PUBLIC KEY CRYPTOGRAPHY AND A FRAMEWORK THEREFOR

BACKGROUND AND FIELD OF THE INVENTION

This invention relates to public key cryptography, more particularly to verification of a public key.

Public-key cryptography has been widely used in various security applications since its invention by Diffie and Hellman in 1976. In contrast to conventional cryptography, a pair of keys are used: a private key that is kept confidential by a certain party, and a public key that is available to the public. Depending on the public-key cryptographic algorithm, the public key can be used to encrypt a message or to verify a signature while the corresponding private key could be used to decrypt the cipher text or to generate the signature. It is computationally hard to derive the private key from the public key.

Digital signature is one of the most important applications of public-key cryptography, and is the fundamental mechanism for authentication and non-repudiation services. The signer can generate a digital signature of a message using his private key. The receiver can check the origin and integrity of the message by verifying the digital signature with the corresponding public key. Moreover, if the private key is only known to the signer, the signer cannot deny originating the signature since other parties are unable to forge the signature without the private key.

To determine the origin of a signed message, the verifier first needs to make sure what is the claimed signer's public key. Public-key certificates play an important role in binding the public key with the identity of the owner of the corresponding private key. X.509 is an industry standard which defines the format of a public-key certificate. The elements of a public-key certificate include the public key of an entity, the entity's distinguished identifier, an expiry date of the certificate, and an identifier of the cryptographic algorithm with which the public key is to be used. To ensure the authenticated identity of a certificate owner, the certificate needs to be issued by a trusted third party (TTP) called the certification authority (CA).

As a private key might be compromised before the corresponding public-key certificate's scheduled expiry date, any party holding a compromised key can forge the signature. Therefore, additional security mechanisms are needed to prevent this. A straightforward approach is that the owner of the certificate requests the issuing CA to revoke the certificate. The certificate revocation information will be accessible to public thus the relevant users will be aware that an apparently valid certificate is no longer valid.

The IETF PKIX Working Group is developing an Internet standard to support an X.509 based public-key infrastructure (PKI) (R. Housley, W. Ford, W. Polk, and D. Solo. *"Internet X.509 public key infrastructure certificate and CRL profile"*. RFC 2459, January 1999). The PKI provides a framework for services relating to

issuing public-key certificates and distributing revocation information. Certificate revocation and validation is a significant burden to the PKI.

It is an object of the invention to provide a public key cryptographic framework which alleviates at least one of the disadvantages of the prior art and/or provides the public with a useful choice.

SUMMARY OF THE INVENTION

According to the invention in a first aspect, there is provided a method of performing a public/private key operation comprising the steps of: generating a private and public key pair, the public key having a lifetime; selecting a plurality of intervals within the lifetime and a plurality of respective identifiers, each interval being associated with a said identifier; and having a trusted third party generate a public key certificate which includes the public key and a final identifier, the identifiers being selected so that it is relatively difficult to derive an identifier associated with a said interval from the final identifier but relatively easy to derive the final identifier from an identifier associated with a said interval, a said identifier being provided by a user with any matter signed or otherwise operated upon using the private key in the associated interval to confirm the validity of the certificate in that interval.

Preferably, a user of the private key holds the certificate or the certificate is held in a public location. One or more identifiers or means from which the identifiers are

generated may be stored on a remote server with a user wishing to encrypt matter using the private key typically either obtaining the identifier for the current period from the server or obtaining said means from the server and generating the identifier using the means, with the user and server preferably communicating via a secure channel.

The identifiers are preferably generated by recursive application of a one-way hash function to a root.

The user may obtain a time stamped version of the encrypted matter to be accompanied by the identifier and/or the private key may have validity within a time period, the user sending a time of encryption with the encrypted matter and/or each interval may have associated therewith an interval private key, the interval private keys being each usable together with the public key in a public/private key encryption/decryption operation, the interval keys being such that it is relatively easy to derive a later interval key from an earlier interval key but relatively difficult to derive an earlier interval key from a later one.

In the claimed method typically a third party operates on the current interval identifier to derive the final identifier to confirm the validity of the certificate in the current interval, together with any additional authentication/verification procedures as mentioned by way of example in the preceding paragraph.

A third party wishing to confirm the validity of the certificate may request from the user the identifier of the current time period and determines if the final identifier of the certificate can be derived from the current identifier. The third party may then use the public key verified by the certificate to encrypt matter to send to the user.

According to the invention in a second aspect, there is provided a method of conducting a public key transaction comprising the steps of: generating a public and private key pair, the private key being held by the user and the public key being accessible to a party to the transaction and having associated therewith a public key certificate issued by a trusted third party; sending to the party matter signed or otherwise operated upon using the private key by the user together with a matter identifier; the certificate including a certificate identifier , the identifiers being such that it is relatively easy to derive the certificate identifier from the matter identifier but relatively difficult to derive the matter identifier from the certificate identifier; and operating on the matter identifier provided by the user to derive the certificate identifier to confirm the validity of the certificate.

According to the invention in a third aspect, there is provided a public key certificate issued by a trusted third party comprising a public key, a final identifier and means defining validity periods between a start date and a finish date, a plurality of identifiers not forming part of the certificate each being associated with a said validity period and arranged to accompany matter signed or otherwise operated upon by the user using a private key associated with the public key within the respective validity period and wherein the plurality of identifiers are selected so that it is relatively easy to derive the

final identifier from one of the plurality of identifiers but relatively difficult to derive any of the plurality of identifiers from the final identifier, whereby the validity of the public key certificate within the validity period may be confirmed by operating upon the identifier provided by the user with the matter .

Preferably the identifiers are the product of the recursive application of a one-way hash function to a root. Said means may include a start date and the number and duration of renewal intervals, and the identifier and said means are preferably integrated into one or more extensions of the certificate, typically in at least one of: a private extension; a subject key id; a subject alternative name. The certificate is preferably constituted using X.509.

According to the invention in a fourth aspect, there is provided a public/private key cryptography framework in which matter operated upon using a private key of a public and private key pair is associated with a separate authentication which accompanies the matter, the authentication being related to a verifier, the verifier and public key being available to a recipient of the encrypted matter and accompanying authentication in the form of a public key certificate issued by a trusted third party and the arrangement being such that the verifier may be derived from the authentication but not vice versa to confirm the validity of the public key certificate.

The public key and verifier are preferably bound together in a public key certificate, and the certificate once generated may reside with a user of the private key and accompany any message signed with the private key and sent by the user.

The described embodiment discloses a public-key framework that avoids the certificate revocation in authentication, non-repudiation, and public-key encryption services. Either the certificate owner or his manager can control the validity of his certificate that has an extensible expiry date. The undeniable information that extends the certificate's expiry date is released by the certificate owner to the certificate verifiers directly. The certificate verifiers can determine whether the certificate is valid without contacting the CA or other trusted third parties.

According to the invention in a fifth aspect, there is provided a method of performing a public/private key operation comprising the steps of:

generating a public key and a root private key, the public key having a lifetime;

operating on the root private key to generate a respective plurality of interval private keys, the interval private keys being each usable together with the public key in a public/private key operation;

selecting a plurality of intervals within the lifetime and a respective plurality of identifiers, each interval being associated with an identifier and a said interval private key;

associating the public key with a final identifier;

the identifiers being such that it is relatively difficult to derive an identifier associated with a said interval from the final identifier but relatively easy to derive the final identifier from an identifier associated with a said interval ; and the interval keys being such that it is relatively easy to derive a later interval key from an earlier interval key but relatively difficult to derive an earlier interval key from a later one.

The public key and final identifier preferably form part of a public key certificate issued by a trusted third party.

The identifiers may be generated by recursive application of a one-way hash function to a root.

Within a current said interval, a user may send, to a third party, matter encrypted using the private key associated with the current interval, the identifier associated with the current interval being provided with the encrypted matter with the third party operating on the current interval identifier to derive the final identifier to confirm the validity of the private key and hash value in the current interval.

In the described embodiment a forward-secure digital signature scheme is disclosed in which transactions for validity periods both before and after the current period remain secure even if the private key for the current period is compromised.

BRIEF DESCRIPTION OF THE DRAWINGS

An embodiment of the invention will now be described, by way of example, with reference to accompanying drawings in which:

Figure 1 illustrates certificate expiry date extension using the embodiment of the invention;

Figure 2 illustrates public key framework;

Figure 3 illustrates a procedure for validating signatures using a forward secured digital signature scheme; and

Figure 4 illustrates how the embodiment of present invention may be integrated with the X.509 industry standard which defines the format of a public key certificate.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

Before describing the preferred embodiment of the invention, two standardized certificate revocation mechanisms in the IETF will be described namely:

- CRL – Certificate Revocation List, which provides periodic revocation information.

- OCSP – On-line Certificate Status Protocol, which provides timely revocation information.

Certificate Revocation List (CRL)

A CRL is a time-stamped list of serial numbers or other certificate identifiers for those certificates that have been revoked by a particular CA. The CRL is signed by the relevant CA and made freely available in a public repository. Updates should be issued at regular intervals, even if the list has not changed (thus enabling users possessing a CRL to check that it is the current one). The revoked certificates should remain on the list until their scheduled expiry date.

X.509 v2 CRL format profiled for Internet use in RFC2459 defines the required and optional fields. The required fields identify the CRL issuer, the algorithm used to sign the CRL, the date and time the CRL was issued, and the date and time by which the CA will issue the next CRL. Optional fields include lists of revoked certificates and CRL extensions. The revoked certificate list is optional to support the case where a CA has not revoked any unexpired certificates that it has issued.

Certificates revoked by the CA are uniquely identified by the certificate serial number. The date on which the revocation occurred is specified. Additional information may be supplied in CRL entry extensions which include

- Reason Code – identifies the reason for the certificate revocation.

- Hold Instruction Code – indicates the action to be taken after encountering a certificate that has been placed on hold.
- Invalidity Date – provides the date on which it is known or suspected that the private key was compromised or that the certificate otherwise became invalid.
- Certificate Issuer – identifies the certificate issuer associated with an entry in an indirect CRL.

CRL extensions provide methods for associating additional attributes with CRLs. The X.509 v2 CRL format also allows communities to define private extensions to carry information unique to those communities. Each extension in a CRL may be designated as critical or non- critical. A CRL validation must fail if it encounters a critical extension which it does not know how to process. However, an unrecognized non-critical extension may be ignored. The extensions used within Internet CRLs include

- Authority Key Identifier – identifies the public key corresponding to the private key used to sign a CRL.
- Issuer Alternative Name – allows additional identities to be associated with the issuer of the CRL.
- CRL Number – allows users to easily determine when a particular CRL supersedes another CRL.
- Delta CRL Indicator – contains the changes between the base CRL and the current CRL issued along with the delta-CRL.

- Issuing Distribution Point – identifies the CRL distribution point for a particular CRL.

Operational protocols that deliver CRLs to client systems could be built based on a variety of different means such as LDAP, HTTP, FTP, and X.500.

An advantage of this revocation method is that CRLs may be distributed by exactly the same means as certificates themselves, namely, via untrusted communications and server systems. One limitation of the CRL revocation method is that the time granularity of revocation is limited to the CRL issue period. For example, if a revocation is reported now, that revocation will not be reliably notified to certificate-using systems until the next periodic CRL is issued -- this may be up to one hour, one day, or one week depending on the frequency that the CA issues CRLs.

On-line Certificate Stating Protocol (OCSP)

The OCSP is described in 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 as a supplement to checking against a periodic CRL, it may be necessary to obtain timely information regarding the revocation status of a certificate. The OCSP enables applications to determine the state of an identified certificate. An OCSP client issues a status request to an OCSP responder and suspends acceptance of the certificate in question until the responder provides a

response. The OCSP responder must be one of the following parties.

- The CA who issued the certificate in question,
- A trusted responder whose public key is trusted by the requester, or
- A designated responder who holds a specially marked certificate issued directly by the CA, indicating that the responder may issue OCSP responses for that CA.

Upon receipt of a request, the OCSP responder either returns a definitive response, or produces an error message. All definitive response messages should be digitally signed. The response for each of the certificates in a request consists of

- target certificate identifier
- certificate status value
- response validity interval
- optional extensions

The certificate status value of a response is defined as follows. "Good" indicates a positive response to the status inquiry. "Revoked" indicates that the certificate has been revoked. "Unknown" indicates that the responder does not know about the certificate being requested.

There are two response validity intervals. "ThisUpdate" indicates the time at which the status being indicated is known to be correct. "NextUpdate" indicates the time at which newer information will be available about the certificate status. If "NextUpdate" is not

set, the responder is indicating that newer revocation information is available all the time.

Prior to accepting a signed response as valid, OCSP clients should confirm that

- The certificate identified in a received response corresponds to the one identified in the request.
- The signature on the response is valid.
- The identity of the signer matches the intended recipient of the request.
- The signer is currently authorized to sign the response.
- The time "ThisUpdate" is sufficiently recent.
- The time "NextUpate" is greater than the current time if it is set.

Both of the CRL and OCSP mechanisms require the certificate verifier to obtain the revocation information from a trusted third party to check the status of a public-key certificate. That could be a significant burden to applications relying on public-key cryptography for security.

A public-key framework being an embodiment of the invention, that avoids the certificate revocation in authentication, non-repudiation, and public-key encryption services will now be described. In this framework, a public-key certificate has an extensible expiry date under the control of the certificate owner or his manager. The validity of a certificate could be refreshed by the owner or his manager at a regular interval according to the security requirements of a given application. The certificate

verifiers can check the validity without retrieving the revocation information from the CA. This is based on the security building block of a "one-way hash chain".

A one-way hash chain is constructed by recursively applying an input string to a one-way hash function, which can be denoted as $H^i(r) = H(H^{i-1}(r))$ $(i = 1,2, \ldots)$ where $H^0(r) = r$ is the root of the hash chain. According to the feature of one-way hash function, if r is chosen randomly and the hash chain is kept secret, given $H^i(r)$, it is computationally infeasible for anyone except the originator of the hash chain to find the input $H^{i-1}(r)$.

As a basic security building block, a one-way hash chain has been used in some applications including one-time password authentication and micro-payment. The one-way hash chain generated in the above way could also be bound to a public-key certificate as disclosed in J. Zhou and K.Y. Lam. "*Securing digital signatures for non-repudiation*". Computer Communications, 22(8):710--716, Elsevier, May 1999., where the hash chain is included in a temporary public-key certificate that is generated by an ordinary user (and time-stamped by a trusted third party) so that the certificate expiry date is extensible under the control of that user. Another application has been proposed in S. Micali. "*Certificate revocation system*". US Patent 6292893, September 2001, where the CA updates the certificate status regularly by providing an intermediate hash value for a validity period to a CRL, with the final value of the hash chain being included in the public key certificate. However, in both these proposals a CRL is required to allow confirmation of validity. In Zhou and Lam, the temporary public key certificate includes matter signed by a private key associated with a further public key

which has a public key certificate issued by a trusted third party, the validity of which requires confirmation using a CRL in the normal way. In US 6292893, the intermediate hash chain value is provided in the CRL.

The public-key framework of the described embodiment avoids any trusted third parties' involvement in confirming the current validity of the public-key certificate once the certificate has been issued by the CA. The certificate will be initially not valid for use when issued by the CA. Only when the certificate owner releases a chained hash value, will it be valid for a limited time from the starting valid date. The certificate owner extends the expiry date of his certificate by releasing the chained hash values at a regular interval. If the certificate owner wants to invalidate his certificate, he can simply abandon the rest of unreleased hash values in the one-way hash chain. The interval for refreshing the certificate validity can be defined as short as desired by the certificate owner, so that there is no need of certificate revocation during that interval.

Generation of public-key certificate will now be described, considering first the situation that only the certificate owner can control the validity of his certificate.

A user U's public-key certificate with an extensible expiry date is generated in the following way.

1. U generates a pair of keys: private key SK_U and public key PK_U.
2. U defines the maximum life time of the key pair (SK_U, PK_U) as T, and the starting valid date as D. U also selects the time period for refreshing the

validity of the key pair as L. Suppose j = T/L is an integer. The refreshing points are denoted as $D_1 = D+L$, $D_2 = D+2*L$, ..., $D_j = D+j*L$. (The certificate's life time and refreshing points are illustrated in Figure 1.)

3. U selects a random number r as the root of a one-way hash chain, and generates the one-way hash chain $H^i(r) = H(H^{i-1}(r))$ (i = 1,2, ..., j).
4. U sends his public key PK_U, the starting valid date D, the last hash value $H^j(r)$, the hash chain length j, and the refreshing time period L, to the certification authority CA.
5. The CA authenticates U's request for a certificate in an out-of-band method.
6. Then, the CA generates a public-key certificate $CERT_U = SIGN_{CA}(U, PK_U, D, H^j(r), j, L)$. For simplicity, other less related information is omitted in $CERT_U$.
7. The CA sends $CERT_U$ to U.

Compared with an ordinary public-key certificate, $CERT_U$ contains extra data ($H^j(r)$, j, L) which will be used to control the validity of $CERT_U$.

Once $CERT_U$ is generated, this could either be delivered by the certificate owner U during a transaction, or be retrieved from a public directory maintained by a third party. At the starting valid date D, U will release $H^{j-1}(r)$ to initialize the validity of $CERT_U$, which then has an expiry date $D_1 = D+L$.

The use of the public-key certificate in digital signatures will now be described. Suppose the next refreshing point of $CERT_U$ is D_e. When U generates a digital signature with SK_U, he will attach $(H^i(r), i)$, where $i = j - (D_e-D)/L$, to the signature. The hash value release at each refreshing point is illustrated in Figure 1.

When a transacting party V wants to verify U's signatures, he first needs to check the validity of $CERT_U$. Suppose V holds the CA's public verification key, and the current time that V verifies $CERT_U$ is D_v. V needs to take the following steps to check the validity of $CERT_U$.

1. V verifies the CA's signature on $(U, PK_U, D, H^j(r), j, L)$. If true, V will be sure that U's public key is PK_U. The starting valid date is D, the maximum life time is $T = j*L$, the refreshing time period is L, and the last hash value in the one-way hash chain is $H^j(r)$.
2. V checks that $0 \leq i < j$, and $H^{j-i}(H^i(r)) = H^j(r)$. (where H^{j-i} is the hash function H applied recursively to $H^i(r)$ (j-i) times. If true, V believes that $H^i(r)$ is a valid hash value in the one-way hash chain ended with $H^j(r)$.
3. V checks that $D_v \leq D + (j-i)*L$. If true, V concludes that $CERT_U$ is valid now, and remains valid until $D_e = D + (j-i)*L$.

In such a way, U can control the validity of $CERT_U$ by releasing the corresponding $H^i(r)$ when generating digital signatures. V can check the validity of $CERT_U$ without retrieving the revocation information from the CA. Thus, certificate revocation can be avoided.

The random number r serving as the root of the one-way hash chain is critical to the security of the above public-key framework. The certificate owner U relies on it to control the expiry date of his public-key certificate $CERT_U$. In case that the private key SK_U is compromised, U only needs to destroy the hash chain root r. Then $CERT_U$ will expire shortly thereafter at the next refreshing point.

There might be a risk, however, if the hash chain root r and the private key SK_U are stored in the same computer system. If the system is broken, both r and SK_U will be compromised. Then, a hacker holding r and SK_U can always generate valid signatures by refreshing the validity of $CERT_U$ until its maximum life time T.

The following mechanisms may be used to protect the hash chain root r, these mechanisms being used by different users having different requirements, as shown in Fig. 2.

M1: Manually Input "r"

The most straightforward approach is to remember the hash chain root r and manually input r at the time of refreshing $CERT_U$. After the hash value needed for refreshing is generated, r will be erased from the computer system. That will minimize the possibility of compromise caused by system break-in. If SHA is used in the generation of one-way hash chain, the length of r could be as short as 20 bytes. As r must be random, it might

be a bit hard for an ordinary user to memorize a 20-byte random string. Such a technique would be appropriate for use with ordinary users U_1-U_n of Fig. 2.

M2: Protect "r" with Password

Alternatively, the hash chain root r could be protected with a password. A password, or its hash value then serving as a symmetric key for encryption. The encrypted r is stored in the local computer system. When $CERT_U$ needs to be refreshed, the password is input to get the decrypted r. r will be erased soon after use. If DES is used for encryption, the length of the password could be as short as 8 bytes. As a password is usually not truly random and an off-line dictionary attack is also possible, this mechanism for safeguarding r is not highly secure. Such a technique would be appropriate for use with ordinary users U_1-U_n of Fig. 2.

M3: Protect "r" Using Security Server

In this approach, the hash chain root r is be encrypted and stored locally while the secret key K for encryption is stored with a security server SS_K. Suppose U has registered his password at the security server. Then U and the security server can establish a secure and authenticated channel with password-based protocols (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), over which U could upload and download K safely.

When a refreshing date is approaching, U retrieves K from the security server, decrypts the cipher text of r, and calculates the corresponding chained hash value, after which r and K will be erased from the local system. U could optionally update the secret key K and upload the new key to the security server.

As the security server does not have the cipher text of r, it does not have the knowledge of r. Therefore, only the certificate owner has the control on the validity of his certificate.

This architecture is shown for users U_1_K - U_n_K in Fig. 2 and is especially good for corporate clients in which a security server will centrally manage the secret key for each client. As the security server only needs to serve its internal clients, its connection to the Internet could be tightly controlled to minimize the risk of compromise. As the corporate clients rely on the security server in the extension of expiry date of their certificates, it is important to ensure that the security server is well configured to function properly.

As just described, the hash chain root r is generated by the certificate owner U. Therefore, U has the full control of the validity of $CERT_U$ until $CERT_U$ reaches its maximum life time T. This can only address the need of certificate revocation caused by the compromise of private keys. However, a public-key certificate may have to be

revoked by the manager of the certificate owner for other reasons such as termination of job or change of name.

To address this problem the hash chain root may be generated by a security server SS_r, which will be administered by the manager of corporate users. This is shown for users U_1_r - U_n_r in Fig. 2. The process of certificate generation will be changed as follows.

1. U generates a pair of keys: private key SK_U and public key PK_U.
2. Suppose U has registered his password at the security server. Then U could sends the request of a certificate for corporate use, together with his public key PK_U, to the security server over a secure and authenticated channel with password-based protocols (as disclosed in [Bellovin and Merrit], above).
3. According to the corporate security policy, the security server defines the maximum life time of U's certificate as T, and the starting valid date as D. It also selects the time period for refreshing the validity of the certificate as L.
4. Suppose j = T/L is an integer. The security server selects a random number r as the root of a one-way hash chain, and generates the one-way hash chain $H^i(r) = H(H^{i-1}(r))$ (i = 1,2, …, j).
5. The security server sends U's public key PK_U, the starting valid date D, the last hash value $H^j(r)$, the hash chain length j, and the refreshing time period L, to the certification authority CA.
6. The CA authenticates the security server's request for generating a public-key certificate in an out-of-band method. This will prevent U from requesting a public-key certificate for corporate use without authorization.

7. The CA may further challenge U in an out-of-band method to ensure U holds the corresponding private key. This will prevent the security server from requesting a public-key certificate in the name of U who is unaware of it.
8. Then, the CA generates a public-key certificate $CERT_U = SIGN_{CA}(U, PK_U, D, H^j(r), j, L)$.
9. The CA sends $CERT_U$ to the security server.
10. The security server forwards $CERT_U$ to U.

When a refreshing date is approaching, the security server distributes the corresponding hash value to U. Suppose the next refreshing date of $CERT_U$ is D_e. The security server calculates $H^i(r)$ from r where $i = j - (D_e-D)/L$, and distributes $(H^i(r),i)$ to U. No protection is needed in distribution. U can easily verify $H^i(r)$ is the hash value to be released on the date D_e by checking whether $j-i = (D_e-D)/L$ and $H^{j-i}(H^i(r)) = H^j(r)$.

If the security server wants to revoke U's certificate for some reason instructed by the corporate management, the security server can do so by stopping release of U's hash values, thus $CERT_U$ will expire soon at the next refreshing point. If U suspects a compromise of his private key, U could send a request to the security server for stopping distribution of the next hash value.

The security server's role in this embodiment is fundamentally different from the CA's role in certificate revocation.

- The security server only needs to communicate with the certificate owners while the CA needs to make the revocation information available to any potential certificate verifier.
- The chained hash values released by the security server need no protection while the authenticity and integrity of the revocation information released by the CA need to be protected.

In the described embodiment, a user or his manager can control the validity of his public-key certificate and others can check the validity of such a certificate without retrieving the revocation information from the CA. This improves the efficiency in authentication, non-repudiation, and public-key encryption services. Some applications will now be described:

Authentication

Authentication is a basic security service that provides protection against masquerade. This consists of entity authentication that verifies a claimed identity, and data origin authentication that verifies the source and integrity of a message. Digital signature is an important security mechanism to support authentication. In authentication services, the signature verifier will use the signer's public key to verify the signature. More importantly, the verifier should check whether the signer's public key is valid at the time of verification.

The signer's public key is usually bound to the signer's identity in a public-key certificate issued by the CA. If the public-key certificate is constructed with an extensible expiry date as described above, it becomes very efficient to check the validity of the certificate.

Suppose the signer U generates a digital signature σ using his private key SK_U, and the next refreshing date of $CERT_U$ is D_e. U sends σ and $(H^i(r), i)$, where $i = j - (D_e-D)/L$, to a transacting party V for authentication.

As the validity of $CERT_U$ is decided by the hash value released by U (before $CERT_U$ reaches the maximum life time T defined in $CERT_U$), V only needs to check whether $H^i(r)$ is the hash value that extends the expiry date of $CERT_U$ to D_e. If so, V can use $CERT_U$ safely before its current expiry date D_e to verify U's signatures.

Non-Repudiation

Non-repudiation is another basic security service that provides protection against false denial of having been involved in an electronic transaction. It creates, collects, validates, and maintains cryptographic evidence in order to support the settlement of possible disputes. Digital signature is one of the most convincing types of cryptographic evidence. However, there are stronger security requirements when digital signatures are used for non-repudiation purpose.

As there exists the possibility of private key compromise and thus the forgery of digital signatures, there is a need for the revocation of the corresponding public-key certificate. Then, signatures generated after certificate revocation will be regarded as invalid. On the other hand, all signatures generated before certificate revocation should remain valid, thus can be used in the settlement of disputes that may arise at a time well after the end of a transaction.

With the described embodiment, two approaches to support a non-repudiation service, a trusted time-stamping service and a forward-secure digital signature scheme are provided.

For trusted time-stamping, suppose the signer U generates a digital signature σ using his private key SK_U. Different from the authentication service, U should get a trusted time-stamp D_g on σ, denoted as $SIGN_{TSA}(D_g, \sigma)$, from a time-stamping authority (TSA). Suppose the next refreshing date of $CERT_U$ is D_e. U sends $SIGN_{TSA}(D_g, \sigma)$ and $(H^i(r), i)$, where $i = j - (D_e-D)/L$, to a transacting party V as non-repudiation evidence.

As discussed above, with $(H^i(r), i)$, it is easy for V to check whether $CERT_U$ is valid until the date D_e. Suppose V holds the TSA's public verification key. Then, V could use $SIGN_{TSA}(D_g, \sigma)$ to further check whether σ is U's signature generated before D_e, i.e., $D_g \leq D_e$. If so, V could accept $SIGN_{TSA}(D_g, \sigma)$, $(H^i(r), i)$, $CERT_U$ safely as valid non-repudiation evidence. They could be used to prove to any third party that U's signature

σ was generated while the corresponding certificate $CERT_U$ was still valid, and thus undeniable.

U could further extend the expiry date of $CERT_U$ to D_e+L by releasing $(H^{i-1}(r), i-1)$ when $D_e < D_j$, or let $CERT_U$ to expire at D_e by destroying the unreleased one-way hash chain. In either case, it will not affect the validity status of σ.

This approach avoids the CA's involvement for certificate revocation. However, it still relies on the trusted third party TSA to provide the time-stamping service.

A forward-secure digital signature approach is more efficient as it avoids the involvement of both the CA and the TSA.

Forward-secure digital signature schemes (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; 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), which update the private signing key at regular intervals while the public key is fixed throughout the lifetime of the certificate preserve the validity of past signatures without using a trusted time-stamping service even if the current private key has been compromised. However, such schemes cannot address the issue on signature forgery with the subsequent signing keys

derived from the current compromised one.

With reference to Figure 3, a forward-secure digital signature scheme using the embodiment of the invention will be described. The scheme uses four functional components: FWKG for key generation, FWUPD for private key update, FWSIGN for signing, and FWVER for signature verification, and details of these components may be found in the prior art mentioned above. FWKG generates the public key PK and the root private key SK_0. The lifetime T is divided into j intervals as before, each interval k of length L being associated with a value of the hash chain $H^k(r)$, PK will be certificated by the CA, together with $(H^j(r), j, L)$. Each interval I has also associated therewith an interval private key SK_k .

FWUPD updates the private key at each refreshing point as follows: SK_1 = FWUPD(SK_0) at point D, SK_2 = FWUPD(SK_1) at point D_1, ..., SK_j = FWUPD(SK_{j-1}) at point D_{j-1} as shown in Figure 3. FWUPD is a one-way function in the sense that no one can derive SK_{j-1} from SK_j but computing SK_j from SK_{j-1} is easy. Once the private key is updated, the old private key must be erased, thus nobody can derive any past private keys from the current one.

According to the time of signature generation, the digital signatures generated under this mechanism can be classified into three types as illustrated in Figure 3: signatures of past time periods, signatures of current time period, and signatures of subsequent time-periods.

Suppose the current private key is SK_c. The signer U must attach the current time D_g to the message M that is to be signed, denoted as FWSIGN(M, D_g, SK_c). Suppose the next refreshing point of $CERT_U$ is D_e. U will attach ($H^i(r)$, i), where $i = j - (D_e-D)/L$, to the above signature.

With FWSIGN(M, D_g, SK_c), ($H^i(r)$, i), and $CERT_U$, the verifier V is able to check , as before, whether $CERT_U$ is valid until D_e and $D_g \leq D_e$. If so, V believes that FWSIGN(M, D_g, SK_c) was generated at the time that $CERT_U$ was still valid. V further checks U's signature FWSIGN(M, D_g, SK_c) which requires D_g as an input of FWVER. The verification will fail if $D_g \leq D_{e-1}$ or $D_g > D_e$. That means the private key of the current time period can only be used to generate valid signatures of that time period.

Put another way, the interval private keys SK are arranged such that an earlier interval key cannot be derived from a later one and the hash values are arranged so that a later interval hash value cannot be derived from an earlier one. Used together, the interval private key and interval hash value can relate to one interval only and since it is not possible to derive both a later hash value and a later private key or both an earlier hash value and an earlier private key from the current hash value and current interval private key if these are compromised, transactions in all past and future intervals remain secure.

If FWSIGN(M, D_g, SK_c), (H^i(r), i), and $CERT_U$ pass the above verification, they could serve as valid non-repudiation evidence.

If U suspects the compromise of his current private key SK_c, he could let $CERT_U$ to expire at D_e by destroying the unreleased one-way hash chain. The attacker cannot derive past private keys from the compromised current one, and thus cannot forge valid signatures of past time periods. Although the attacker could derive the subsequent private keys from the compromised current one, he cannot use them to forge valid signatures of subsequent time periods since $CERT_U$ has expired. The risk of signature forgery in the current time period could be well controlled if the refreshing time period L is carefully defined according to the individual's security requirement, which is more flexible than the CRL-based revocation mechanism whose update interval must be acceptable to all certificate users.

Public Key Encryption

In the described embodiment, the procedure for public-key encryption is slightly different. When a user signs a message, he can attach (H^i(r), i) to his signature for verifying $CERT_U$. When a user encrypts a message, he has to first obtain such information from the intended recipient.

In the normal PKI, the user doing public-key encryption first needs to obtain the revocation information from the issuing CA in order to check the validity of the intended recipient's certificate. In the described embodiment, the CA will not provide any revocation information once the certificate has been issued. Instead, the certificate will have an extensible expiry date which is under the control of the certificate owner or his manager. The user doing encryption has to obtain the information of current expiry date from the certificate owner.

The process of public-key encryption could be described as follows.

1. The user doing encryption sends an enquiry to the intended recipient about the status of his public-key certificate $CERT_U$.
2. If $CERT_U$ has expired, the intended recipient replies with an expiry status. Otherwise, the intended recipient provides $(H^i(r), i)$, which indicates $CERT_U$ will be valid until the refreshing point $D_{j-i} = D + (j-i)*L$. No protection is needed in distribution of $(H^i(r), i)$.
3. The user can check whether $H^i(r)$ is the matching hash value that extends the expiry date of $CERT_U$ to D_{j-i}. If the current time is earlier than D_{j-i}, the user will be sure that $CERT_U$ is still valid, thus the corresponding public key can be used safely for encryption.
4. The user sends the encrypted message to the intended recipient.

X.509 is an industry standard which defines the format of a public-key certificate and this is illustrated in Fig. 4. The applicability of our new public-key framework is closely

related to the issue on whether the extra data for an extensible expiry date can be integrated into the existing X.509 certificate format with the minimum impact on interoperability.

The X.509 v3 certificate basic syntax includes version number, serial number, issuer's signature algorithm identifier, issuer name, validity period, subject name, subject public key information, issuer unique id, subject unique id, and extensions. The most flexible part of a X.509 v3 certificate is its "extensions" field. Each extension contains an extension id and the extension value, and may be designated as critical or non-critical.

The extensions defined for X.509 v3 certificates provide methods for associating additional attributes with users or public keys and for managing the certification hierarchy. The X.509 v3 certificate format also allows communities to define private extensions to carry information unique to those communities. Current standard extensions are authority key id, subject key id, key usage, certificate policies, subject alternative name, issuer alternative name, basic constraints, name constraints, policy constraints, and extended key usage.

To support the extensible expiry date, the data $(H^j(r), j, L)$ should be included in the certificate. From the structure of a X.509 v3 certificate, there are three possible extensions into which the data $(H^j(r), j, L)$ could be integrated.

The first option is "private extension". This extension could be defined locally, and allows X.509 v3 certificates to include more attributes. A new private extension could be defined that specifies the data format as $(H^j(r), j, L)$ to support the extensible expiry date.

The second option is "subject key id". The subject key id extension provides a means of identifying certificates that contain a particular public key. As the hash chain root r is randomly selected when generating a public-key certificate, the data $(H^j(r), j, L)$ could be regarded as a subject key identifier that uniquely links to the public key.

The third option is "subject alternative name". The subject alternative name extension allows additional identities to be bound to the subject of the certificate. The data $(H^j(r), j, L)$ could be regarded as an additional identity bound to the subject in the form of locally defined other name.

Having now fully described the invention, it should be apparent to one of ordinary skill in the art that many modifications can be made hereto without departing from the scope as claimed.

CLAIMS

1. A method of performing a public/private key operation comprising the steps of:

(a) generating a private and public key pair, the public key having a lifetime;

(b) selecting a plurality of intervals within the lifetime and a plurality of respective identifiers, each interval being associated with a said identifier; and

(c) having a trusted third party generate a public key certificate which includes the public key and a final identifier, the identifiers being selected so that it is relatively difficult to derive an identifier associated with a said interval from the final identifier but relatively easy to derive the final identifier from an identifier associated with a said interval, a said identifier being provided by a user with any matter signed or otherwise operated upon using the private key in the associated interval to confirm the validity of the certificate in that interval.

2. A method as claimed in claim 1 wherein a user of the private key holds the certificate.

3. A method as claimed in claim 1 wherein the certificate is held in a public location.

4. A method as claimed in any one of the preceding claims wherein one or more identifiers or means from which the identifiers may be generated is/are stored on a remote server.

5. A method as claimed in claim 4 wherein, at the time a user wishes to encrypt matter using the private key, the user obtains the identifier for the current period from the server.

6. A method as claimed in claim 4 wherein at the time a user wishes to encrypt matter using the private key, the user obtains said means from the server and generates the identifier using the means.

7. A method as claimed in claim 5 or claim 6 wherein the user and server communicate via a secure channel.

8. A method as claimed in any one of the preceding claims wherein the identifiers are generated by recursive application of a one-way hash function to a root.

9. A method as claimed in any one of the preceding claims wherein a third party operates on the current interval identifier to derive the final identifier to confirm the validity of the certificate in the current interval.

10. A method as claimed in any one of the preceding claims wherein the user obtains a time stamped version of the encrypted matter to be accompanied by the identifier.

11. A method as claimed in claim 10 wherein the third party operates on the current interval identifier to derive the lifetime identifier to confirm the

validity of the certificate in the current interval and that the timestamp is within the current interval.

12. A method as claimed in any one of the preceding claims wherein the private key has validity within a time period and the user sends a time of encryption with the encrypted matter.

13. A method as claimed in claim 12 wherein third party operates on the current interval identifier to derive the final identifier to confirm the validity of the certificate in the current interval and that the time of encryption is within the time period.

14. A method as claimed in any one of the preceding claims wherein, each interval has associated therewith an interval private key, the interval private keys being each usable together with the public key in a public/private key encryption/decryption operation, the interval keys being such that it is relatively easy to derive a later interval key from an earlier interval key but relatively difficult to derive an earlier interval key from a later one.

15. A method as claimed in any one of the preceding claims wherein a third party wishing to confirm the validity of the certificate requests from the user the identifier of the current time period and determines if the final identifier of the certificate can be derived from the current identifier.

16. A method as claimed in claim 15 wherein the third party uses the public key verified by the certificate to encrypt matter to send to the user.

17. A method of conducting a public key transaction comprising the steps of:

(a) generating a public and private key pair, the private key being held by the user and the public key being accessible to a party to the transaction and having associated therewith a public key certificate issued by a trusted third party;

(b) sending to the party matter signed or otherwise operated upon using the private key by the user together with a matter identifier;

(c) the certificate including a certificate identifier , the identifiers being such that it is relatively easy to derive the certificate identifier from the matter identifier but relatively difficult to derive the matter identifier from the certificate identifier; and

(d) operating on the matter identifier provided by the user to derive the certificate identifier to confirm the validity of the certificate.

18. A public key certificate issued by a trusted third party comprising a public key, a final identifier and means defining validity periods between a start date and a finish date, a plurality of identifiers not forming part of the certificate each being associated with a said validity period and arranged to accompany matter signed or otherwise operated upon by the user using a private key associated with the public key within the respective validity period and wherein the plurality of identifiers are selected so that it is relatively easy to derive the final identifier from one of the plurality of identifiers but relatively difficult to derive any of the plurality of identifiers from the final identifier, whereby the validity of the public key certificate

within the validity period may be confirmed by operating upon the identifier provided by the user with the matter .

19. A certificate as claimed in claim 18 wherein the identifiers are the product of the recursive application of a one-way hash function to a root.

20. A certificate as claimed in claim 18 or claim 19 wherein said means include a start date and the number and duration of renewal intervals.

21. A certificate as claimed in any one of claims 18 to 20 wherein the identifier and said means are integrated into one or more extensions of the certificate.

22. A certificate as claimed in claim 22 wherein said one or more extensions comprise at least one of:

(a) a private extension;

(b) a subject key id;

(c) a subject alternative name.

23. A certificate as claimed in any one of claims 19 to 23 wherein the certificate is constituted using X.509.

24. A public/private key cryptography framework in which matter operated upon using a private key of a public and private key pair is associated with a separate authentication which accompanies the matter, the authentication being related to a verifier, the verifier and public key being available to a recipient of the encrypted matter and accompanying authentication in the form of a public key certificate issued by a trusted third party and the arrangement being such that the verifier may be derived from the

authentication but not vice versa to confirm the validity of the public key certificate.

25. A framework as claimed in claim 24 wherein the public key and verifier are bound together in a public key certificate.

26. A framework as claimed in claim 24 or claim 25 wherein the certificate once generated resides with a user of the private key and accompanies any encrypted message sent by the user.

27. A method of performing a public/private key operation comprising the steps of:

(a) generating a public key and a root private key, the public key having a lifetime;

(b) operating on the root private key to generate a respective plurality of interval private keys, the interval private keys being each usable together with the public key in a public/private key operation;

(c) selecting a plurality of intervals within the lifetime and a respective plurality of identifiers, each interval being associated with an identifier and a said interval private key;

(d) associating the public key with a final identifier;

(e) the identifiers being such that it is relatively difficult to derive an identifier associated with a said interval from the final identifier but relatively easy to derive the final identifier from an identifier associated with a said interval ; and the interval keys being such that it is relatively easy to derive a later interval key from an earlier

interval key but relatively difficult to derive an earlier interval key from a later one.

28. A method as claimed in claim 27 wherein the public key and final identifier form part of a public key certificate issued by a trusted third party.

29. A method as claimed in claim 27 or claim 28 wherein the identifiers are generated by recursive application of a one-way hash function to a root.

30. A method as claimed in any one of claims 27 to 29 wherein, within a current said interval, a user sends, to a third party, matter encrypted using the private key associated with the current interval, the identifier associated with the current interval is provided with the encrypted matter.

31. A method as claimed in claim 30 wherein the third party operates on the current interval identifier to derive the final identifier to confirm the validity of the private key and hash value in the current interval.