Anonymous public-key certificates for anonymous and fair document exchange

N.Zhang, Q.Shi and M.Merabti

Abstract: Two protocols are presented for the issuing and identity tracing of anonymous public-key certificates, used by different parties to engage in an anonymous and fair document exchange without revealing their real identities while still being held accountable for their activities. The certificate issuing protocol allows a party to apply to certificate authorities for anonymous public-key certificates using a certificate already issued. The identity tracing protocol enables a legal authority to trace the pseudonym in an anonymous certificate back to the real identity of the corresponding party. The protocols are also analysed with regard to anonymity and accountability.

1 Introduction

Exchange of valuable documents between different parties is an important activity in electronic commerce, and its applications include certified mail, contract signing, valuable electronic goods exchange, and payment for receipt. Owing to the valuable nature of such documents (e.g. payments and signatures), the exchange must be fair to avoid a situation where some parties can receive their expected documents, while others cannot.

Like any other activities in electronic commerce, accountability is an essential property for fair document exchange. Without its enforcement, fair exchange over the Internet is unlikely to become a reality. In this paper, accountability is concerned mainly with non-repudiation of origin to prevent the sender of a message from falsely denying having sent the message.

Another important property of document exchange is anonymity, which protects the privacy of personal information such as identities and locations. For example, when an individual engages in exchanges of electronic goods with a number of vendors, he/she would like to conceal his/her identity from the vendors to prevent them from assembling a profile of his/her personal interests, lifestyles, whereabouts etc.

Clearly, anonymity is to hide a party's identity, whereas accountability is to expose the party's identity, thereby holding the party responsible for its activities. Therefore, effective solutions are needed to resolve the conflicts between these two properties, in order to achieve anonymous, accountable and fair document exchange between several parties.

Normally, the accountability of a message is achieved by a digital signature on the message produced by the message sender using its private key. To verify the signature, a message recipient ought to hold the sender's public key that must be certified, e.g. by a certificate authority (CA) [1], to provide a secure binding between the sender's identity and public key.

One way to prevent the recipient from knowing the sender's real identity in the certificate is to permit the sender to apply to a CA for an anonymous public-key certificate based on the original certificate. The anonymous certificate includes a new public key and a pseudonym. Only the CA issuing the anonymous certificate can establish a link from the pseudonym to the real identity of the sender. In this way, not only can the sender sign their message using the private key associated with the new public key without disclosing their real identity, but also the recipient is still assured of the message authenticity by the anonymous certificate. Should any dispute about the message arise, the CA can trace the sender's pseudonym back to its real identity and hold the sender accountable for the message.

To date, little work has been done to address the issue of anonymous but traceable public-key certificates. The most relevant work [2] has proposed several schemes for a party to use its original certified public key to acquire an anonymous (public-key) certificate from an CA. Each anonymous certificate of the party includes a public key converted from the original one, and shares the same private key associated with the original public key. The main problems with the schemes are two-fold. First, an anonymous certificate issued by a CA can be linked to the original public key and real identity of the party if the CA is compromised. Secondly, if the party's private key is compromised, all the anonymous certificates of the party may be linked to its original public key and real identity.

The aim of this paper is to propose a scheme for a party to use its original public-key certificate to obtain anonymous certificates from multiple CAs, which are difficult to link together even if some of the CAs are compromised, or the private key associated with the original certificate is compromised. Such anonymous certificates can be used directly by existing fair exchange protocols to achieve anonymous and fair document exchange. The proposed scheme is implemented by two protocols. A certificate issuing protocol allows a party to apply to CAs for multiple anonymous certificates using a certificate already issued.

IEE Proceedings online no. 20000778

DOI: 10.1049/ip-com:20000778

Paper first received 15th October 1999 and in revised form 16th August 2000

N. Zhang is with the Department of Computer Science, University of Manchester, Manchester, M13 9PL, UK

Q. Shi and M. Merabti are with the School of Computing & Mathematical Sciences, Liverpool John Moores University, Liverpool, L3 3AF, UK

An identity tracing protocol enables a legal authority to trace the pseudonym in an anonymous certificate back to the real identity of its associated party.

2 Anonymous public-key certificates

Assume that there exists a number of CAs A_l for issuing public-key certificates. A certificate with the real identity of a party is called a real (public-key) certificate of that party. Assume also that every party possesses at least one real certificate, and that every party can request any CA to issue an anonymous certificate based on its real certificate or one of its anonymous certificates. This means that multiple anonymous certificates can be issued to a party based on its real certificate.

The reasons for allowing one party to have multiple anonymous certificates are two-fold. First, different certificates of a party can be used for different sessions of document exchange. This prevents a malicious party from linking these sessions of exchange to the same party, as these certificates appear to have no connection. It is important in anonymous applications to avoid such associations. For example, when a party uses the same anonymous certificate for a number of sessions of document exchange with a vendor, the vendor can assemble the party's profile such as personal liking, based on those documents exchanged. If the vendor manages to associate the certificate to a real certificate of the party, then the party's profile can be linked to the party's real identity.

Secondly, anonymous certificates of a party can be issued sequentially (Fig. 1), to complicate association of an anonymous certificate with a real one. In Fig. 1, $C_{i,c0}$ denotes a real certificate of a party P_i, which is issued by a CA Ac_0 according to P_i's request, and $C_{i,cj}$ $(1 \leq j \leq n)$ is an anonymous certificate of P_i, which is issued by A_{cj} upon P_i's request using certificate $C_{i,cj-1}$. Note that some of these CAs may be identical. If we only allow the link from $C_{i,cj-1}$ to $C_{i,cj}$ to be established jointly by A_{cj-1} and A_{cj}, then an adversary has to compromise all the CAs in the sequence in order to link $C_{i,cn}$ to $C_{i,c0}$ This is difficult to accomplish in practice.

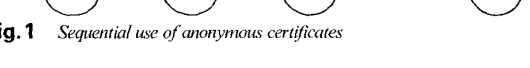

Fig. 1 *Sequential use of anonymous certificates*

The basic idea of the sequential certificate issuing shown in Fig. 1 is analogous to that of untraceable authentication for mobile users [3]: a mobile user can request a recently visited domain to ensure their solvency to the domain which the user is currently visiting. This prevents an unauthorised party or a visited domain authority from tracing the mobile user's home domain and identity. However, the use of anonymous certificates described in this paper has the following considerations.

- Control over the number of anonymous certificates issued: an unrestricted number of anonymous certificates for a party may impose heavy communication and processing loads on CAs, which could lead to degradation of their performance and problems with certificate management. Thus the number of anonymous certificates issued to a party should be limited.
- Certificate expiration times: to simplify certificate management and to increase certificate assurance, when a certificate of a party expires, it would be desirable for all the certificates, issued directly or indirectly using the expired certificate (i.e. all its down-stream certificates), to expire as well. If the same expiration time is used for different certificates of a party, then the time could be utilised by an adversary to link these certificates to the same party. Hence certain measures should be taken for determination of expiration times.
- Real identity tracing: when a dispute occurs about a message signed with the private key associated with an anonymous certificate of a party, a legal authority can be called to trace the anonymous certificate back to its corresponding real certificate, and thus hold the party accountable for the message. To fulfil such tracing, the legal authority must possess undeniable and verifiable evidence to prove the link from the anonymous certificate to the real one.

In the subsequent sections, we address these issues in detail, and formally present the protocols for anonymous certificate issuing and real identity tracing.

3 Notation

The notation to be used throughout this paper is summarised as follows.

- $(pk_{i,a}, sk_{i,a})$ is a pair of public and private keys for a party P_i certified by a CA A_a. (pk_a, sk_a) with a single subscript is a pair of public and private keys owned by a CA A_a.
- $E_k(x)$ expresses the ciphertext of a data item x encrypted with a key k. $E_k(x)$ is computed using a public-key cryptosystem if the corresponding decryption key is different from k, and using a conventional cryptosystem otherwise.
- x, y denotes the concatenation of data items x and y.
- $h(x)$ is a one-way hash function with the following properties: for any x, it is easy to compute $h(x)$; given $h(x)$, it is difficult to compute x; and given x, it is difficult to find x' $(x \neq x')$ such that $h(x) = h(x')$.
- $P_i \rightarrow P_j : m$ expresses that a party P_i sends a message m to another party P_j through a normal communication channel.
- $P_i \rightarrow^A P_j : m$ signifies two cases. In the first case, P_i sends m to P_j through an anonymous communication channel, so that P_i's identity and address are hidden from P_j. This means that P_i knows P_j's address/identity, but P_j does not know P_i's. If P_i requires P_j to respond to its message, P_i needs to send to P_j m together with an anonymous reply address. This is omitted from this notation for simplicity. The details of such anonymous communication are described elsewhere [4, 5].

In the second case, P_i responds to an anonymous message from P_j with m using the anonymous reply address defined in P_j's message. This implies that P_j knows P_i's address/identity, but not conversely. The notation does not distinguish between the above two cases as they can be easily separated based on the context of the protocol presentation to be given.

$C_{i,b}$ is a (real or anonymous) certificate of a party P_i issued by a CA A_b, which is denoted as $C_{i,b} = (A_b, I_{i,b}, pk_{i,b}, e_{i,b}, s_{i,b})$, where $s_{i,b} = E_{sk_b}(h(A_b, I_{i,b}, pk_{i,b}, e_{i,b}))$ is A_b's signature, and $I_{i,b}$, $pk_{i,b}$, and $e_{i,b}$ are a real identity or pseudonym of P_i, public key and certificate expiration time. We also assume that every party knows the public keys of all the CAs.

4 Anonymous certificate issuing

4.1 Certificate issuing protocol

To limit the number of anonymous certificates, we assume that each party is permitted to have a tree-like structure of certificates with one of its real certificates as the root.

Fig. 2 depicts such a certificate tree, where the root node is a real certificate of a party P_i, issued by CA A_o, and the others are anonymous certificates issued directly or indirectly based on the real certificate. For example, $C_{i,o}$ in the Figure is the parent of $C_{i,a}$, issued directly using $C_{i,o}$ by A_a, and the grandparent of $C_{i,b}$, issued indirectly through $C_{i,a}$ by A_b. Clearly, if P_i has m real certificates, then there will be m corresponding certificate trees for P_i.

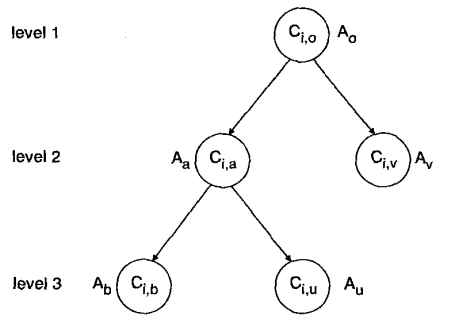

Fig. 2 *Certificate tree*
For A_o: $mc_o = 3$ and $gr_{i,o} = (2, le_{i,o})$
For A_a: $mc_a = 2$ and $gr_{i,a} = (1, le_{i,a})$
For A_o: $ml_o = 3$, $mc_o = 2$ and $gr_{i,o} = (3, le_{i,o})$
For A_v: $mc_v = 0$ and $gr_{i,v} = (2, le_{i,v})$
For A_u: $mc_u = 3$ and $gr_{i,u} = (1, le_{i,u})$

For a certificate tree, we impose limitations on the number of its levels and the number of child certificates for each certificate. To fulfil this, we require that every CA A_j specifies two constants ml_j and mc_j, which remain unchanged to any certificate issued by A_j. ml_j states that for any real certificate issued by A_j, the number of levels for its corresponding certificate tree cannot exceed ml_j. mc_j means that, for any real or anonymous certificate issued by A_j, if it is permitted to have any child certificates, then the number of its child certificates cannot exceed mc_j. A certificate with no child is called a leaf of the tree.

For example, the certificate tree in Fig. 2 meets the limitations with regard to the numbers ml_o and mc_j ($j \in \{o, a, b, u, v\}$) shown. This is because the number of certificate levels is 3 (= ml_o), and the number of child certificates for any certificate $C_{i,j}$ is not greater than mc_j. Note that numbers ml_l ($l \in \{a, b, u, v\}$) are omitted as they have no effect on the tree. Obviously, $C_{i,b}$ and $C_{i,u}$ are not allowed to have any child certificate although $mc_b = 2$ and $mc_u = 3$, as they are on the last level of the tree. $C_{i,o}$ cannot have any more child certificates due to $mc_o = 2$, $C_{i,a}$ can have one more child certificate because $mc_a = 3$, and $C_{i,v}$ cannot have a child due to $mc_v = 0$.

When a party P_i requests a CA A_b to issue a new anonymous certificate $C_{i,b}$, based on an existing one $C_{i,a}$, the expiration time $e_{i,b}$ of $C_{i,b}$ should not be later than that $e_{i,a}$ of $C_{i,a}$, as discussed in Section 2. To issue $C_{i,b}$, A_b asks the CA A_a, which has issued $C_{i,a}$, to authenticate P_i's request, as A_b is not permitted to view $C_{i,a}$. A_a responds to A_b with a guarantee of the authenticity and the latest expiration time $le_{i,b}$ (< $e_{i,a}$) of $C_{i,b}$ without knowing $C_{i,b}$, (as explained later). The main problem with $le_{i,b}$ is that if A_a is compromised and knows that very few certificates will expire before $le_{i,b}$, then A_a could identify $C_{i,b}$ and link it to $C_{i,a}$. Similarly, compromised A_b could guess $C_{i,a}$ if very few certificates expire after $le_{i,b}$.

We can avoid the problem by hiding A_b from A_a, allowing the certificate tree to have a sufficient number of levels, and letting P_i mainly use certificates on the last few levels of the tree. In this way, although an adversary could link $C_{i,b}$ to $C_{i,a}$ if the adversary compromises A_b, it would still be very difficult for the adversary to associate a certificate on one of the last few levels to its corresponding real certificate, as discussed in Section 2.

By combining the certificate tree and expiration time discussed above, we can define a granted right $gr_{i,b} = (st_{i,b}, le_{i,b})$ for a CA A_b to issue an anonymous certificate for a party P_i. This right states that A_b can issue a certificate $C_{i,b}$ for P_i, with an expiration time before $le_{i,b}$, and permits $C_{i,b}$ to have a certificate sub-tree with $C_{i,b}$ as the root and a maximum of $st_{i,b}$ levels. Obviously, if $st_{i,b} = 0$, A_b is not allowed to issue any certificate for P_i. $gr_{i,b}$ is granted by CA A_a which has issued parent certificate $C_{i,a}$, which is in turn granted by the CA that issued its parent certificate, and so on. When $C_{i,a}$ is the real certificate, $gr_{i,a}$ is $(ml_a, le_{i,a})$ defined by A_a. $gr_{i,b}$ and $gr_{i,a}$ should meet the conditions $st_{i,b} = st_{i,a} - 1$ and $le_{i,b} < le_{i,a}$.

Fig. 2 shows an example of such rights where $le_{i,a}$, $le_{i,v} < le_{i,o}$ and $le_{i,b}$, $le_{i,u} < le_{i,a}$. For example, $gr_{i,a}$ states that the expiration time of $C_{i,a}$ must be earlier than $le_{i,a}$ and that the certificate sub-tree with $C_{i,a}$ as the root is allowed to have a maximum of two levels due to $st_{i,a} = 2$. Evidently, $gr_{i,b}$ and $gr_{i,u}$ do not permit $C_{i,b}$ and $C_{i,u}$ to have any child certificates because $st_{i,b} = 1$ and $st_{i,u} = 1$, respectively. $gr_{i,o}$ is defined by A_o with $st_{i,o} = ml_o = 3$. $gr_{i,a}$ is granted to A_a by A_o with $st_{i,a} = st_{i,o} - 1 = 2$, and $gr_{i,b}$ is granted to A_b by A_a with $st_{i,b} = st_{i,a} - 1 = 1$. Similar explanation can be applied to the other certificates.

We can now describe the protocol for issuing an anonymous certificate, which consists of four stages. Suppose that a party P_i requests a CA A_b to issue a new certificate $C_{i,b}$, based on an existing certificate $C_{i,a}$ issued by a different CA A_a. In stage 1, P_i sends A_b its request, which includes identity A_a and an item $x_{i,b}$, verifiable only by A_a, over an anonymous channel without revealing its identity to A_b. In stage 2, A_b simply forwards $x_{i,b}$ together with other relevant items to A_a. In stage 3, A_a verifies $x_{i,b}$ to prove the authenticity of P_i's request, grants A_b a right for issuing a certificate for P_i accordingly, and transfers the right to A_b. In stage 4, A_b issues a new certificate $C_{i,b}$, based on the right granted.

Table 1: Certificate issuing protocol and associated definitions

I1. $P_i \rightarrow^A A_b : E_{pk_b}(A_a, pk_{i,b}, se_{i,b}, sn_{i,b}, x_{i,b})$
I2. $A_b \rightarrow^A A_a : x_{i,b}$
I3. $A_a \rightarrow^A A_b : sn_{i,b}, E_{k_{i,b}}(sr_{i,b})$
I4. $A_b \rightarrow^A P_i : C_{i,b}$

Item	Definition
$sn_{i,b}$, $se_{i,b}$	randomly chosen by P_i
$I_{i,a}$	P_i's identity in $C_{i,a}$ issued by A_a
$pk_{i,b}$	P_i's public key to be certified
$k_{i,b}$	$= h(A_b, pk_{i,b}, se_{i,b})$
$au_{i,b}$	$= E_{sk_{i,a}}(A_a, I_{i,a}, sn_{i,b}, k_{i,b})$
$x_{i,b}$	$= E_{pk_a}(I_{i,a}, au_{i,b})$
$gr_{i,b}$	granted right by A_a
$sr_{i,b}$	$= E_{sk_a}(sn_{i,b}, k_{i,b}, gr_{i,b})$
$C_{i,b}$	P_i's certificate issued by A_b

Table 1 illustrates this certificate issuing protocol, together with the definitions of the items used. The protocol is explained below.

(1) In this stage, P_i first decides a new pair of public and private keys $(pk_{i,b}, sk_{i,b})$, and chooses two random numbers $sn_{i,b}$ and $se_{i,b}$. $sn_{i,b}$ is used as the current session number. $se_{i,b}$ is a secret number used for the calculation of a session

key $k_{i,b}$. P_i then computes $k_{i,b}$, $au_{i,b}$, and $x_{i,b}$ defined in Table 1. Here, the encryption for $x_{i,b}$ with A_a's key pk_a is to allow only A_a to know P_i's identity $I_{i,a}$. The encryption for $au_{i,b}$ with P_i's secret key $sk_{i,a}$, is to allow A_a to authenticate P_i. $x_{i,b}$ enables A_b to acquire from A_a an approval of certifying public key $pk_{i,b}$ related to $k_{i,b}$ without letting A_a know $pk_{i,b}$. If necessary, a time stamp can be added into the message.

(2) In this stage, A_b first decrypts P_i's message with key sk_b and verifies that A_a is a valid CA. If the verification fails, A_b terminates the protocol run. Note that, in the following, a failure of any verification will lead to such protocol termination without being explicitly mentioned. If the verification succeeds, A_b saves all the items and forwards $x_{i,b}$ to A_a.

(3) In this stage, A_a decrypts $x_{i,b}$ with sk_a, and verifies that identity $I_{i,a}$ in $x_{i,b}$ has a certificate $C_{i,a}$, issued and held by A_a, which has neither expired nor been revoked. If the verification result is positive, A_a decrypts $au_{i,b}$ in $x_{i,b}$ with key $pk_{i,a}$ in $C_{i,a}$, to obtain A'_a, $I'_{i,a}$, $sn_{i,b}$ and $k_{i,b}$. If $A'_a = A_a$ and $I'_{i,a} = I_{i,a}$, A_a is convinced that the request originated from P_i.

Based on expiration time $e_{i,a}$ in $C_{i,a}$ and right $gr_{i,a} = (st_{i,a}, le_{i,a})$, A_a checks that

(*a*) $st_{i,a} > 1$, i.e. A_a is permitted to grant child certificates for P_i.

(*b*) the number of unexpired child certificates with regard to their latest expiration times, granted for P_i by A_a, is less than mc_a defined earlier, i.e. A_a can grant another child certificate for P_i.

If both conditions (*a*) and (*b*) hold, A_a chooses a future time $le_{i,b}$ ($< e_{i,a}$) that makes the number of existing certificates, issued by A_a and to expire after $le_{i,b}$, sufficiently large to avoid an adversary guessing $C_{i,a}$ using compromised A_b, as discussed earlier. A_a then assigns A_b the right $gr_{i,b} = (st_{i,a} - 1, le_{i,b})$. If either condition (*a*) or (*b*) is not met, the right $gr_{i,b} = (0, 0)$.

Additionally, A_a computes $sr_{i,b}$ (defined in Table 1), and sends $E_{k_{i,b}}(sr_{i,b})$ to A_b. Here, key $k_{i,b}$ is used for the encryption $E_{k_{i,b}}(sr_{i,b})$, because A_a does not know the identity of A_b.

Note that A_a needs to store $sn_{i,b}$, $k_{i,b}$, $le_{i,b}$ and $au_{i,b}$ when $st_{i,a} - 1 > 0$. These are needed for identity tracing (presented later).

(4) In this stage, A_b decrypts $E_{k_{i,b}}(sr_{i,b})$ with $k_{i,b}$ and then $sr_{i,b}$ with pk_a. A_b examines that $sn_{i,b}$ and $k_{i,b}$ in $sr_{i,b}$ are correct, and that $st_{i,b}$ in $gr_{i,b}$ is greater than 0 ($st_{i,b} > 0$). If the result is positive, A_b generates a unique pseudonym $I_{i,b}$, chooses an expiration time $e_{i,b}$($< le_{i,b}$), and issues the following certificate:

$$s_{i,b} = E_{sk_b}(h(A_b, I_{i,b}, pk_{i,b}, e_{i,b}))$$

$$C_{i,b} = (A_b, I_{i,b}, pk_{i,b}, e_{i,b}, s_{i,b})$$

If A_b is not permitted to issue a certificate for P_i (i.e. $st_{i,b} = 0$), A_b executes the following transaction:

$$A_b \rightarrow^A P_i : sn_{i,b}, E_{sk_b}(sn_{i,b}, \text{'no certification'})$$

For intuitiveness, Table 2 lists the items stored for a certificate $C_{i,b}$ by its corresponding CA A_b.

Table 2: Items stored for certificate $C_{i,b}$ by CA A_b

Certificate	Stored items
$C_{i,b}$	$sn_{i,b}$, $C_{i,b}$, A_a, $se_{i,b}$, $sr_{i,b}$
Child 1 - $C_{i,c1}$	$sn_{i,c1}$, $k_{i,c1}$, $le_{i,c1}$, $au_{i,c1}$
Child 2 - $C_{i,c2}$	$sn_{i,c2}$, $k_{i,c2}$, $le_{i,c2}$, $au_{i,c2}$
...	...

4.2 Protocol analysis

We now analyse the protocol shown in Table 1 with regard to anonymity and accountability. The anonymity of the protocol is argued based on the following cases.

- Compromised A_b is unable to link $C_{i,b}$ to $C_{i,a}$ when the number of existing certificates, issued by A_a and to expire after $le_{i,b}$, is sufficiently large. This is due to the following reasons. First, the sufficiently large number means that the probability for A_b to guess $C_{i,a}$ among these certificates is very low. Secondly, P_i does not sign the message sent to A_b, and the communication between P_i and A_b is conducted over anonymous channels, and so A_b does not know the address and identity of P_i. Finally, $gr_{i,b}$ granted by A_a does not involve any information on $C_{i,a}$. When the number is small, A_b might be able to link $C_{i,b}$ to $C_{i,a}$, which can be handled by the solution suggested in Section 4.1. A similar discussion can also be applied to the case of linking $C_{i,b}$ to $C_{i,a}$, by compromised A_a.
- A_a and A_b are able to associate $C_{i,b}$ to $C_{i,a}$ by collusion. This problem can be alleviated when the certificate tree of P_i has a sufficient number of levels, as discussed in Section 2.

The accountability of the protocol is demonstrated by the following cases:

- A_b cannot dishonestly issue $C_{i,b}$ under P_i's name for another party P_j ($\neq P_i$). The reasons for this are that $au_{i,b}$ in $x_{i,b}$ is signed by P_i's private key $sk_{i,a}$; $sr_{i,b}$ is A_a's signature on granted right $gr_{i,b}$; $k_{i,b}$ in both $au_{i,b}$ and $sr_{i,b}$ prevents A_b from changing key $pk_{i,b}$ in $C_{i,b}$; and neither A_b nor P_j knows private key $sk_{i,b}$ associated with $pk_{i,b}$. Similarly, we can show that even by collusion with A_a, A_b is still unable to fulfil such dishonest issuing, as A_a and A_b cannot forge $au_{i,b}$ and $k_{i,b}$. This also implies that P_i cannot falsely deny its request for $C_{i,b}$, and that A_a cannot falsely deny granting $gr_{i,b}$ for the issuing of $C_{i,b}$.
- $C_{i,b}$ can be traced back to its corresponding real certificate only by a legal authority with verifiable and non-repudiatable evidence (described in Section 5).
- Where another party P_j has compromised P_i's private key $sk_{i,a}$ associated with $C_{i,a}$, P_j may be able to acquire new certificates using $C_{i,a}$, and to abuse them, with P_i being accused for P_j's misbehaviour. This problem can be avoided by allowing P_i to have m (> 1) real certificates and to request a new anonymous certificate using at least l ($1 < l \leq m$) existing certificates. In this case, P_j has to compromise l private keys of P_i in order to obtain a new certificate linked to P_i, which is harder to fulfil. This implies that P_i has a graph-like structure of certificates. The protocol given in Section 4.1 can be extended to deal with such a certificate structure (to be addressed in future work).

5 Real identity tracing

5.1 Identity tracing protocol

When a dispute about a message signed with the private key associated with an anonymous certificate of a party P_i occurs, a legal authority LA can be called to trace the anonymous certificate back to its corresponding real certificate, thereby holding P_i accountable for the message. Here we assume that LA is a trusted authority and does not disclose the link between an anonymous certificate and its corresponding real certificate to any unauthorised party.

To trace an anonymous certificate $C_{i,b}$ of P_i back to its corresponding real certificate $C_{i,o}$ containing the real identity of P_i, a process can be devised to produce the following certificate sequence from $C_{i,b}$ to $C_{i,o}$ in a bottom-up way:

$$C_{i,b}, C_{i,a}, C_{i,p}, ..., C_{i,j}, C_{i,l}, C_{i,o}$$

This means that, A_b passes the non-repudiatable evidence about the granting and issuing of $C_{i,b}$ to A_a which issued $C_{i,a}$. A_a then sends its evidence about $C_{i,a}$, together with the evidence received from A_b, to A_p which issued $C_{i,p}$. This process continues until A_o associated with $C_{i,o}$ has received the evidence from A_l which issued $C_{i,l}$. A_o finally transfers its evidence along with that received to LA. This enables LA to obtain all the evidence about the issuing and granting of the certificates in the sequence, and to assess the authenticity of the link between $C_{i,b}$ and $C_{i,o}$ based on the evidence. Note that the evidence is verifiable only by LA.

Table 3 shows such a protocol for real identity tracing, along with the definitions of the new items used.

Table 3: Protocol for real identity tracing and associated definitions

T1. $LA \rightarrow A_b : E_{pk_b}(C_{LA}, C_{i,b}, s_{LA}, si_{LA})$

T2. $A_b \rightarrow^A A_a : E_{pk_a}(C_{LA}, s_{LA}, sn_{i,b}, ev_b, ch_b)$

...

T2'. $A_l \rightarrow^A A_o : E_{pk_o}(C_{LA}, s_{LA}, sn_{i,l}, ev_l, ch_l)$

T3. $A_o \rightarrow LA : ev_o$

Item	Definition
n_{LA}	session number chosen by LA
t_{LA}	time stamp
s_{LA}	$= E_{sk_{LA}}(n_{LA}, t_{LA})$
si_{LA}	$= E_{sk_{LA}}(h(C_{i,b}, s_{LA}))$
ev_b	$= E_{pk_{LA}}(C_{i,b}, E_{sk_b}(n_{LA}, se_{i,b}, sr_{i,b}))$
ch_b	$= h(k_{i,b}, C_{LA}, s_{LA}, sn_{i,b}, ev_b)$
ev_l	$= E_{pk_{LA}}(C_{i,l}, ev_j, E_{sk_l}(n_{LA}, se_{i,l}, sr_{i,l}, au_{i,j}, h(ev_j)))$
ch_l	$= h(k_{i,l}, C_{LA}, sn_{i,l}, ev_l)$
ev_o	$= E_{pk_{LA}}(C_{i,o}, ev_l, E_{sk_o}(n_{LA}, au_{i,l}, h(ev_l)))$

The protocol is explained in detail below:

(1) In transaction $T1$, C_{LA} is LA's public-key certificate, and $C_{i,b}$ is the certificate to be traced. s_{LA} is LA's signature on the current session number n_{LA}, which will be passed to all relevant CAs. si_{LA} is LA's signature on $C_{i,b}$ and s_{LA}.

(2) Having received the message from LA, A_b decrypts it with key sk_b, and confirms that C_{LA} indeed represents a legal authority. A_b then verifies the correctness of s_{LA} and si_{LA}. Additionally, A_b checks that $C_{i,b}$ is a certificate issued by A_b.

Once the above verification process is successfully completed, A_b locates items $sn_{i,b}$, A_a, $se_{i,b}$ and $sr_{i,b}$ stored for $C_{i,b}$ (as illustrated in Table 2), derives key $k_{i,b} = h(A_b, pk_{i,b}, se_{i,b})$, and computes ev_b and ch_b, (defined in Table 3) to form its message in $T2$. Here, ev_b provides LA with the evidence that A_b issued $C_{i,b}$ upon P_i's request (detailed later) and that ch_b is a message checksum.

Upon the receipt of the message from A_b through $T2$, A_a decrypts it with key sk_a, and uses $sn_{i,b}$ to find its related certificate $C_{i,a}$ and stored items $k_{i,b}$ and $au_{i,b}$. A_a then examines the correctness of C_{LA}, s_{LA} and ch_b. If correct, A_a locates another set of items $sn_{i,a}$, A_p, $se_{i,a}$ and $sr_{i,a}$ saved for $C_{i,a}$. Similar to the operations performed by A_b, A_a computes

$$k_{i,a} = h(A_a, pk_{i,a}, se_{i,a})$$
$$ev_a = E_{pk_{LA}}(C_{i,a}, ev_b, E_{sk_a}(n_{LA}, se_{i,a}, sr_{i,a}, au_{i,b}, h(ev_b)))$$
$$ch_a = h(k_{i,a}, C_{LA}, s_{LA}, sn_{i,a}, ev_a)$$

to form its message. Note that ev_a includes two extra items $au_{i,b}$ and ev_b. $au_{i,b}$ allows LA to verify the authenticity of P_i's request for $C_{i,b}$, and ev_b is the evidence received from A_b. A_a finally executes a transaction similar to $T2$ to send its message to A_p, i.e.

$$A_a \rightarrow^A A_p : E_{pk_p}(C_{LA}, s_{LA}, sn_{i,a}, ev_a, ch_a)$$

Similarly, A_p repeats the above operations described for A_a. This continues until A_l has executed $T2'$.

(3) ev_o in $T3$ includes ev_l received from A_l and $au_{i,l}$ stored by A_o for $C_{i,l}$ (as defined in Table 3). Note that ev_o does not contain items $se_{i,o}$ and $sr_{i,o}$ as $C_{i,a}$ is the real certificate.

Having received ev_o from A_o, LA decrypts it with key sk_{LA} and then $E_{sk_o}(n_{LA}, au_{i,l}, h(ev_l))$ with key pk_o, to recover $C_{i,o}$, ev_l, n_{LA}, $au_{i,l}$, and $h(ev'_l)$. Here we assume that LA knows the public keys of all the CAs. LA then verifies that n_{LA} is correct and $h(ev_l)$ is equal to $h(ev'_l)$ (i.e. $h(ev_l) = h(ev'_l)$). If the verification is positive, LA decrypts ev_l with sk_{LA} and then $E_{sk_l}(n_{LA}, se_{i,l}, sr_{i,l}, au_{i,j}, h(ev_j))$ with pk_l, to obtain $C_{i,l}$, ev_j, n_{LA}, $se_{i,l}$, $sr_{i,l}$, $au_{i,j}$ and $h(ev_j)$, where LA determines pk_l based on A_l's identity in $C_{i,l}$. In addition to repeating the verification above, LA carries out the following verifications.

(i) Computes $k_{i,l} = h(A_l, pk_{i,l}, se_{i,l})$, and decrypts $au_{i,l}$ with $pk_{i,o}$ in $C_{i,o}$ to obtain A_o, $I_{i,o}$, $sn_{i,l}$ and $k'_{i,l}$ (as defined in Table 1). If A_o and $I_{i,o}$ are both in $C_{i,o}$, and $k'_{i,l} = k_{i,l}$, then LA believes that P_i did apply to A_l for $C_{i,l}$ using $C_{i,o}$, and that $pk_{i,l}$ in $C_{i,l}$ is correct as no other party could have forged $au_{i,l}$, $se_{i,l}$ and $k_{i,l}$.

(ii) Decrypts $sr_{i,l}$ with pk_o to get $sn_{i,l}$, $k''_{i,l}$, and $gr_{i,l} = (st_{i,l}, le_{i,l})$. If $k''_{i,l} = k_{i,l}$, $st_{i,l} > 0$, and $e_{i,l} < le_{i,l}$, LA is convinced that A_o did grant A_l the right for issuing $C_{i,l}$, as $sr_{i,l}$ is signed with A_o's private key sk_o.

LA continues with the same verifications for each evidence ev_j until ev_b has been verified.

We use an example to demonstrate the operations of the above protocol. Suppose that LA wants to trace $C_{i,b}$ in Fig. 2 back to $C_{i,o}$. The transactions performed by the protocol are listed as follows:

$$(a) \quad LA \rightarrow A_b : E_{pk_b}(C_{LA}, C_{i,b}, s_{LA}, si_{LA})$$
$$(b) \quad A_b \rightarrow^A A_a : E_{pk_a}(C_{LA}, s_{LA}, sn_{i,b}, ev_b, ch_b)$$
$$(c) \quad A_a \rightarrow^A A_o : E_{pk_o}(C_{LA}, s_{LA}, sn_{i,a}, ev_a, ch_a)$$
$$(d) \quad A_o \rightarrow LA : ev_o$$

Our main concern here is to show what each evidence consists of; and so only the details of items ev_b ev_a and ev_o are given below:

$$ev_b = E_{pk_{LA}}(C_{i,b}, E_{sk_b}(n_{LA}, se_{i,b}, sr_{i,b}))$$
$$ev_a = E_{pk_{LA}}(C_{i,a}, ev_b, E_{sk_a}(n_{LA}, se_{i,a}, sr_{i,a}, au_{i,b}, h(ev_b)))$$
$$ev_o = E_{pk_{LA}}(C_{i,o}, ev_a, E_{sk_o}(n_{LA}, au_{i,a}, h(ev_a)))$$

5.2 Protocol analysis

We now examine the anonymity and accountability of the protocol presented in Table 3. The protocol's anonymity is obvious because information on the certificate sequence is encrypted with LA's public key, i.e. only LA is able to link together the certificates in the sequence.

The accountability is demonstrated by the following cases:

• The certificate sequence from $C_{i,b}$ to $C_{i,o}$ is undeniable. As described in Section 5.1, for each certificate $C_{i,b}$ in the

sequence, LA can prove that P_i did request $C_{i,b}$ from A_b using its parent certificate $C_{i,a}$; that A_a which issued $C_{i,a}$ indeed granted A_b a right for the issuing of $C_{i,b}$; and that A_b did issue $C_{i,b}$, in which the public key and expiration time are correct with respect to the granted right.

• A compromised CA could disrupt the tracing process by not executing transaction $T2$. If such a case happens, LA can perform the protocol stepwise in such a way that LA sends its request to A_b in transaction $T1$. Instead of transferring evidence ev_b to A_a, A_b is required to send ev_b, $sn_{i,b}$ and A_a directly to LA in $T2$. LA repeats the same operations for every such CA A_a. In this way, LA is able to spot the compromised CA when it refuses to respond to LA's request, or provides incorrect evidence.

6 Conclusions

We have presented two protocols for anonymous certificate issuing and real identity tracing. The anonymity analysis of the protocols has demonstrated that it is very difficult for an adversary to link an anonymous certificate to its corresponding real certificate, even under the circumstances where the adversary is capable of compromising some of the CAs. The accountability analysis of the protocols has indicated that no CA could issue an anonymous certificate without a request from the certificate holder, and that a legal authority is able to collect verifiable and non-repudiatable evidence to prove the link from an anonymous certificate to its associated real certificate.

Different anonymous certificates, coupled with anonymous communication, enable a party to operate existing fair exchange protocols [6–12] for different sessions of document exchange with other parties. Not only can this hide the party's identity and location, while retaining its messages accountable, but it also prevents these sessions from being linked to the party unlawfully. Thus, the fair exchange protocols can be easily enhanced to achieve anonymity.

7 Acknowledgment

The authors would like to thank John Haggerty for his useful comments.

8 References

1 STALLINGS, W.: 'Network and internetwork security: principles and practice' (Prentice Hall, 1995)
2 OISHI, K., MAMBO, M., and OKAMOTO, E.: 'Anonymous public key certificates and their applications', *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, 1998, **E81-A**, (1), pp. 56–64
3 SAMFAT, D., MOLVA, R., and ASOKAN, N.: 'Untraceability in mobile networks'. Proceedings of first annual international conference on *Mobile computing and networking*, New York, USA, 1995, pp. 26–36
4 CHAUM, D.: 'Untraceable electronic mail, return addresses, and digital pseudonyms', *Commun. ACM*, 1981, **24**, (2), pp. 84–88
5 GULCU, C., and TSUDIK, G.: 'Mixing e-mail with babel'. Proceedings of 1996 symposium on *Network and distributed system security*, Los Alamitos, California, USA, 1996, pp. 2–16
6 BAO, F., DENG, R., and MAO, W.: 'Efficient and practical fair exchange protocols with off-line TTP'. Proceedings of IEEE symposium on *Security and privacy*, Oakland, California, USA, May 1998, pp. 77–85
7 BAO, F., and DENG, R.: 'An efficient fair exchange protocol with an off-line semi-trusted third party'. Proceedings of workshop on *Cryptographic techniques and e-commerce*, City University, Hong Kong, 1999, pp. 37–47
8 BAO, F., DENG, R., NGUYEN, K.Q., and VARADHARAJAN, V.: 'Multi-party fair exchange with an off-line trusted neutral party'. Proceedings of tenth international workshop on *Database and expert systems applications*, Los Alamitos, California, USA, 1999, pp. 858–862
9 FRANKLIN, M., and REITER, M.: 'Fair exchange with a semi-trusted third party'. Proceedings of fourth ACM conference on *Computer and communications security*, Zurich, Switzerland, April 1997, pp. 1–5
10 ASOKAN, N., SHOUP, V., and WAIDNER, M.: 'Asynchronous protocols for optimistic fair exchange'. Proceedings of IEEE symposium on *Security and privacy*, Oakland, California, USA, May 1998, pp. 86–100
11 ZHANG, N., and SHI, Q.: 'Achieving non-repudiation of receipt', *Comput. J.*, 1996, **39**, (10), pp. 844–853
12 ZHANG, N., SHI, Q., and MERABTI, M.: 'A flexible approach to secure and fair document exchange', *Comput. J.*, 1999, **42**, (7), pp. 569–581