ELSEVIER

Computer Communications 22 (1999) 710–716

J. computer communications

Securing digital signatures for non-repudiation

J. Zhou[a,*], K.Y. Lam[b]

[a]*Kent Ridge Digital Labs, 21 Heng Mui Keng Terrace, Singapore 119613, Singapore*
[b]*School of Computing, National University of Singapore, 10 Kent Ridge Crescent, Singapore 119260, Singapore*

Received 29 June 1998; received in revised form 22 December 1998; accepted 22 December 1998

Abstract

Dispute of transactions is a common problem that could jeopardise business. Hence non-repudiation services are essential in business transactions which provide evidence to enable dispute resolution. To be eligible as non-repudiation evidence, the digital signature on an electronic document should remain valid until its expiry date which is specified by some non-repudiation policy. The conventional approaches are either inefficient or insecure to achieve non-repudiation in electronic commerce. This article presents a practical scheme to secure digital signatures as non-repudiation evidence with an adjustable degree of risk. © 1999 Elsevier Science B.V. All rights reserved.

Keywords: Digital signature; Non-repudiation; Secure electronic commerce

1. Introduction

Dispute of transactions is a common problem that could jeopardise business. Typical disputes that may arise in a simple transaction such as transferring a message *M* (e.g. electronic cash or electronic contracts) from Alice to Bob could be

- Alice claims that she has sent *M* to Bob while Bob denies having received it;
- Bob claims that he received *M* from Alice while Alice denies sending it.

In order to settle these disputes by a third party arbitrator, e.g. a judge, Alice and Bob need to present evidence to prove their own claims. Such evidence may be provided by non-repudiation services [1–3].

Non-repudiation services protect the transacting parties against any false denial that a particular event or action has taken place. Some international standards on non-repudiation have been established [4–7]. The basic non-repudiation services that address the aforementioned disputes are:

- *Non-repudiation of Origin (NRO)* provides the recipient of a message with evidence of origin (EOO) of the message which will protect against any attempt by the originator to falsely deny having sent the message.
- *Non-repudiation of Receipt (NRR)* provides the originator of a message with evidence of receipt (EOR) of the message which will protect against any attempt by the recipient to falsely deny having received the message.

Digital signature is an important security mechanism for generating non-repudiation evidence and is receiving legal recognition [8].

2. Security requirements on digital signatures

Digital signatures are widely used in security services such as authentication and non-repudiation [7,9]. They can be used to authenticate the origin and integrity of a message. Moreover, they can be used as non-repudiation evidence to establish accountability. The security of digital signatures relies not only on the cryptographic strength of signature algorithms, but also on the management of signature keys.

We should be aware of the varying security requirements which need to be satisfied when digital signatures are used for different purposes. For example, in authentication services, the signature verifier only needs to check whether a signature is valid at the time of verification, and does not care about its validity afterwards. In non-repudiation services, however, the signature that was verified should remain valid until its expiry date, which is specified by some non-repudiation policy, because disputes may arise after the signature is accepted as non-repudiation evidence.

In practice, a signature key may be compromised and a signature could be forged by using a compromised key.

* Corresponding author.
E-mail addresses: jyzhou@krdl.org.sg (J. Zhou); lamky@comp.nus. edu.sg (K.Y. Lam)

PII: S0140-3664(99)00031-6

Therefore, the compromised key needs to be revoked, so that all signatures generated after the corresponding public key certificate was revoked will be regarded as invalid. In authentication services, the signature verifier can first check the Certificate Revocation List (CRL) to see whether the public key certificate is valid, and only uses the valid key to check the signature. Although the public key certificate may be revoked afterwards, this has no effect on authentication services. In non-repudiation services, however, further evidence is required to prove that the signature was generated before the public key certificate was revoked and thus the signature is valid. Otherwise, the signer may deliberately compromise his signature key and ask for revocation of the corresponding public key certificate before a dispute arises so that the signer can repudiate the validity of his signature by falsely claiming the signature as forged by somebody else after the key had been revoked.

A typical approach to secure digital signatures as non-repudiation evidence relies on the existence of an *on-line* trusted time-stamping authority (TS) [10–14]. Each newly generated digital signature has to be time-stamped by a TS so that the trusted time of signature generation can be identified. Such an approach may be employed in high value business transactions where security is the most important requirement. However, it is likely to be much too expensive to support non-repudiation for large volume of low risk business transactions in electronic commerce networks. Hence a cost-effective solution is needed.

This article presents a new approach to maintain the validity of digital signatures by delegating the signing authority from a long-term revocable signature key to a temporary irrevocable one. Signatures generated with a temporary key will remain valid within a specific period. Thus users can be relieved from invoking the TS in on-line transactions while they are liable for the compromise of temporary signature keys. As digital signatures generated in such a way have a limited valid period, all disputes that require these signatures as non-repudiation evidence should be settled within this period. The new approach will significantly improve the efficiency of mass on-line transactions in electronic commerce with an adjustable level of security.

The rest of the article is organised as follows. In Section 3, we examine the roles of time stamps in digital signatures from non-repudiation perspective. In Section 4, we discuss possible approaches to maintaining the validity of digital signatures. We propose a new scheme in Section 5, and analyse the scheme in Section 6. Section 7 concludes the article.

We use the following notation to represent messages and protocols in this article:

- X, Y: concatenation of two messages X and Y.
- $H(X)$: a one-way hash function applied to message X.
- V_A and S_A: the public and private key or principal A.
- $sK(X)$: digital signature of message X with the private key K. The algorithm is assumed to be a 'signature with appendix', and the message is not recoverable from the signature.
- $A \rightarrow B$: X: principal A dispatches message X addressed to principal B.

3. Time stamps in digital signatures

Time stamps are found in many electronic transactions to indicate the time that a particular event or action took place, e.g. the time that a message was sent or received, the time that a digital signature was generated, or the time that a signature key was revoked. These time stamps may be used to convince other parties involved in a transaction of the validity of an event or action, or used to prove to a third party the truth of an event or action. Users should take care to identify precisely the role of time stamps in a given application.

3.1. Ordinary time stamps

Time stamps can be used in authentication protocols to indicate the freshness of a message [15–17]. The sender who aims to convince the receiver of the freshness of the message is responsible for ensuring the accuracy of the clock value. If the receiver does not believe that the message is fresh, it can simply abort the protocol run. No judgement is required from a third party in authentication protocols.

The situation is different when time stamps are used for non-repudiation purposes. An ordinary time stamp T, which is bound to a message M by the sender A's digital signature in the form of $sS_A(M, T)$, may not serve as a proper part of non-repudiation evidence. To illustrate this, we examine two possible roles that T may play in non-repudiation services.

1. *Identifying the time that A's signature was generated.* This is an important element to check the validity of A's digital signature. As A's signature key S_A may be compromised, S_A should be allowed to be revoked, but A's signatures generated before S_A was revoked should remain valid. Therefore, a time stamp is required to identify the time that A's signature was generated. However, a time stamp included in A's signature by A itself cannot be used for this purpose. If S_A is compromised, there is no way to prove whether A's signature is valid as anybody with the compromised S_A can forge A's signature which could be stamped with a time prior to the compromise time of S_A.
2. *Identifying the time that the message M was sent.* This is an important element to resolve the dispute of late submission. For example, there is usually a deadline for the payment of a bill and a surcharge may be imposed after the deadline. A time stamp identifying the time of sending the payment is critical to resolve a possible dispute over late payment. Again, a time stamp included in A's signature by A itself cannot be used for this

purpose. Although A cannot repudiate what it has signed, A can fake a back-dated time stamp T, and B may not believe that A sent the message M at the time of T.

The non-repudiation evidence defined in ISO/IEC 13888-3 [7] does include ordinary time stamps. For example, a simplified form of EOO in ISO/IEC 13888-3 is

$$\mathrm{EOO} = A, B, T_g, T_1, M, sS_A(A, B, T_g, T_1, M),$$

where A and B are the identifiers of the originator and the recipient, respectively, T_g is the date and time *EOO* was generated, T_1 is the date and time the message M was sent. From the above analysis we find that the ordinary time stamps T_g and T_1 generated by A cannot be used as proper evidence to prove what they claim to be unless additional security mechanisms are imposed on EOO.

3.2. Trusted time stamps

A trusted time stamp is generated by a TS and bound to a message being time-stamped by TS's signature. For example, the sender A can send its signature $sS_A(M)$ to TS to be time-stamped, which may have the form of $sS_{TS}(sS_A(M), T)$. Such a time stamp can be used as non-repudiation evidence to identify the time that $sS_A(M)$ was generated. Thus, $sS_A(M)$ remains valid even if S_A was revoked after T. A detailed time-stamping service is discussed in [18].

4. Approaches for securing digital signatures

To secure digital signatures as non-repudiation evidence, the management of signature keys becomes the key issue provided that the signing scheme is secure (e.g. ISO/IEC 9796 [19] and DSS [20]), and different approaches may be adopted by trusted third parties and ordinary users.

4.1. Securing trusted third party's signatures

Trusted third parties play important roles in the provision of security services, especially in non-repudiation services [2]. They may be required to issue certificates, notarise documents, or provide other supporting evidence. These functions are usually provided by digital signatures. To secure a trusted third party's signatures, its signature key should be well protected. The compromise of such a key will cause serious consequences as there is no effective way to revoke it. For example, if a notary's signature key is compromised, anybody having the compromised key can forge a notarised document, and it is impossible to distinguish whether a document is really notarised by the notary unless the notary stores all notarised documents safely.

A possible solution is the use of *multiple trusted third parties* so that their multi-signatures remain secure and valid as long as one of their signature keys is not compromised. For example, if there are two notaries N_1 and N_2, they can notarise a message M as $sS_{N_2}(sS_{N_1}(M))$. Once S_{N_1} is compromised, S_{N_2} can be destroyed immediately to avoid its compromise. Then N_1 and N_2 can use their new signature keys S'_{N_1} and S'_{N_2} to provide notarisation services. The disadvantages of this approach is the high expense of signature generation and verification. Proactive threshold signature schemes [21] allow multiple trusted third parties to sign a message using a *single* signature key while the secret shares held by each signer for constructing the signature key can be refreshed periodically in a way that the signature key (and its corresponding verification key) is kept unchanged.

Another possible solution is the use of a *tamper-resistant device* to protect a trusted third party's signature key. The signature key can be stored in a tamper-resistant device so that it will not be released out of the device. Then the problem becomes the protection of the device against abuse, which relies on the physical security of the trusted third party. In the following discussion, we assume that trusted third parties' signature keys are well protected and their signatures cannot be forged.

4.2. Securing user's signatures

The simplest approach to securing a user's signatures relies on the existence of an *on-line* TS as well as the certificate revocation infrastructure. For example, user A's signature on a message M should be sent to a trusted TS to certify that the signature was generated at the time of T_g:

1. $A \rightarrow \mathrm{TS}$: $sS_A(M)$
2. $\mathrm{TS} \rightarrow A$: $T_g, sS_{TS}(sS_A(M), T_g)$

Before accepting this, A should check the validity of TS's signature. Such a time-stamping process is included explicitly in Coffey and Saidha's non-repudiation protocol [22], but omitted in other non-repudiation protocols [23–25].

To verify A's signature $sS_A(M)$, the verifier needs to make the following checks:

1. The verifier should check TS's signature on $(sS_A(M), T_g)$.
2. The verifier should check the expiry date T_e of A's public key certificate C_A. If $T_e < T_g$, A's signature is invalid.
3. The verifier should check the CRL. If there exists a record showing that C_A was revoked at the time of T_r and $T_r < T_g$, A's signature is invalid.
4. The verifier should use A's public key certified in C_A to check A's signature $sS_A(M)$.

Only if all of the above checks are successful, will A's signature be regarded as valid non-repudiation evidence. This approach is secure against disputes caused by (accidental or deliberate) compromise of signature keys, but is not efficient in on-line business transactions.

A less secure approach happening on the ground is to combine the use of digital signatures with auditing processes, whereby the audit trails of transaction networks and of parties to the clearing process are used to establish whether a signature was generated at a specific time [26]. That is, a bank may claim that, despite the subsequent revocation of a signature key by an individual, the audit logs

show that the signature was generated at a time prior to the revocation, and hence the signature should be accepted as evidence. In a way, physical security of the audit trail is replacing the time-stamping service. Ultimately the success of this approach depends on how well adjudicators trust the integrity of the bank's auditing systems.

5. A new scheme

In a low risk business transaction where digital signatures are used as non-repudiation evidence to settle disputes only within a certain period, the efficiency of the system can be significantly improved if digital signatures will remain valid within that period without being time-stamped.

The main idea of our approach for securing users' signatures is to define two different types of signature keys:

- *Revocable signature keys* — the corresponding verification key certificates are issued by a certification authority (CA), and can be revoked as usual;
- *Irrevocable signature keys* — the corresponding verification key certificates are issued by users themselves and time-stamped by a TS.[1] Such certificates cannot be revoked before their expiry.

The revocable signature key is used as a long-term master key to issue irrevocable verification key certificates while the irrevocable signature key is used as a temporary key to sign electronic documents. The digital signatures generated in such a way will remain valid until the corresponding irrevocable verification key certificates expire, thus can be exempted from being time-stamped by a TS during on-line transactions.

5.1. Certificate generation

Suppose S_A and V_A are user A's *revocable* signature and verification key pair, and C_A is A's *revocable* verification key certificate with expiry date T_e which is issued by a CA in the form of[2]

$$C_A = A, V_A, T_e, sS_{CA}(A, V_A, T_e)$$

Suppose S'_A and V'_A are user A's *irrevocable* signature and verification key pair. TS is *off-line*. User A can generate its *irrevocable* verification key certificate as below:[3]

1. $A \rightarrow$ TS : $sS_A(V'_A, T'_e)$
2. TS $\rightarrow A$: $T'_g, sS_{TS}(sS_A(V'_A, T'_e), T'_g)$

Thus, the irrevocable verification key certificate C'_A can be defined as

$$C'_A = V'_A, T'_e, T'_g, sS_A(V'_A, T'_e), sS_{TS}(sS_A(V'_A, T'_e), T'_g),$$

where T'_e is the expiry date of C'_A, and T'_g is the time that C'_A was generated. C'_A is valid only if S_A is valid at T'_g, and will remain valid until T'_e even if S_A becomes invalid after T'_g. It is important to note that the validity of S_A can only be checked *before* the revocable certificate C_A's expiry date T_e because the revocation information of expired certificates is not maintained in the CRL. Hence C'_A's expiry date T'_e should not be later than T_e, i.e. $T'_e \leq T_e$.[4]

The aforementioned process of certificate generation by the use of a TS is different from the one using a CA. The TS need not check who is sending what to be time-stamped while CA has to authenticate who is sending the request for a public key certificate. Obviously, it is inefficient and undesirable to change a user's signature key frequently. In this new scheme, S_A will be used as user A's long-term master key for generating C'_A, and only needs to be changed after its expiry or revocation. S'_A will be used as user A's temporary key to sign messages, and can be changed periodically.

As C'_A does not contain any explicit reference to the name of A, another party B may ask CA to issue a revocable verification key certificate C_B in which $V_B = V_A$. Then the origin of a message signed with S'_A could be vague as a signature verifier may regard B as the originator of C'_A if C_B is used in the verification of C'_A. This concern will be much reduced if CA further verifies that B knows the private key S_B corresponding to V_B before CA signs C_B [1,13,26].

5.2. Signature generation and verification

With an irrevocable verification key certificate C'_A, user A can generate its signature on a message M using the corresponding irrevocable signature key S'_A. To form complete non-repudiation evidence, the certificate C'_A should be appended to the signature. Thus, A's signature on message M can be represented as

$$C'_A, sS'_A(M).$$

The expiry date of such a signature is defined the same as C'_A's expiry date T'_e.

To verify the above signature, the verifier should first check the validity of C'_A. The verification steps are the same as those outlined in Section 4.2. In addition, the verifier needs to check whether C'_A's expiry date T'_e meets the non-repudiation policy as T'_e will also decide the expiry date of signatures generated with S'_A. Once C'_A is checked to be valid, the verifier can use V'_A to check $sS'_A(M)$. If the result is positive, A's signature can be regarded as valid non-repudiation evidence for the settlement of possible disputes. As the valid C'_A will remain valid until T'_e, the verifier may

[1] This is different from the concept of short-lived certificates proposed in [27] for authentication and access control.

[2] Other information that is required in the practical implementation is omitted here.

[3] Here we use a simplified time-stamping process for a concise description. A more robust time-stamping service [18] for commercial applications could be easily applied to our scheme.

[4] A tighter restriction on C'_A's expiry date T'_e may be imposed because of administrative reasons.

store it and directly use it later to check A's signatures appended with the same certificate C'_A.

5.3. Protection against key compromise

Although user A is not allowed to revoke its irrevocable signature keys, it becomes much easier to protect against the compromise of such keys than that of its revocable signature key. A need not keep its irrevocable signature keys until their expiry. Instead, A can destroy its used irrevocable signature keys and generate new irrevocable signature keys at any time it wishes, thus reducing the requirement of key management and the risk of key compromise.

Further, by imposing additional restrictions on the irrevocable verification key certificate C'_A, user A can limit the loss even if its irrevocable signature key S'_A is compromised. The restrictions may include:

- the types of transactions that S'_A can be applied to;
- the set of legitimate recipients of the signatures generated with S'_A;
- the maximum amount of a transaction which can be authorised by S'_A.

Thus, user A can generate an irrevocable verification key certificate C'_A as follows, which is limited to verify A's signatures on payment orders for recipients R_1, R_2 or R_3 with the maximum amount of \$1000.

limit = (payment order, R_1,R_2,R_3, \$1000)
$C'_A = V'_A, T'_e, limit, T'_g, sS_A(V'_A, T'_e, limit), sS_{TS}(sS_A(V'_A, T'_e, limit), T'_g)$.

Suppose R_1 receives the following signed message from A in a transaction where C'_A is defined with the above limit.

$C'_A, sS'_A(pay\ \$500\ to\ R_1\ldots)$.

In addition to the process of signature verification described in Section 5.2, R_1 should also check whether C'_A is authorised for the verification of A's signatures on payment orders, whether R_1 is specified as a legitimate recipient in C'_A, whether the amount of this payment is within the limit of C'_A. Only if the signed message meets these restrictions, can R_1 accept it safely as non-repudiation evidence.

5.4. Flexible expiry date

In the earlier scheme, user A has to choose C'_A's expiry date T'_e in advance, and C'_A can only be used in non-repudiation services where the expected expiry date of non-repudiation evidence is not later than T'_e. The improvement made below allows A to generate C'_A in advance but decide C'_A's expiry date later according to the non-repudiation service that C'_A will be applied to. This is based on the idea of using a hash chain, which can be found in the password authentication scheme [28] and the micropayment scheme [29].

User A first chooses three expiry dates: T'^{s}_e, T'^{m}_e, T'^{l}_e, representing *short-term*, *medium-term* and *long-term* expiry dates respectively. They could be set as (T'_g + one day), (T'_g + one week), or (T'_g + one month) etc., depending on the non-repudiation services that A may be involved in. A also chooses a random number n, and generates a hash chain

$$H^i(n) = H(H^{i-1}(n)) \qquad (i = 1, 2, 3),$$

Where $H^0(n) = n$. A keeps $H^0(n)$, $H^1(n)$, and $H^2(n)$ secret, but A only needs to store n. Then T'_e in C'_A can be defined as

$$T'_e = T'^{s}_e, T'^{m}_e, T'^{l}_e, H^3(n)$$

and C'_A becomes

$$C'_A = V'_A, T'_e, T'_g, sS_A(V'_A, T'_e), sS_{TS}(sS_A(V'_A, T'_e), T'_g), H^i(n)$$

$(0 \le i \le 2)$.

When A releases a different value of the hash chain, A can get the irrevocable verification key certificate with a different expiry date, that is

- C'_A has the expiry date of T'^{s}_e when $H^2(n)$ is released;
- C'_A has the expiry date of T'^{m}_e when $H^1(n)$ is released;
- C'_A has the expiry date of T'^{l}_e when $H^0(n)$ is released.

As H is a one-way hash function, it is computationally infeasible for anybody else to obtain $H^i(n)$ from $H^{i+1}(n)$ unless A has released $H^j(n)$ $(0 \le j \le i \le 2)$. Therefore, when a verifier presents C'_A with $H^i(n)$ $(0 \le i \le 2)$, A cannot repudiate the corresponding expiry date defined in C'_A.

Of course, A can release $H^j(n)$, even if A has already released $H^i(n)$ $(0 \le j < i \le 2)$, if A wants to use the same irrevocable signature key S'_A in a different non-repudiation service which requires a longer expiry date of non-repudiation evidence. However, this will also extend the expiry date of its signatures generated with S'_A in previous non-repudiation services to the one defined by $H^j(n)$ in C'_A.

5.5. Dispute resolution

In low risk business transactions, transacting parties only need to use non-repudiation evidence to settle disputes within a certain period. Imagine that a user pays his monthly telephone bill by cheque. The authorised amount will be deducted from his bank account. This will be reflected in the user's (monthly) bank statement. The user can question the correctness of the bank statement within a limited period (e.g. two weeks) of issue, which is defined in the bank's code of practice. No complaints from the user will be accepted afterwards. Hence, it will be sufficient for the bank to be sure that the user's signed authorisation is valid within such a limited period, and the bank need not keep the evidence indefinitely.

Our scheme enables the settlement of disputes arising from such applications. As digital signatures generated in our scheme have a limited valid period, all disputes related to a specific transaction that require user A's signature $sS'_A(M)$ as non-repudiation evidence should be brought to

Table 1
A comparison of efficiency

	Conventional approach	New scheme
Communication round with TS	n	1
Signature generation by TS	n	1
Signature generation by A	n	$n+1$
Signature verification	$n \times 2$	$n+2$

an arbitrator before this signature expires at time T'_e (referring to the arbitrator's clock). Users should be aware of the non-repudiation policy of business transactions that they will be involved in, and generate/accept digital signatures with appropriate expiry dates.

Suppose there is a dispute which requires $sS'_A(M)$ as a piece of non-repudiation evidence whose expiry date is T'_e. To prove the validity of this signature, the disputing party is required to submit the following messages to the arbitrator:

$C_A, C'_A, M, sS'_A(M)$.

Suppose the arbitrator received the submission at time T_s. The arbitrator will make the following checks:

1. The arbitrator checks that C_A was issued by CA with the expiry date of T_e.
2. The arbitrator checks that C'_A was signed by A and certified by TS, and C'_A satisfies $T'_g < T'_e \leq T_e$.[5]
3. The arbitrator checks that $sS'_A(M)$ was signed by A, and the signature does not breach the restrictions defined in C'_A.
4. The arbitrator checks that $T_s \leq T'_e$.

Only if all of the aforementioned checks are successful, will A's signature on message M be accepted as valid non-repudiation evidence.

6. Analysis of the scheme

6.1. Efficiency

Our scheme proposed in Section 5 is especially efficient for mass on-line transactions. Suppose user A will be involved in a batch of transactions in which n messages need to be signed. Table 1 shows the comparison result between the conventional time-stamping approach and our new scheme.

Our new scheme slightly increased A's computation overheads. However, it significantly reduced TS's communication and computation overheads, which is vital for efficient on-line transactions. It also reduced the signature verifier's computation overheads as long as $n > 2$. Hence, the performance of our scheme is much better than the conventional time-stamping approach in the case of mass on-line transactions.

6.2. Security

Here we examine possible attacks against our new scheme.

1. *User A denies that $sS'_A(M)$ is A's signature on M.*
   In our scheme, C'_A is an essential part of non-repudiation evidence. As C'_A is signed by A, A cannot deny that C'_A is an irrevocable verification key certificate issued by A. If $sS'_A(M)$ is verified successfully with V'_A which is certified in C'_A by A, A cannot deny that $sS'_A(M)$ is A's signature on M. Hence, both C'_A and $sS'_A(M)$ are required to hold A undeniable towards A's signature on M.
2. *User B generates C'_B in which $V'_B = V'_A$. Then B claims that (C'_B,$sS'_A(M)$) is B's signature on M.*
   As we just explained, $sS'_A(M)$ alone cannot be used to prove that it is A's signature on M. With $sS'_A(M)$, any user B can generate a signature on M by generating C'_B in which $V'_B = V'_A$. However, this kind of passive signature generation is harmful to B. Once A holds such C'_B, A can generate signatures on any messages which hold B undeniable. Hence, B should not generate signatures in that way.
3. *After C_A was revoked, user A continues using a valid C'_A created before revocation of C_A for signature generation.*
   This has no effect if A's signature is to make A undeniable in a transaction. However, it will be dangerous if A's signature represents the delegated authorisation, which means A is still able to act as an authority even if the delegation was revoked. Hence, it should be explicitly specified in C_A whether A is permitted to use S_A to generate C'_A.

7. Conclusion

We proposed a new approach to secure digital signatures as non-repudiation evidence. In this approach, a user delegates the signing authority from his long-term revocable signature key to a temporary irrevocable one for a fixed period of time, possibly with restrictions as to what can be signed with that temporary key. The temporary irrevocable verification key certificate is then countersigned by a trusted TS. Signatures generated with the temporary key are non-repudiable through the lifetime of the temporary verification key certificate. The new approach has the following properties:

- The process of signature generation is simplified. Users do not need to send each of their signatures to a TS.
- The process of signature verification is simplified. Once the irrevocable verification key certificate is checked to be valid, it remains valid before its expiry.
- Irrevocable verification key certificates can be generated

[5] If user A wants to deny the validity of C'_A, A is required to present evidence of revocation, e.g. the CRL, to prove that S_A was revoked before T'_g.

off-line before transactions take place. No extra messages need to be exchanged with a TS in on-line transactions.

The new approach will significantly improve the efficiency of mass on-line transactions in electronic commerce with an adjustable degree of risk that users prepare to take.

Acknowledgements

We would like to thank Chris Mitchell and Dieter Gollmann for helpful discussions, and the anonymous referees for valuable comments. The research towards this article was funded by the National Science and Technology Board of Singapore, and was conducted in the School of Computing, National University of Singapore.

References

[1] M. Roe, Cryptography and evidence. PhD Thesis, University of Cambridge, 1997.
[2] J. Zhou, Non-repudiation, PhD Thesis, University of London, December 1996.
[3] J. Zhou, D. Gollmann, Evidence and non-repudiation, Journal of Network and Computer Applications 20 (3) (1997) 267–281.
[4] ISO/IEC 10181-4. Information technology—open systems interconnection—security frameworks for open systems, Part 4: Non-repudiation framework, ISO/IEC, 1997.
[5] ISO/IEC 13888-1. Information technology—security techniques—non-repudiation, Part 1: General. ISO/IEC, 1997.
[6] ISO/IEC 13888-2. Information technology—security techniques—non-repudiation, Part 2: Mechanisms using symmetric techniques, ISO/IEC, 1998.
[7] ISO/IEC 13888-3. Information technology—security techniques—non-repudiation, Part 3: Mechanisms using asymmetric techniques, ISO/IEC, 1997.
[8] L. Hollaar, A. Asay, Legal recognition of digital signatures, IEEE Micro 16 (3) (1996) 44–45.
[9] ISO/IEC DIS 9798-3. Information technology—security techniques—entity authentication, Part 3: Mechanisms using digital signature techniques (second edition), ISO/IEC, 1998.
[10] S.G. Akl, Digital signatures: a tutorial survey, Computer 16 (2) (1983) 15–24.
[11] K.S. Booth, Authentication of signatures using public key encryption, Communications of the ACM 24 (11) (1981) 772–774.
[12] R. DeMillo, M. Merritt, Protocols for data security, Computer 16 (2) (1983) 39–50.
[13] A.J. Menezes, P.C. van Oorschot, S.A. Vanstone, Handbook of Applied Cryptography, CRC Press, Boca Raton, 1996.
[14] B. Schneier, Applied cryptography-Protocols, algorithms, and source code in C, 2, Wiley, New York, 1996.
[15] K.Y. Lam, D. Gollmann, Freshness assurance of authentication protocols, in: Lecture Notes in Computer Science 648, Computer Security: Proceedings of ESORICS'92, Toulouse, France, November 1992, pp. 261–271.
[16] K.Y. Lam, T. Beth, Timely authentication in distributed systems, in: Lecture Notes in Computer Science 648, Computer Security: Proceedings of ESORICS'92, Toulouse, France, November 1992, pp. 293–303.
[17] K.Y. Lam, Building an authentication service for distributed systems, Journal of Computer Security 2 (1) (1993) 73–84.
[18] S. Haber, W.S. Stornetta, How to time-stamp a digital document, Journal of Cryptology 3 (2) (1991) 99–111.
[19] ISO/IEC 9796. Information technology—security techniques—digital signature scheme giving message recovery, ISO/IEC, 1991.
[20] NIST FIPS PUB 186, Digital signature standard, National Institute of Standards and Technology, May 1994.
[21] A. Herzberg, M. Jakobsson, S. Jarecki, H. Krawczyk, M. Yung, Proactive public key and signature systems, in: Proceedings of Fourth ACM Conference on Computer and Communications Security, Zurich, Switzerland, April 1997, pp. 100–110.
[22] T. Coffey, P. Saidha, Non-repudiation with mandatory proof of receipt, Computer Communication Review 26 (1) (1996) 6–17.
[23] B. Cox, J.D. Tygar, M. Sirbu, NetBill security and transaction protocol, in: Proceedings of the First USENIX Workshop on Electronic Commerce, July 1995, pp. 77–78.
[24] R.H. Deng, L. Gong, A.A. Lazar, W. Wang, Practical protocols for certified electronic mail, Journal of Network and Systems Management 4 (3) (1996) 279–297.
[25] J. Zhou, D. Gollmann. A fair non-repudiation protocol, in: Proceedings of 1996 IEEE Symposium on Security and Privacy, Oakland, California, May 1996, pp. 55–61.
[26] C.J. Mitchell, Private communications, May 1998.
[27] Y.K. Hsu, S.P. Seymour, An Intranet security framework based on short-lived certificates, IEEE Internet Computing 2 (2) (1998) 73–79.
[28] L. Lamport, Password authentication with insecure communication, Communications of the ACM 24 (11) (1981) 770–772.
[29] T.P. Pedersen, Electronic payments of small amounts, in: Lecture Notes in Computer Science 1189, Proceedings of Cambridge Workshop on Security Protocols, Cambridge, April 1996, pp. 59–68.