Validating Digital Signatures without TTP's Time-Stamping and Certificate Revocation

Jianying Zhou, Feng Bao, and Robert Deng

Institute for Infocomm Research
21 Heng Mui Keng Terrace
Singapore 119613
{jyzhou,baofeng,deng}@i2r.a-star.edu.sg

Abstract. In non-repudiation services where digital signatures usually serve as irrefutable cryptographic evidence for dispute resolution, trusted time-stamping and certificate revocation services, although very costly in practice, must be available, to prevent big loss due to compromising of the signing key. In [12], a new concept called *intrusion-resilient signature* was proposed to get rid of trusted time-stamping and certificate revocation services and a concrete scheme was presented. In this paper, we put forward a new scheme that can achieve the same effect in a much more efficient way. In our scheme, *forward-secure signature* serves as a building block that enables signature validation without trusted time-stamping, and a *one-way hash chain* is employed to control the validity of public-key certificates without the CA's involvement for certificate revocation. We adopt a model similar to the intrusion-resilient signature in [12], where time is divided into predefined short periods and a user has two modules, *signer* and *home base*. The signer generates forward-secure signatures on his own while the home base manages the validity of the signer's public-key certificate with a one-way hash chain. The signature verifier can check the validity of signatures without retrieving the certificate revocation information from the CA. Our scheme is more robust in the sense that loss of synchronization between the signer and the home base could be recovered in the next time period while it is unrecoverable in [12]. Our scheme is also more flexible in the real implementation as it allows an individual user to control the validity of his own certificate without using the home base.

1 Introduction

Digital signature is a fundamental mechanism in security services such as authentication and non-repudiation. A pair of private key and public key are used in signature generation and verification, respectively. A digital signature of a message can be used to verify the origin and integrity of that message. Moreover, it can be used to protect against the signer's false denial of creating the signature if the private key is only known to the signer.

The relationship between a public key and an identity of the owner of the corresponding private key is usually established in the form of a public-key certificate that is issued by a *trusted third party* (TTP) called the *certification*

authority (CA). An expiry date specified in the certificate indicates its maximum lifetime that could be used to verify the digital signatures. In practice, as the private key might be compromised before the corresponding public-key certificate's scheduled expiry date, additional security mechanisms are required to prevent signature forgery with the compromised key.

1.1 Validity of Digital Signatures

Security requirements on digital signatures are different when they are used in authentication and non-repudiation services, respectively. While authentication services protect against masquerade, non-repudiation services provide evidence to enable the settlement of disputes [21].

In authentication services, the signature verifier only needs to make sure that both the signature and the public-key certificate are valid at the time of verification, and does not care about their validity afterwards. In non-repudiation services, however, the validity of a signature accepted earlier must be verifiable at the time of dispute resolution even if the corresponding public-key certificate has expired or been revoked. The scenarios illustrated in Figure 1 gives a clearer view on this problem.

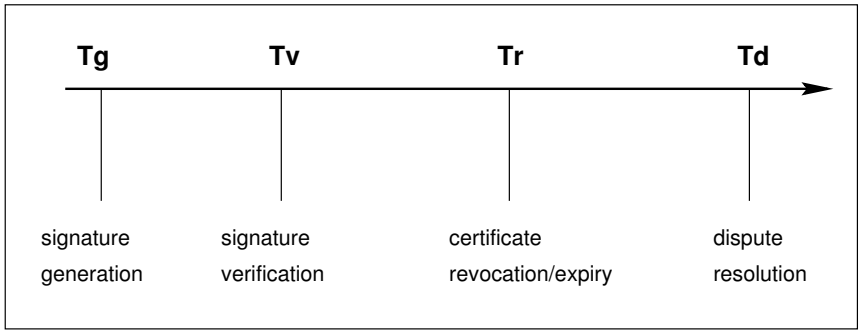

Fig. 1. Validity of Digital Signatures

Suppose a signature is generated at time T_g. The authentication service ends by time T_v, at which the validity of signature is checked. Success of authentication relies on whether the signature and the corresponding public-key certificate are valid at T_v. Revocation or expiry of the certificate at time T_r, where $T_r > T_v$, has no effect on the authentication service.

Suppose a dispute resolution takes place at time T_d, where $T_d > T_r$. Obviously, the certificate is invalid at T_d. On the other hand, the signature has been accepted as non-repudiation evidence at T_v, where $T_v < T_r$. If the signature is treated as invalid at T_d because of certificate revocation (or expiry), any party can generate a signature and later deny it by revoking the certificate. Therefore, it is critical to ensure that once a signature is accepted as valid evidence, it

remains valid even if the corresponding certificate is revoked or expires at a later time.

A number of mechanisms exist for maintaining validity of digital signatures as non-repudiation evidence. However, most of the existing mechanisms (e.g., [3, 6, 7, 23]) rely on supporting services from trusted third parties, e.g., time-stamping [2] and certificate revocation [10, 17]. That means a transacting party in an on-going session needs to establish extra connections with TTPs.

- It may need to connect with a *time-stamping authority* (TSA) for trusted time-stamping on a signature.
- It may need to connect with a CA for certification revocation information.

Obviously, this is less efficient for on-line transactions, and sometimes infeasible for a mobile device to support simultaneous connections.

There exist two mechanisms that validate digital signatures without trusted time-stamping and certificate revocation, *one-way sequential link* signatures [22] and *intrusion-resilient* signatures [12]. However, the first mechanism is only limited to B2B applications, where the transacting parties usually have a regular business relationship, and both sides maintain a long-term transaction log. The second mechanism is fragile in real applications as loss of synchronization between the signer and the home base can cause the signer unable to generate valid signatures any more. A brief review of these mechanisms will be given in Section 2.

1.2 Our Results

In this paper, we propose a new scheme for validating digital signatures as non-repudiation evidence. We adopt a model similar to the intrusion-resilient signature in [12] and achieve the same effect (i.e., validating signatures without relying on trusted third parties for time-stamping and certificate revocation) in a much more efficient way. In our scheme, *forward-secure signature* serves as a building block that enables signature validation without trusted time-stamping. A *one-way hash chain* is employed to control the validity of public-key certificates without the CA's involvement for certificate revocation. A user has two modules, *signer* and *home base* as in [12], but they function differently. The signer generates forward-secure signatures on his own while the home base manages the validity of the signer's public-key certificate with a one-way hash chain.

When the signer requests a new public-key certificate, the home base generates a one-way hash chain and keeps the root hash value confidential, then sends the last chained hash value to the CA to be embedded into the certificate. The maximum lifetime of the certificate is divided into predefined short periods, each of which is related to a chained hash value. At the beginning of a time period, the signer obtains the corresponding hash value from the home base to refresh the validity of his certificate.

The signer also updates his forward-secure signing key at the beginning of a time period and generates forward-secure signatures. A valid forward-secure

signature can only be generated with the signing key related to the time period in which the signature is generated. In other words, forward-secure signatures are time-related, and their validity can be preserved without relying on a trusted time-stamping service.

Signing key update is a one-way function, and the compromise of the current signing key will not lead to the compromise of past signing keys. However, it will result in the compromise of future signing keys, thus the forgery of signatures in future time periods. In our scheme, a one-way hash chain is used to control the validity of the signer's public-key certificate. The certificate may expire at the end of the current time period thus invalidating all future signing keys if the home base destroys the hash chain root and stops releasing hash values. The signature verifier can check the status of such a certificate without retrieving the revocation information from the CA or a designated directory.

Our scheme is more robust in the sense that loss of synchronization between the signer and the home base could be recovered in the next time period while it is unrecoverable in [12]. Our scheme is also more flexible in the real implementation as it allows an individual user to control the validity of his own certificate without using the home base.

2 Previous Work

Here we give a brief review of two existing mechanisms that validate digital signatures without trusted time-stamping and certificate revocation, and point out their limitations and weaknesses.

2.1 One-Way Sequential Link Signature

One-way sequential link mechanism provides a solution for maintaining validity of digital signatures generated in transactions between two parties without the TTP's intervention for time-stamping and certificate revocation [22]. The main idea is to link all digital signatures generated by a transacting party in a way that any change to the link will be detected. The transacting party can revoke his signing key by sending the first and the latest digital signatures in the link to the trading partner for counter-signing. With the trading partner's approval, the transacting party can deny digital signatures that are generated with his revoked key but are not in the counter-signed link.

Suppose two parties A and B are going to do a series of transactions, and A needs to generate signatures on messages $X_1, X_2, \cdots, X_i$. A can establish a one-way sequential link of his digital signatures $s_{A_1}, s_{A_2}, \cdots, s_{A_i}$ as follows.

$$
\begin{aligned}
s_{A_1} &= S_A(X_1, n_1) \\
s_{A_2} &= S_A(X_2, H(s_{A_1}), n_2) \\
&\cdots \\
s_{A_i} &= S_A(X_i, H(s_{A_{i-1}}), n_i)
\end{aligned}
$$

Here, H is a collision-resistant one-way hash function. $n_1, \cdots, n_i$ are incremental sequential numbers or local time stamps serving as an index of the one-way sequential link, which could be used to facilitate dispute resolution. Suppose A's public-key certificate is C_A. When B receives s_{A_k}, B needs to make the following checks before accepting and saving it.

1. B verifies A's signature s_{A_k}.
2. B checks that C_A has not expired and is not marked as revoked in B's transaction log.
3. B checks that s_{A_k} is linked properly in the one-way sequential link associated with C_A.

Similarly, B can also establish a one-way sequential link of his digital signatures $s_{B_1}, s_{B_2}, \cdots, s_{B_j}$ for transactions conducted with A.

Suppose A wants to revoke his C_A while the latest signatures in the one-way sequential links of A and B are $s_{A_{i-1}}$ and $s_{B_{j-1}}$, respectively. A can send B a request

$$s_{A_i} = S_A(revoke, s_{A_1}, s_{A_{i-1}}, H(s_{A_{i-1}}), n_{A_i})$$

and B can reply with an approval

$$s_{B_j} = S_B(approve, s_{A_1}, s_{A_i}, H(s_{B_{j-1}}), n_{B_j})$$

and mark C_A as revoked. Then, A is only liable to the signatures appeared in the link starting from s_{A_1} and ending at s_{A_i}, and can deny any other signatures intended for B and associated with C_A. If needed, B can also terminate his one-way sequential link in the same way.

Limitation 1: One-way sequential link mechanism is only limited to B2B applications, where the transacting parties usually have a regular business relationship. Each party should maintain a long-term transaction log, and cooperation is needed in approving the other party's revocation request.

Limitation 2: One-way sequential link mechanism only supports certificate revocation by the transacting parties themselves. In some applications, however, a transacting party's certificate may need to be revoked by its manager to terminate the power of signing.

2.2 Intrusion-Resilient Signature

Intrusion-resilient signature was proposed in [12] for the purpose of getting rid of trusted time-stamping and certificate revocation in validating digital signatures. The provably-secure scheme presented in [12] is refined in mathematics. However, a weakness exists in the scheme from the implementation viewpoint. We reason that the theoretical merit of the scheme does not lead to the practical implementation.

The scheme in [12] takes the model where each user (signer), besides securely storing his signing key SKU, has a secret key SKB stored in his *home base*,

which is used for updating the private signing key. There are two operations in updating the signing key, *key update* and *key refresh*. Time is divided into T short periods. *Key update* is conducted once at the end of each period while *key refresh* may be done multiple times within each period.

The operation of *key refresh* brings fragility to the scheme in practical application of the scheme. In this operation, the secret SKB stored at the home base is refreshed with a randomly chosen number r, i.e., $SKB := refresh_{base}(SKB, r)$. The r and old SKB are then destroyed by the home base. The new SKB is sent to the user, which is used to refresh the signing key SKU, i.e., $SKU := refresh_{user}(SKU, SKB)$. This is equivalent to requiring a sort of synchronous status between SKU and SKB. The "synchronization" might be broken due to various possibilities, such as the user misses the SKB sent to him or an attacker succeeds somehow to conduct refresh operation with the home base. In that case, the SKU held by the user and the SKB stored at the home base are not consistent, and the synchronization can never be recovered as the home base has erased r and old SKB. As a result, the signing key cannot be updated correctly any more, and the corresponding public key has to be given up for ever. Our scheme does not suffer from this fragility in the sense that even if the update message from the home base to the user is lost or intercepted, it only affects the current period but not the periods after the current period.

Key-insulated signature [8] employs a model similar to the above intrusion-resilient signature scheme. However, its security is weaker in some sense. If the signer's current signing key is compromised, and the current key update information is also intercepted by an attacker, signatures of both the current and the subsequent periods could be forged. Moreover, efficiency is another concern because the length of public key grows linearly with the number of key-insulated periods. This is also true to master key stored in the home base for signing key update.

There is a common weakness in both the intrusion-resilient and key-insulated signature schemes which did not deal with signature forgery once the signing key is compromised in the *current* time period. Although it might be a short time period that an adversary can forge signatures, the insecure window could leave the validity of all signatures generated in such a period in question. We will further discuss this problem in our scheme.

3 A New Signature Validation Scheme

Here we propose a new scheme that validates digital signatures without the TTP's intervention for time-stamping and certificate revocation. The objective of our new scheme is to remove the limitations and weaknesses existing in the previous mechanisms.

3.1 Forward-Secure Signature

Forward-secure signature serves as a building block that enables signature validation without trusted time-stamping. In a forward-secure signature scheme,

time is divided into predefined short periods, and the public key is fixed while the corresponding signing key is updated at the beginning of each period with a public one-way function. Several schemes have been proposed in the past few years [1, 4, 11, 14–16, 19].

Definition A forward-secure digital signature scheme is a quadruple of algorithms, (FSkeygen, FSsign, FSupdate, FSverify), where:

> FSkeygen, the key generation algorithm, is a probabilistic algorithm which takes as input a security parameter k and returns a pair (SK_1, PK), the initial signing key and the public key; (PK may be registered with the CA.)
>
> FSsign, the signing algorithm, takes as input the signing key SK_j for the current time period j and the message M to be signed and returns a pair $<j, sign>$, the signature of M for time period j;
>
> FSupdate, the signing key update algorithm, takes as input the signing key for the current period SK_j, returns the new signing key SK_{j+1} for the next period and erases the past signing key SK_j; (The signing key update in each time period is illustrated in Figure 2.)
>
> FSverify, the verification algorithm, takes as input the public key PK, a message M, and a candidate signature $<j, sign>$, and returns 1 if $<j, sign>$ is a valid signature of M or 0, otherwise.

Suppose a signer A's signing key SK_j is compromised at the time period j. Others cannot derive the past signing key SK_p from SK_j ($p<j$) as FSupdate is a one-way function. In addition, others cannot use the compromised key SK_j to forge a valid signature of a time period other than j as the verification will fail if $p \neq j$ is used as an input of FSverify for verifying a signature generated with SK_j. Therefore, even if the current signing key SK_j is compromised, the validity of signatures generated before the time period j is not affected. In other words, forward-secure signatures of the past time periods could be validated without trusted time-stamping and certificate revocation.

However, as FSupdate is a public function, once SK_j is compromised, others can derive the future signing key SK_f from SK_j ($f>j$), thus can forge valid signatures of the future time period f. Therefore, once SK_j is compromised, A needs to inform the CA to revoke the corresponding certificate. Meanwhile, the signature verifier needs to get the revocation information from the CA to check the validity of the certificate.

A new mechanism is required for validating forward-secure signatures without the CA's intervention for certificate revocation.

3.2 Refreshable Certificate

The *refreshable certificate* is an extension of the standard public-key certificate. The maximum lifetime of such a certificate is divided into short periods. The

signer A is allowed to refresh the validity of his own certificate under the control of his home base, and the signature verifier can check the validity of A's certificate without retrieving the revocation information from the CA.

Role of Home Base A home base serves as a manager who has authority over a group of signers. It plays a role different from the CA.

- A home base is only recognized by the signers under his management, and manages the validity of their certificates.
- The CA is a more widely recognized trusted third party, and issues public-key certificates for a larger population.

A home base is not required to be trusted by or connected with any entity other than the signers under his management. Therefore, the cost for establishment of a home base is much lower than that of the CA, and the bottleneck problem related to a home base is much less serious than to the CA. Co-operation with a home base in signature generation is not a weakness, but an advantage, in our scheme. As the home base acts as a manager of those signers, it can terminate a signer's power of signing at the end of current time period if needed (e.g. because of change of job).

Generation of Certificate Generation of a refreshable certificate involves three parties: the signer A, A's home base, and the CA.

Step 1. Actions by the signer A:

1. Generate a pair of keys: private key SK_A and public key PK_A.
2. Send PK_A to his home base over an authenticated channel [1].

Step 2. Actions by A's home base:

1. Define the maximum lifetime of A's certificate as T time periods and the starting valid date as D, and select the length of each time period as L. (The refreshing points are denoted as $D_1 = D + L, D_2 = D + 2 * L, \cdots, D_T = D + T * L$ and illustrated in Figure 2.)
2. Select a random number r, and generate a one-way hash chain $H^i(r) = H(H^{i-1}(r))$ $(i = 1, 2, \cdots, T)$, where $H^0(r) = r$.
3. Send $(A, PK_A, D, H^T(r), T, L)$ to the CA.

Step 3. Actions by the CA:

1. Authenticate the certificate request with A's home base in an out-of-band method to ensure the request is authorized by the home base [2].

[1] Suppose the signer A has registered with his home base. The authenticated channel can be established with a password-based protocol (e.g., [5, 20]).

[2] On-line authentication could be performed if a secure channel exists between A's home base and the CA.

2. Challenge A for a signature to ensure A holds the corresponding private key.
3. Generate a certificate $Cert_A = S_{CA}(A, PK_A, D, H^T(r), T, L)$ [3].
4. Issue $Cert_A$ to A (via A's home base).

Compared with a standard public-key certificate, $Cert_A$ contains the extra data $(H^T(r), T, L)$ [4]. They will be used to control the validity of $Cert_A$.

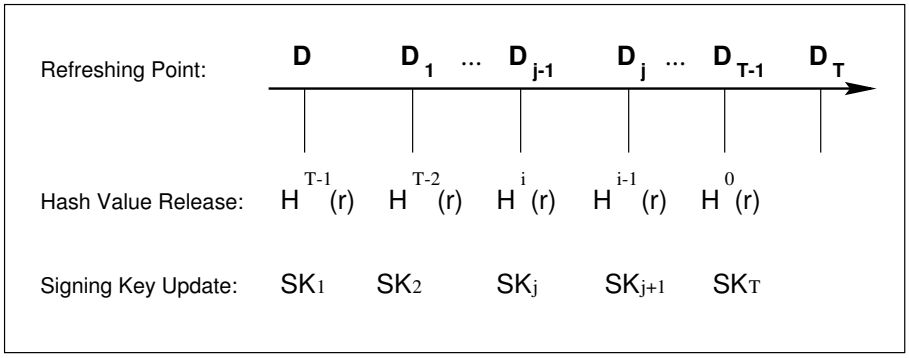

Fig. 2. Hash Value Release and Signing Key Update

Use of Certificate At the starting valid date D, A retrieves $H^{T-1}(r)$ from his home base to initialize the validity of $Cert_A$ [5], which then has an expiry date $D_1 = D + L$. Suppose the next refreshing point of $Cert_A$ is D_j. A retrieves $H^i(r)$, where $i = T - (D_j - D)/L = T - j$, from his home base, and attaches $(H^i(r), i)$ to each signature generated in the period between D_{j-1} and D_j [6]. (The hash value release at each refreshing point is illustrated in Figure 2.)

Note that it is entirely up to A for retrieving the hash value from his home base at a refreshing point. For example, if A does not generate any signature in the period between D_{j-1} and D_j, A does not need to retrieve $H^i(r)$. But later if A wants to generate signatures in the period between D_j and D_{j+1}, A can directly retrieve $H^{i-1}(r)$. On the other hand, the home base has the full control on the validity of A's certificate. If A's authorization on signing with SK_A must be revoked for some reasons such as change of job or key compromise, the home base can stop releasing the next hash value.

Suppose A released $(H^i(r), i)$; the current time is D_v; a signature verifier B holds the CA's public verification key. B can take the following steps to check the status of $Cert_A$.

[3] For simplicity, other less related information is omitted in $Cert_A$.

[4] $Cert_A$ should also include an identifier of the hash function used to generate and verify the hash chain.

[5] The home base could select an optimal technique in the computation-storage trade-off of hash chain traversal [18].

[6] $(H^i(r), i)$ need not be a part of message to be signed. Instead, it is only the data that will be stored or transmitted together with the signature.

1. B verifies the CA's signature on $(A, PK_A, D, H^T(r), T, L)$. If valid, B is sure that A's public key is PK_A. The starting valid date is D, the maximum lifetime is $T * L$, the refreshing time period is L, and the last hash value in the one-way hash chain is $H^T(r)$.
2. B checks that $0 \leq i < T$ and $H^{T-i}(H^i(r)) = H^T(r)$. If true, B believes that $H^i(r)$ is a valid hash value in the one-way hash chain ended with $H^T(r)$.
3. B checks that $D_v \leq D + (T - i) * L$. If true, B concludes that $Cert_A$ is valid now, and remains valid until $D_j = D + (T - i) * L$.

In such a way, the validity of $Cert_A$ can be controlled by releasing the corresponding hash value when A generates digital signatures. B can check the status of $Cert_A$ without retrieving the revocation information from the CA. Thus, the CA is exempted from certificate revocation.

3.3 Signature Validation

With the forward-secure signature and refreshable certificate, we can validate digital signatures without relying on the trusted third parties for time-stamping and certificate revocation.

Suppose a signer A wants to generate a signature in the period $j - 1$, i.e., between refreshing points D_{j-1} and D_j. A's corresponding signing key is SK_j. A forward-secure signature on message M is denoted as $\mathsf{FSsign}(M, SK_j) = <j, s>$. If A is sure that SK_j is not compromised at the end of this period, A performs $\mathsf{FSupdate}$ and retrieves $H^i(r)$ and $H^{i-1}(r)$, where $i = T - j$ for $j \leq T - 1$, from his home base [7]. Then A attaches $<H^i(r), H^{i-1}(r), i>$ to the signature.

By releasing $H^i(r)$ and $H^{i-1}(r)$, the signer A is liable for all signatures generated with forward-secure signing keys SK_p for $p \leq j$ (i.e., signatures generated before the refreshing point D_j). But A can deny all signatures generated with SK_f for $f > j$ (i.e., signatures generated after the refreshing point D_j) if $H^{i-2}(r)$ and its pre-images in the hash chain are not released.

Suppose a signature verifier B receives $<M, j, s, H^i(r), H^{i-1}(r), i>$ from A. B makes the following checks to conclude whether the signature and the related information could serve as valid non-repudiation evidence.

1. B verifies A's signature by checking the output of $\mathsf{FSverify}(M, j, s, PK)$. If true, B believes that $<j, s>$ is A's signature on M generated during the time period $j - 1$, i.e., between refreshing points D_{j-1} and D_j.
2. B further verifies the status of $Cert_A$ with $<H^i(r), i>$ as described before. If true, B believes that $Cert_A$ is valid until the refreshing point D_j.
3. With $H^{i-1}(r)$, B concludes that A is committed being responsible for all signatures generated between refreshing points D_{j-1} and D_j. Thus B can accept $<j, s, H^i(r), H^{i-1}(r), i>$ as valid non-repudiation evidence.

[7] To reduce the delay between signature generation in the period $j - 1$ and release of $H^{i-1}(r)$ at the beginning of the period j, the refreshing period L may need to be defined short. Alternatively, A may release $H^{i-1}(r)$ in the period $j - 1$ if A is willing to undertake the risk of compromise of SK_j.

Suppose SK_j is compromised, and A has released $H^{i-1}(r)$ in the period $j-1$. Although an adversary can derive SK_{j+1} from SK_j and $Cert_A$ is valid in the corresponding period j (i.e., between refreshing points D_j and D_{j+1}), the adversary cannot forge valid signatures in the period j as long as A has not released $H^{i-2}(r)$ which is a part of non-repudiation evidence in the above signature validation process.

The above signature validation process shows that signature generation and verification are not necessary to be taken in the same period. Instead, given a forward-secure signature being generated in an arbitrary period, the verifier simply makes sure that the certificate is not only valid in the period that the signature was generated but also valid in the next period.

The clocks of the signer and the verifier may not be strictly synchronized to the real time, and a relative time could be used in our scheme. The signer A can generate a signature with SK_j in a relative time period $j-1$ while the actual time may not be the one between D_j and D_{j+1}. The verifier can check the validity of the signature without referring to the actual time of signature generation and verification [8]. Therefore, the time information related to such a forward-secure signature cannot be used to prove the actual time of signature generation.

Even if the clocks of the signer and the verifier are synchronized to the real time, the time information still cannot be used to prove that a forward-secure signature was actually generated within that time period. The signer can always use the signing key corresponding to a specific time period to generate a back-dated forward-secure signature (if the signer keeps the signing keys of the previous periods) or a post-dated forward-secure signature. Therefore, if the evidence on the time of signature generation is required in non-repudiation services, trusted time-stamping is necessary.

3.4 Optimization for Individual Users

In the above signature validation scheme, the home base acts as a manager of a group of signers for controlling the validity of their public-key certificates. This is especially useful in an enterprise environment, where each staff's power of signing is authorized by his manager.

If a signer is an individual user who takes the full responsibility on the validity of his own certificate, our scheme could be optimized to enable signature validation without using the home base. In such a case, the hash chain root r will be generated by the signer instead of his home base. Then it is up to the signer whether to release a hash value to refresh the validity of his certificate.

There is an advantage on the use of a separate secret r to protect the signing key SK_i for an individual signer A. The system remains secure as long as either r or SK_i is not compromised. If SK_i is compromised, A could destroy r then $Cert_A$

[8] In practice, however, it is safer for the verifier to synchronize his clock to the real time, and only accept signatures while the corresponding certificate has not reached its maximum lifetime.

will expire shortly at the next refreshing point. Similarly, if r is compromised, A could destroy SK_i and stop using it for signing.

The hash chain root r and the signing key SK_i are different in two aspects.

- The signing key might be used at any time while the hash chain root is needed only at the refreshing points. That means SK_i should be highly available in a system while r could be kept "off-line".
- The signing key usually has a length of 1024 bit or above while the hash chain root can be as short as 128 bits. That implies SK_i is usually beyond the human's capability to memorize while r might be memorized.

Consequently, the signer A could protect r in a way different from SK_i. A might remember r and manually input r at the time of refreshing $Cert_A$. After the hash value needed for refreshing is generated, r will be erased from the local computer system thus minimizing the possibility of compromise caused by system break-in.

There might be other ways to protect r "off-line" in a temper-proof hardware while keeping SK_i "on-line" in a local system.

3.5 Comparison

Our signature validation scheme removes the limitations of the one-way sequential link mechanism in [22]. It is not restricted to B2B applications. It is especially useful in B2C applications where the number of customers are much more than the number of merchants.

Suppose a merchant does 1000 transactions with 1000 customers, and generates one signature in each transaction within a time period (e.g., one day). With our new signature validation scheme, the merchant only needs to retrieve one hash value from his home base for generating 1000 signatures, and the customers do not need to make any connection to the CA when verifying the merchant's signatures and certificate. In comparison, if a *certificate revocation list* (CRL) is used, each customer needs to retrieve the CRL from the CA in order to verify the merchant's signatures and certificate. The operation cost is at least 1000 times higher than our scheme.

Many of customers in B2C applications are likely individual users, then our scheme even supports validation of signatures generated by such customers without using the home base. This allows for a more flexible implementation of our signature validation scheme compared with the intrusion-resilient signature in [12].

The requirement on the synchronization between the signer and the home base in our signature validation scheme is much weaker than in [12]. Loss of synchronization in [12] is fatal and unrecoverable. In our scheme, however, unavailability of a hash value from the home base only causes the signer unable to generate signatures in one time period. The signer's power of signing can be recovered when receiving the next hash value in the next time period.

Actually, temporary unavailability of a hash value from the home base is a feature of our signature validation scheme. It allows the home base (manager) to temporarily disable the signer's power of signing for some reason (e.g., on leave), and restore the signer's power later if necessary [9]. Therefore, our scheme is more robust with enriched functions in the real implementation.

There is a risk of signature forgery in [12] if the *current* signing key is compromised but has not expired yet. It may leave the validity of all signatures generated in the current period in question. Such a risk is handled in our scheme by requiring the singer to postpone releasing the hash value of the next period (which serves as a part of non-repudiation evidence) until the current signing key expires. That means a delay might be introduced in signature validation, though the delay could be reduced by defining a short refreshing period L. Therefore, our scheme will not completely replace *instant* and *trusted* time-stamping and certificate revocation services which are still useful for high value transactions that cannot tolerate any risk of signature forgery and delay of signature validation.

4 Conclusion

Maintaining validity of digital signatures is of significant importance in non-repudiation services. The conventional approach is to rely on trusted third parties to provide time-stamping and certificate revocation services, both of which are very costly in practice. Intrusion-resilient signature is a new concept proposed in [12] to get rid of time-stamping and certificate revocation in signature validation. The scheme is secure but has some weaknesses which may affect its robustness, efficiency, and flexibility in practical implementation.

In this paper, we proposed a new approach for signature validation. It achieves the same effect (i.e., validating signatures without trusted time-stamping and certificate revocation) in a much more efficient way, and overcomes the weaknesses in the existing schemes. In our scheme, *forward-secure signature* serves as a building block that enables signature validation without trusted time-stamping. A *one-way hash chain* is employed to control the validity of public-key certificates without the CA's involvement for certificate revocation. Some attractive features of our signature validation scheme include

- The signer's power of signing can be temporarily disabled by the home base and recovered later if necessary.
- Signatures generated by an individual user can be validated without using the home base.

References

1. M. Abdalla and L. Reyzin. *A new forward-secure digital signature scheme.* Lecture Notes in Computer Science 1976, Advances in Cryptology: Asiacrypt'00, pages 116–129, Kyoto, Japan, December 2000.

[9] The manager should be careful when deciding to restore a signer's power of signing as this will also result in a recovery of the signer's power in the past disabled periods.

2. C. Admas, P. Cain, D. Pinkas and R. Zuccherato. *Internet X.509 public key infrastructure time-stamp protocol (TSP)*. RFC 3161, August, 2001.
3. S. G. Akl. *Digital signatures: a tutorial survey*. Computer, 16(2):15–24, February 1983.
4. M. Bellare and S. Miner. *A forward-secure digital signature scheme*. Lecture Notes in Computer Science 1666, Advances in Cryptology: Proceedings of Crypto'99, pages 431–438, Santa Barbara, California, August 1999.
5. S. Bellovin and M. Merritt. *Encrypted key exchange: Password-based protocols secure against dictionary attacks*. Proceedings of 1992 IEEE Symposium on Security and Privacy, pages 72–84, Oakland, California, May 1992.
6. K. S. Booth. *Authentication of signatures using public key encryption*. Communications of the ACM, 24(11):772–774, November 1981.
7. R. DeMillo and M. Merritt. *Protocols for data security*. Computer, 16(2):39–50, February 1983.
8. Y. Dodis, J. Katz, S. Xu, and M. Yung. *Strong key-insulated signature schemes*. Lecture Notes in Computer Science 2567, Proceedings of 2003 International Workshop on Practice and Theory in Public Key Cryptography, pages 130–144, Miami, January 2003.
9. L. C. Guillou and J. J. Quisquater. *A paradoxical identity-based signature scheme resulting from zero-knowledge*. Lecture Notes in Computer Science 403, Advances in Cryptology: Proceedings of Crypto'88, pages 216–231, Santa Barbara, California, August 1988.
10. R. Housley, W. Ford, W. Polk, and D. Solo. *Internet X.509 public key infrastructure certificate and CRL profile*. RFC 2459, January 1999.
11. G. Itkis and L. Reyzin. *Forward-secure signatures with optimal signing and verifying*. Lecture Notes in Computer Science 2139, Advances in Cryptology: Proceedings of Crypto'01, pages 332–354, Santa Barbara, California, August 2001.
12. G. Itkis and L. Reyzin. *SiBIR: Signer-base intrusion-resilient signatures*. Lecture Notes in Computer Science 2442, Advances in Cryptology: Proceedings of Crypto'02, pages 499–514, Santa Barbara, California, August 2002.
13. ITU-T. *Information technology – Open systems interconnection – The directory: Public-key and attribute certificate frameworks*. ITU-T Recommendation X.509(V4), 2000.
14. A. Kozlov and L. Reyzin. *Forward-secure signatures with fast key update*. Proceedings of 3rd Conference on Security in Communication Networks, Amalfi, Italy, September 2002.
15. H. Krawczyk. *Simple forward-secure signatures from any signature scheme*. Proceedings of 7th ACM Conference on Computer and Communications Security, pages 108–115, Athens, Greece, November 2000.
16. T. Malkin, D. Micciancio, and S. Miner. *Efficient generic forward-secure signature with an unbounded number of time period*. Lecture Notes in Computer Science 2332, Advances in Cryptology: Proceedings of Eurocrypt'02, pages 400–417, Amsterdam, The Netherlands, April 2002.
17. M. Myers, R. Ankney, A. Malpani, S. Galperin and C. Adams. *X.509 Internet public key infrastructure on-line certificate status protocol (OCSP)*. RFC 2560, June 1999.
18. Y. Sella. *On the computation-storage trade-offs of hash chain traversal*. Lecture Notes in Computer Science, Proceedings of 2003 Financial Cryptography, Gosier, Guadeloupe, January 2003.

19. D. Song. *Practical forward secure group signature schemes*. Proceedings of 8th ACM Conference on Computer and Communication Security, pages 225–234, Philadelphia, November 2001.
20. T. Wu. *The secure remote password protocol*. Proceedings of 1998 Internet Society Network and Distributed System Security Symposium, pages 97–111, San Diego, California, March 1998.
21. J. Zhou. *Non-repudiation in electronic commerce*. Computer Security Series, Artech House, 2001.
22. J. Zhou. *Maintaining the validity of digital signatures in B2B applications*. Lecture Notes in Computer Science, Proceedings of 2002 Australasian Conference on Information Security and Privacy, pages 303–315, Melbourne, Australia, July 2002.
23. J. Zhou and K. Y. Lam. *Securing digital signatures for non-repudiation*. Computer Communications, 22(8):710–716, Elsevier, May 1999.