper extends this concept using attributes embedded within each signature in order to enable the evaluation of trust information on any signed document without revealing the identity of the author.

Various attributes can be relevant to evaluate trust. When some clear hierarchy exists among entities, a public key infrastructure [8] is sufficient to define trust relationships. A *web of trust* [9] 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 [7]: *personal observations* of the entity's behavior, *recommendations* from trusted third parties, and *reputation* of an entity. However, other sources of information exist: sometimes, the *physical context* is also taken into account in the trust evaluation [14, 11]. In a simple example, any person present in a room can be authorized to turn on the light. In this paper, we add the notion of *proof of context*, which certifies that some entity has been to some location at some time. It provides evidence for trustworthiness based on contextual parameters such as location and history.

This paper suggests a new signature scheme that takes those sources of trust into account. The scheme ensures anonymity and untraceability of signers. When signing, authors choose which part of their history will be shown to readers. For instance, a report relating some event can be signed by *an employee who was there when this event occurred*; an e-mail can be signed by *an inhabitant of a given district of a town*; or an article could be signed by *a member of a trade union who attended a given demonstration*. Like this, the signature is not based anymore on the identity of the signer but rather on his 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 degree of accuracy of the details he wants to disclose, 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.

The remaining of the paper is organized as follows: section 2 presents the requirements and some related work. Section 3 describes the group signature scheme that is modified in Section 4 to define a history-based signature scheme.

Section 5 introduces a mechanism to code context and relation so that these can only be modified in a controlled way. Finally, Section 6 evaluates the security of this scheme.

2 Problem Statement

This section gives an overview of the interactions necessary to build a provable history and to use this for history-based signatures. Related work is discussed with respect to the feasibility of a provable history scheme.

2.1 Principle

Users anonymously collect evidence of their activity and store it as a provable history. In Figure 1, a user gets a proof that he has been at a location. 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.

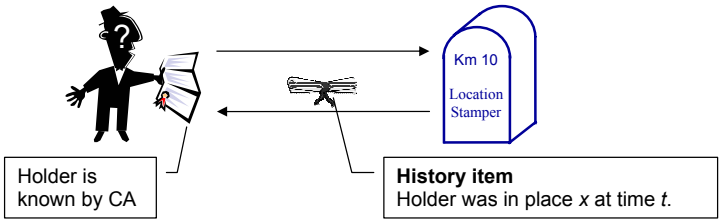

Fig. 1. Getting history items

When signing a document, the author chooses some credentials in his history, modifies them, and signs the document with those credentials. In Figure 2, a user is able to prove that he was at a location x at time t, that he is said reliable by some entity Z, that he is a member of group G, and that he has a given name and address (electronic id card). He chooses to sign the document as *someone that was at location x at time t*. The signature does not reveal more information on the signer and it is even not possible 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 he was in the building at some time.

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

Fig. 2. History-based signature

- *Non-transferability*: 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.
- *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.

2.2 Related Work

Some existing work [4, 2] already allow for privacy-preserving attribute verification. However, the target of those works is anonymous attribute certificates and untraceable access control. Credentials defined in [4] rely on pseudonyms and thus it is necessary to know the verifier before starting the challenge-response protocol. Credentials defined in [2] do not ensure non-transferability and have to be used only once to ensure untraceability. The one-time property of these credentials also does not suit multiple interactions as required by our scenario.

Using information on the user's context to evaluate trust or define rights is not new: [6] proposes a generalization of the role-based access control paradigm taking into account contextual information. Location verification techniques range from ultrasound-based challenge response [14] to distance bounding protocols [3], which forbid Mafia fraud attacks and thus defeat collusion of insiders. In this paper we assume that the location stamper implements one of those techniques to verify the presence of entities before delivering a proof of location.

3 Basic Mechanisms

This section presents the first group signature of Camenisch [5] that will be modified in the sequel of this paper in order to define a history-based signature scheme.

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; $a \in \mathcal{Z}_n^*$ is an element of the multiplicative group; and λ is a security parameter (see [5] for more details).

3.1 Interactive Proof of Knowledge

A *proof of knowledge* (PK) allows an entity to prove the knowledge of some secret without revealing this secret. For instance, the prover P claims to know the double discrete logarithm of y to the bases g and a. The verifier V tests if P indeed knows x. This is denoted $\mathrm{PK}[\alpha \mid y = g^{(a^{\alpha})}]$.

P sends a witness to V: $w = g^{(a^r)}$ where r is a random value and V returns a random challenge bit $c \in_R \{0,1\}$. Finally P sends a response $s = r$ (if $c = 0$) or $s = r - x$ (if $c = 1$). The verifier checks that

$$\begin{aligned} c = 0 \ : \ w \stackrel{?}{=} g^{(a^s)} \quad &= g^{(a^r)} \\ c = 1 \ : \ w \stackrel{?}{=} y^{(a^s)} \quad &= \left(g^{(a^x)}\right)^{(a^s)} = g^{(a^{x+s})} = g^{(a^r)} \end{aligned}$$

This protocol has to be run l times where l is a security parameter.

3.2 Signature based on a Proof of Knowledge

A *signature based on a proof of knowledge* (or signature of knowledge) of a double discrete logarithm of z to the bases g and a, on message m, with security parameter l is denoted $\mathrm{SPK}_l[\alpha \mid z = g^{(a^{\alpha})}](m)$. It is a non-interactive version of the protocol depicted in Section 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 $1 \leq i \leq l$, 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 $s_i = \begin{cases} r_i & \text{if } c[i] = 0 \\ r_i - x & \text{otherwise} \end{cases}$

3.3 Camenisch's Group Signature

The group signature scheme in [5] 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 e^{th} 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 $de = 1 \bmod \phi(n)$. When Alice *joins* the group, i.e. becomes a member, she uses her secret x to compute a membership key (y, z) where $y = a^x \bmod n$ and $z = g^y$. A sends (y, z) to the group manager, proves that she knows x and receives a group certificate $(y+1)^d \bmod n$ 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 e^{th} root of the discrete logarithm that can be computed using the certificate $(y+1)^d \bmod n$. 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 $\beta = (a^\alpha + 1)^d \bmod n$. 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.

4 Solution: History-Based Signature Scheme

History-based signature is an extension of the group signature scheme described in Section 3. Alice (A) is the signer. She collects some credentials to subsequently prove some history. 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 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.

4.1 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, 12].

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: $\text{cert}_{1b} = (a_b^x + 1)^{d_b} \bmod n_b$. Now, A is certified and can act anonymously as a member of group or as an entity certified by a given CA in order to get credentials and build a provable history.

4.2 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

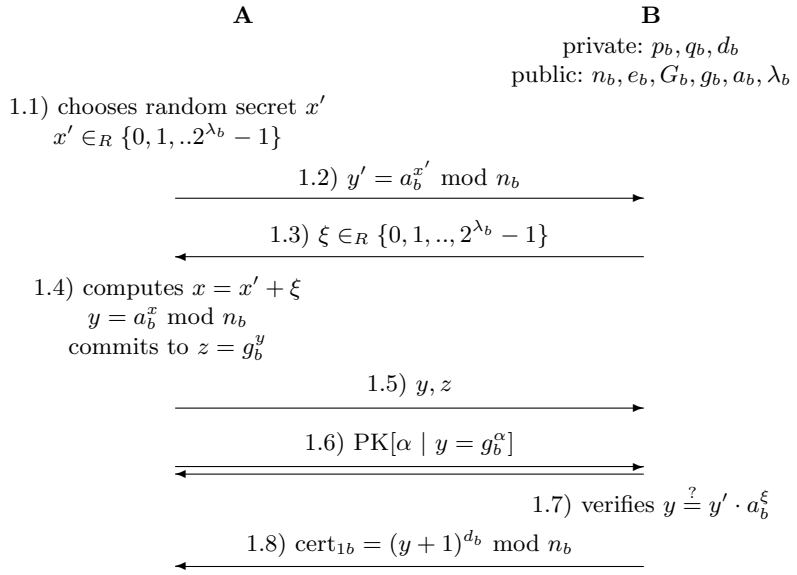

Table 1. Creation and first certification of A's secret x

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 proofs of knowledge 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, 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 $\text{cert}_{2c} = (a_c^x + b_c^t)^{d_h} \bmod n_c$ 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 6). 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 5. Two other credentials can be provided: $\text{cert}_{1c} = (a_c^x + 1)^{d_c} \bmod n_c$ is a certification of the secret that can replace cert_{1b}. To avoid a potential attack (see Section 6), we add $\text{cert}_{3c} = (b_c^t + 1)^{d_c} \bmod n_c$.

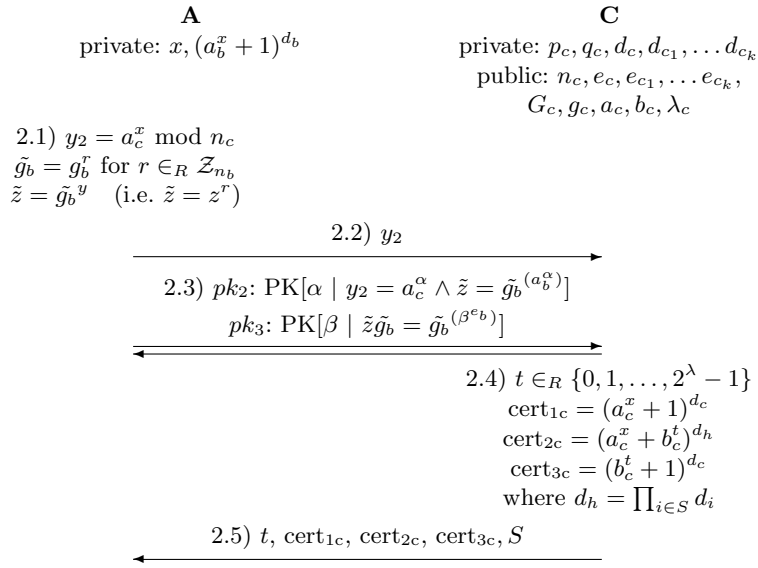

Table 2. Obtaining some credential to build history

4.3 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 (cert_{1c}), and some credential that is part of her history (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:

$$
\begin{aligned}
&\hat{g_c} = g_c^{r_1},\ \hat{z_2} = \hat{g_c}^{y_2}, \text{ and } \hat{z_3} = \hat{g_c}^{(b_c^t)} \\
&spk_1 = \text{SPK}[\alpha \mid \hat{z_2} = \hat{g_c}^{(a_c^\alpha)}](m) \\
&spk_2 = \text{SPK}[\beta \mid \hat{z_2}\hat{g_c} = \hat{g_c}^{(\beta^{e_c})}](m) \\
&spk_3 = \text{SPK}[\delta \mid \hat{z_3} = \hat{g_c}^{(b_c^\delta)}](m) \\
&spk_4 = \text{SPK}[\gamma \mid \hat{z_2}\hat{z_3} = \hat{g_c}^{(\gamma^{e_{h'}})}](m) \quad \text{where} \quad e_{h'} = \textstyle\prod_{i \in S'} e_i \text{ and } S' \subseteq S \\
&spk_5 = \text{SPK}[\epsilon \mid \hat{z_3}\hat{g_c} = \hat{g_c}^{(\epsilon^{e_c})}](m)
\end{aligned}
$$

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 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 6), we added spk_5 to prove the knowledge of cert_{3c}. spk_3 and spk_5 prove that t was generated by C: $\epsilon = (1 + b_c^\delta)^{d_c} \bmod n_c$.

When credentials from different entities (e.g. B and C) have to be used together, it is necessary that A generate a random number $r_2 \in_R \mathcal{Z}_{n_b}$

and compute $\hat{g_b} = g_b^{r_2}$ and $\hat{z} = \hat{g_b}^y$ $(= z^{r_2})$. spk_1 and spk_2 are modified as follows:

$$spk'_1 = \text{SPK}[\alpha \mid \hat{z_2} = \hat{g_c}^{(a_c^\alpha)} \wedge \hat{z} = \hat{g_b}^{(a_b^\alpha)}](m)$$
$$spk'_2 = \text{SPK}[\beta \mid \hat{z}\hat{g_b} = \hat{g_b}^{(\beta^{e_b})}](m)$$

spk'_1 and spk'_2 prove that the signer knows $\text{cert}_{1\text{b}}$: $\beta = (a_b^\alpha + 1)^{d_b} \bmod n_b$ and spk'_1 proves that $\text{cert}_{1\text{b}}$ and $\text{cert}_{2\text{c}}$ are linked to the same secret x. spk'_1 is a signature based on a proof of equality of two double discrete logarithms (see Appendix A). The new signature of message m is $\{spk'_1, spk'_2, spk_3, spk_4, spk_5, \hat{g_b}, \hat{z}, \hat{g_c}, \hat{z_2}, \hat{z_3}, S'\}$.

5 Encoding Attribute Values

In Section 4, the user receives $\text{cert}_{2\text{c}}$ and signs with $\text{cert}'_{2\text{c}}$ to hide part of the attributes when signing. This section presents a flexible mechanism for atteibute 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, this requires too much credentials when transversal attributes have different granularities (longitude, latitude, time, etc.).

5.1 Principle

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 $S' \subseteq S$:

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

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

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

This technique ensures that part of the signed attributes can be modified. For instance, the 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. $d_h = d_4 \cdot d_3 \cdot d_1 \bmod \phi(n)$. 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).

5.2 Possible Codes

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

- *more-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 bits equal to one can be replaced by zeros, it can be transformed into $v' \in \{13, 12, 9, 8, 5, 4, 1\}$.
- *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 $v' \in \{13, 15, 29, 31\}$.
- *unary more-or-equal*: the problem with binary encoding is that they cannot be reduced to any value. For instance, $7_d = 111_b$ can be shown as 7, 6, 5, 4, 3, 2, 1, or 0 but $6_d = 110_b$ can only be shown as 6, 4, 2, or 0. This limitation can be solved by using a binary representation of unary: $v = 6_d = 111111_u \rightarrow 0111111_b \rightarrow S = \{1, 2, 3, 4, 5, 6\}$ can be shown as $v' \in \{6, 5, 4, 3, 2, 1, 0\}$. The overhead is important (l bits data is encoded with 2^l bits) and thus unary has to be restricted to small values.
- *unary less-or-equal*: unary representation a similar approach can be used for less-or-equal too: $v = 2_d \rightarrow 1111100_b \rightarrow S = \{3, 4, 5, 6, 7\}$ can be transformed in $v' \in \{2, 3, 4, 5, 6, 7\}$.
- *frozen*: values are encoded so that they cannot be changed. In this case, the number of bits have to be larger: l bits becomes $l + \lfloor \log_2(l) \rfloor + 1$ 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 increase the number of zero but the checksum can only be decreased. It is not possible to change frozen values.
- *blocks*: data are cut into blocks. Each block is encoded with one of the previous schemes.

5.3 Example: Location-and-Time Stamper

This section describes how the previous encoding schemes can be used. 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 some distance bounding protocol. 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 3 shows an example of the attributes that could be delivered by some LTS in Eurecom Institute.

Value	**Meaning**
180432	UTC in hhmmss format (18 hours, 4 minutes and 32 seconds)
24112003	Date in ddmmyyyy format (November 24, 2003)
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 3. Context data: location and time

It can be represented by four attributes `[180432, 24112003, 436265, -0070470]` that can be divided into frozen blocks: `[18|04|32, 24|11|2003, 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 of the time in seconds (0-59 : 6 bits), e_6 is the most significant bit 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|11|2003, 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, 24|11|2003, 43|XX|XX, -007|XX|XX]`, i.e. *someone who was in the South of France the* 24^{th} *of November*.

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. Thus it is not possible to convert 09:08:30 into 09:00:30. The only way to suppress minutes is to remove seconds as well: 09:`XX:XX`. This value does not mean that some action occurred at nine o'clock but that it occurred between nine and ten o'clock.

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.

6 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 [5] that computing membership certificates is hard.

Our proposal is based on the group signature scheme of [5], whose join protocol is subject to a collusion attack [1]. Modifications suggested in [12] and that prevent this attack have been taken into account (see Table 1). Even with this modification, there is no proof that the scheme is secure. The security does, however, rest on a well-defined number-theoretic conjecture.

6.1 Unforgeability of Signature

The signature produced by the above protocol is not forgeable. Specifically, only an entity having received a given credential could have issued this signature. This holds because, in the random oracle model, 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 integers α and δ are known by the signer. On the other hand, 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.

6.2 Unforgeability and Integrity of Credentials

In order to code 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 each d_i is kept secret and each modulo n is 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 easier brute force attacks that aim at creating a valid signature for a given message: an attacker can choose a message m and a random $d_R \in_R \mathcal{Z}_n$ and compute a signature $m^{d'} \bmod n$. If e_i and d_i are defined for $i \in \{1, \ldots, k\}$, there are 2^k valid $d = \prod_{i \in S' \subseteq S} d_i$. The probability

that a random d_R be acceptable is 2^k times higher than with standard RSA where $k = 1$. However, even if the number of possible signatures for a given message increases, it is necessary to find out the set S corresponding to the randomly chosen signature. In other words, the attacker has to test whether $\forall S' \subseteq S \mid m \stackrel{?}{=} (m^{d'})^{\prod_{i \in S'} e_i} \bmod n$. There are 2^k possible sets S' to check and thus the security of this scheme 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 h_1 states that Alice was present from 8 a.m. to 10 a.m. and h_2 states that she was present from 4 p.m. to 6 p.m., it is necessary to forbid that Alice could create a h' stating that she was present from 8 a.m. to 6 p.m. To avoid this attack, a unique secret t is associated to each credential. Hence $(a^x + b^{t_1})^{d_{h_1}}$ cannot be combined with $(a^x + b^{t_2})^{d_{h_2}}$.

6.3 Non-Transferability of History

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 $a^x + b^t = a^{x'} + b^{t'}$. 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. 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 equivalent pairs is equivalent to a difficult problem (e.g. the discrete logarithm problem) would allow for important simplifications of the history-based signature scheme.

7 Conclusions and Future Work

This paper introduces a *history-based signature* scheme that makes it possible to sign data with one's history. In this scheme, signers collect credentials (proof of location, recommendation, etc.) in order to build a provable history. This scheme preserves the privacy of authors and makes a large variety of attributes possible for defining trust: recommendations, contextual 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 some

parameters defining trustworthiness, i.e. art critic, location, or recommended by the museum. The authors of note have a guarantee that they cannot be traced. 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.

There are two main limitations to this scheme. First, 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. It could be interesting to study other approaches to define more efficient history-based signatures. Second, the deployment of the scheme is easy when some authorities (CA, TTP, group manager, LTS, etc.) provide proofs of context and recommendations and some users collect those credentials in order to sign. Peer-to-peer frameworks where each entity acts as a signer and as a credential provider would require the binding of members' secrets with the group manager's keys.

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. S. Brands. *A technical Overview of Digital Credentials*. Research Report, February 2002.
3. L. Bussard and Y. Roudier, *Embedding Distance-Bounding Protocols within Intuitive Interactions*, in Proceedings of Conference on Security in Pervasive Computing (SPC'2003), Boppard, Germany, March, 2003.
4. J. Camenisch and A. Lysyanskaya, *An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation*, LNCS 2045, 2001.
5. 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.
6. M.J.Covington, M.J.Moyer, and M.Ahamad, *Generalized Role-Based Access Control for Securing Future Applications*. In 23rd National Information Systems Security Conference (2000).
7. Nathan Dimmock. *How much is 'enough'? risk in trust-based access control*, In IEEE International Workshops on Enabling Technologies (Special Session on Trust Management), June 2003.
8. C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T.Ylonen. *Rfc 2693 spki certificate theory*, 1999.
9. Simson Garfinkel. *PGP : Pretty Good Privacy*. International Thomson Publishing, 1995.
10. D. Ingram. *Trust-based filtering for augmented reality*. In Proceedings of the First International Conference on Trust Management, volume 2692. LNCS, May 2003.
11. T.Kindberg, K.Zhang, and N.Shankar, *Context authentication using constrained channels*, in Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications (WMCSA), pages 14–21, June 2002.
12. Zulfikar Amin Ramzan. *Group blind digital signatures: Theory and applications*, Master Thesis, MIT, 1999.

13. J.M. Seigneur, S. Farrell, C.D. Jensen, E. Gray, and Y. Chen *End-to-end Trust Starts with Recognition*, in Proceedings of Conference on Security in Pervasive Computing (SPC'2003), Boppard, Germany, March, 2003.
14. N. Sastry, U. Shankar, and D. Wagner. *Secure verification of location claims*, In Proceedings of the 2003 ACM workshop on Wireless security, 2003.

A Signature Based on a Proof of Equality of Double Discrete Logarithms

Section 4.3 uses a signature based on a proof of equality of two double discrete logarithms (SPKEQLOGLOG).

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

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

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

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 following:

1. For $1 \leq j \leq l$, generate random r_j where $r_j \geq x$.
2. For $1 \leq i \leq k$, for $1 \leq j \leq l$, 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 following:

if $c[j] = 0$: $P_{i,j} = g_i^{\left(a_i^{r_j}\right)} = g_i^{\left(a_i^{s_j}\right)}$

if $c[j] = 1$: $P_{i,j} = g_i^{\left(a_i^{r_j}\right)} = \left(y_i^{\left(a_i^{-x}\right)}\right)^{\left(a_i^{s_j+x}\right)} = y_i^{\left(a_i^{-x} a_i^{s_j+x}\right)} = y_i^{\left(a_i^{s_j}\right)}$

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