Efficient one-time proxy signatures

M. Mehta and L. Harn

Abstract: Proxy signatures allow a signer to delegate signing ability to a proxy signer. Many schemes have been proposed for proxy signatures under typical security requirements. The authors propose a proxy signature model with extended security requirements. Based on Shamir's online/offline signature scheme, a proxy signature scheme is proposed for the model. In addition to the typical requirements, the proposed scheme satisfies other very important security requirements. It is shown that the scheme can provide signature indistinguishability, restrict the proxy signing power, provide signature unlinkability, resolve internal disputes, and is more efficient.

1 Introduction

Proxy signature is a signature scheme where an original signer delegates his/her signing ability to a proxy signer to enable the proxy signer to generate signature on behalf of the original signer. Proxy signature schemes find applications in a variety of computing environments such as mobile agents for e-commerce [1], grid computing [2], global distribution networks [3], mobile communications [4] etc. There are four main entities involved in proxy signature schemes: original signer (O), proxy signer (P), verifier (V), and judge (J) for dispute resolution.

Mambo *et al.* [5] first introduced the concept of proxy signatures in 1996. The signature schemes were classified in three main types: full delegation, partial delegation, and delegation by warrant. In full delegation, O gives its private key to P. In this type of delegation, the signatures generated by P cannot be distinguished from those generated by O. However, dispute between O and P on a signature cannot be resolved. In partial delegation, O generates a proxy signature key from its private key and gives it to P. P uses this key to sign on behalf of O. In this case, the signatures from O and P are distinguishable, and so the scheme may not be suitable for situations where identity of P needs to be kept secret. In delegation by warrant, O issues a warrant containing a message part and signature key to P. The signatures by P consist of the signature using the signature key in the warrant and the warrant itself. The identity of P may or may not be revealed depending on the construction of warrant.

We introduce a delegation by warrant model for proxy signatures with extended (and different to some extent) set of security requirements as compared to those used in most existing proposals. In our model, the delegation of signing ability is an internal affair between O and P. Any other entity, including V, is considered to be an external entity for the delegation operation. In other words, V is able to verify the signature on given message like a regular signature verification using only O's public key. However, the actual identity of the signer (O or P) is hidden from V. In fact, the delegation of signing ability is considered as a confidential operation where V may not even know if there exists a separate P. On the other hand, in case of dispute between V and O or even between P and O, the model is able to resolve the dispute. Further, it may be necessary for O to delegate the signing ability in a restricted manner. Our model provides one-time proxy signature. Since no more than one proxy signature can be securely generated per delegation warrant, the ability of P can be restricted by issuing required number of delegation warrants. Other constraints such as validity period, liability amount, or even a specific application can be included in the warrant to further restrict P's signing ability.

In the following we list the security requirements for our model. Some of these requirements have been previously listed by Mambo *et al.* [5] and Lee *et al.* [1] for their models.

- *Strong unforgeability:* Only a designated P can generate a valid proxy signature; any other party, including O, cannot generate a valid proxy signature to impersonate a particular proxy signer. However, O can designate itself as a selfproxy.
- *Verifiability:* Validity of a proxy signature as well as the delegation of O on a given message can be verified using the public key of O and other public parameters.
- *Signature indistinguishability:* Regardless of identity of the actual signer, selfproxy O or a separate P, V can verify the validity of signature using a single regular signature verification algorithm.
- *Signature unlinkability:* (*a*) Given a valid signature, no third party, including V, can link a signature with the identity of the actual signer (selfproxy O or a separate P). (*b*) Given a pool of valid signatures, no third party, including the V, can link two signatures from the same actual signer.
- *External undeniability:* For a signature presented by an external entity, O cannot falsely deny that the signature was not generated by either O or P.
- *Internal undeniability:* For a valid signature, neither O nor P can falsely deny the generation of the signature.
- *One-timeness:* For every delegation operation, no more than one valid proxy signature can be securely generated by P.

IEE Proceedings online no. 20045251

doi:10.1049/ip-com:20045251

Paper first received 14th October and in revised form 8th December 2004

The authors are with the School of Computing and Engineering, University of Missouri - Kansas City, 5100 Rockhill Road, Kansas City, Missouri, 64110, USA

E-mail: mmmef7@umkc.edu

Based on the model introduced, we propose a proxy signature scheme that meets all the above-mentioned security requirements. Since the signatures generated by the original self-proxy signer and P are indistinguishable, our scheme provides privacy to P. Further, our scheme can resolve any repudiation dispute between V and O as well as that between O and P. Furthermore, our scheme is a one-time signature scheme and therefore can be used to restrict the number of proxy signatures P can generate. Moreover, the warrant in our scheme can include additional parameters such as expiration time, liability amount etc. to further restrict the ability of P to an intended application.

Most proxy signature schemes based on delegation by warrant proposed in literature use consecutive execution of signature schemes for delegation operation (a signed certificate containing proxy-key) and the actual proxy signature. This is inefficient in that to generate or verify a proxy signature, the effort spent is twice the effort spent for a regular signature operation. Our scheme is based on online/offline signature scheme proposed by Shamir and Tauman [6]. In our scheme the delegation operation between O and P is done in offline phase and the proxy signature is generated by P in online phase. We show that our scheme, rather than adopting consecutive operations, splits the computational efforts between online and offline phase such that the overall computational effort is reduced. Specifically, our scheme only requires P to solve a linear equation in online phase to generate a valid proxy signature.

2 Related work

Following Mambo *et al.* [5], many variations of proxy signature schemes have been proposed based on different features. Kim *et al.* [7] included a warrant in proxy signature. Owing to the warrant, O can restrict the message type that can be signed. However, the identity of P included in the warrant renders the scheme unsuitable for applications where privacy of P is important. In the scheme proposed by Shum and Wei [8], the real identity of P is hidden behind an alias. Although an external entity cannot know the actual identity of the signer, multiple signatures from the same signer can be linked using the alias. Also, Sun and Hsieh [9] showed that Shum and Wei's scheme is insecure against O's forgery. Lee *et al.* [10] identified several security weaknesses in the schemes originally proposed by Mambo *et al.* and introduced the concept of strong and weak proxy signatures according to undeniable property. Strong proxy signature represents both O's and P's signatures, while weak proxy signature represent only O's signature. The proposed strong proxy signatures provided nonrepudiation for both O and P. However, in this scheme there is no restriction on P's signing ability and therefore P could generate more than one valid signatures for more than one message using the proxy key. Kim *et al.* [11] introduced a one-time proxy signature concept to put restrictions on signing ability of P. In this scheme, like our model, each proxy key can be used to sign only one message. Other one-time signature schemes have been proposed in [12–13]. Hwang *et al.* [14] proposed a *c-times* digital signature scheme which restricts the number of messages that can be signed to a variable c. If more than c-messages are signed, the Lagrange interpolation method can be used with the c-degree polynomial to compute signer's secret signing key. Choi *et al.* [15] also proposed a proxy signature scheme based on Schnorr's scheme to restrict the number of signatures P can generate. This scheme is mainly proposed to counter key exposure problem. Further, the scheme is not presented with a comprehensive analysis of security requirements it meets and therefore it is difficult to analyse security of this scheme. Many other schemes with different variations have been proposed in [4, 16–23].

To the best of our knowledge, only the scheme proposed by Kim *et al.* [11] satisfies all the security requirements recompiled by Bamasak and Zhang in [24]. Kim *et al.*'s scheme is tailored for secret computations for mobile agents using fail-stop signatures. This scheme is more complex and inefficient as compared to our proposed scheme. All the schemes discussed above can be classified into different categories on the basis of cryptography in use, nonrepudiation properties, protection of P's identity, proxy signing restrictions, etc.

3 Review of Shamir and Tauman's scheme

We first review the trapdoor hash families presented by Shamir and Tauman [6] and then review the online/offline signature scheme based on the trapdoor hash families.

3.1 Trapdoor hash families

A trapdoor hash family, introduced in [25] and formally defined in [6], consists of a pair $(\mathcal{I}, \mathcal{H})$, where $\mathcal{I}$ is a probabilistic polynomial-time key generation algorithm, and $\mathcal{H}$ is a family of randomised hash family: (For a comprehensive study on trapdoor related schemes, refer to [26]). $\mathcal{I}$ generates a pair (HK, TK), where HK is a hash key, and TK is its associated trapdoor key. A trapdoor hash function in $\mathcal{H}$ is a hash function with a trapdoor secret. It is represented as $h_{HK}(m, s)$ where HK is a public key, TK a private key, m is a message, and s is an auxiliary random number.

A trapdoor hash function must satisfy the following three requirements [6]:

- *Efficiency:* Given a hash key HK and a pair (m,s), $h_{HK}(m, s)$ is computable in polynomial time.
- *Collision resistance:* There is no probabilistic polynomial-time algorithm $\mathcal{A}$ that on input HK outputs, with a probability which is not negligible, two pairs (m_1, s_1), (m_2, s_2) such that $m_1 \neq m_2$ and $h_{HK}(m_1, s_1) = h_{HK}(m_2, s_2)$
- *Trapdoor collision:* There exists a probabilistic polynomial time algorithm that given a pair $(HK, TK) \leftarrow \mathcal{I}$, a pair (m_1, s_1) and an additional message m_2 outputs a value s_2 such that
  - $h_{HK}(m_1, s_1) = h_{HK}(m_2, s_2)$.
  - If s_1 is uniformly distributed in $\mathcal{S}$ then the distribution of s_2 is computationally indistinguishable from uniform in $\mathcal{S}$.

3.1.1 Krawczyk and Rabin's discrete logarithm-based trapdoor hash family:

Randomly select a large prime q. Randomly choose a safe prime p (i.e. a prime p such that $q = (p-1)/2$ is prime) and an element g of order q. Choose a random element x and compute $y = g^x \mod p$. The public hash key HK is (p, g, y) and the private trapdoor key TK is x. The trapdoor hash function $h_{HK}(m, s)$ is defined as follows

$$h_{HK}(m, s) \stackrel{\text{def}}{=} g^m y^s \mod p$$

To show that the $h_{HK}(m, s)$ is a trapdoor hash function under the discrete logarithm (DL) assumption, one needs to show that it fulfills the three main properties of a trapdoor hash function listed. A lemma and its formal proof asserting

that $h_{HK}(m,s)$ is a DL-based trapdoor hash function can be found in [6].

For a given a pair $(HK, TK) \leftarrow \mathcal{I}$, a pair (m_1, s_1), and additional message m_2, trapdoor collision can be found in the following way. Since the requirement is $h_{HK}(m_1,s_1) = h_{HK}(m_2,s_2)$, the condition to be satisfied is, $g^{m_1} y^{s_1} = g^{m_2} y^{s_2} \bmod p$. That is, $g^{m_1+x\cdot s_1} = g^{m_2+x\cdot s_2} \bmod p$. This gives a linear equation $m_1 + xs_1 = m_2 + xs_2 \bmod q$. Since the only unknown parameter is s_2, a collision can be found by solving the linear equation

$$s_2 = x^{-1}(m_1 - m_2) + s_1 (\bmod q)$$

3.1.2 Shamir and Tauman's factoring-based trapdoor hash function: Choose at random two safe primes p and q (i.e. primes such that $p' = (p-1)/2$ and $q' = (q-1)/2$ are primes) and compute $n = pq$. Choose at random an element g of order $\lambda(n)$, where $\lambda(n) = lcm(p-1, q-1) = 2p'q'$. The public hash key HK is (n, g) and the private trapdoor key TK is (p, q). The trapdoor hash function $h_{HK}(m,s)$ is defined as follows:

$$h_{HK}(m,s) \stackrel{\text{def}}{=} g^{m||s} \bmod n$$

where $||$ denotes concatenation. To show that the $h_{HK}(m,s)$ is a trapdoor hash function under the factoring assumption, one needs to show that it fulfills the three main properties of a trapdoor hash function listed. A lemma and its formal proof asserting that $h_{HK}(m,s)$ is a factoring based trapdoor hash function can be found in [6].

For a given a pair $(HK, TK) \leftarrow \mathcal{I}$, a pair (m_1, s_1), and additional message m_2, trapdoor collision can be found in the following way. Since the requirement is $h_{HK}(m_1,s_1) = h_{HK}(m_2,s_2)$, the condition to be satisfied is, $g^{m_1||s_1} = g^{m_2||s_2} \bmod n$. That is, we want to find s_2 such that $2^k m_1 + s_1 = 2^k m_2 + s_2 \bmod \lambda(n)$, where k is the size of the auxiliary parameter s. Given the trapdoor key $TK = (p, q)$, $\lambda(n)$ can be computed in polynomial time and hence s_2 can be computed in polynomial time by solving the linear equation

$$s_2 = 2^k(m_1 - m_2) + s_1 (\bmod \lambda(n))$$

3.2 Signature scheme

In [6] Shamir and Tauman introduced a hash–sign–switch paradigm in which any regular digital signature scheme combined with a trapdoor hash family in $(\mathcal{I}, \mathcal{H})$ can be converted into a online/offline signature scheme. Basically, in the offline phase, a signer generates a hash value to commit to an arbitrarily selected message. In the online phase, given a message, the signer finds a collision of the trapdoor hash to the previously calculated hash value. The collision point and the signature generated in offline phase can be presented as the signature for message generated in online phase.

Let $h_{HK}(m,s)$ be a trapdoor hash function, HK be the hash key, TK be the associated trapdoor key, VK be the verification key, and SK be the signing key for any regular digital signature scheme. The following describes the online/offline signature scheme:

- *Key generation algorithm GEN*: Generate a pair (SK, VK) using a public-key generation algorithm and a pair (TK, HK) using algorithm $\mathcal{I}$. The signing key is (SK, TK, HK) and the verification key is (VK, HK).
- *Signing algorithm SIGN*: Given a signing key (SK, TK, HK) the signing algorithm operates as follows:
  - *Offline phase*: The signer randomly selects (m,s) and computes $h_{HK}(m,s)$, then uses SK to sign $h_{HK}(m,s)$ to obtain $\langle S_{SK}(h_{HK}(m,s))\rangle$. The signer stores m, s and $S_{SK}(h_{HK}(m,s))$ and can also store $h_{HK}(m,s)$ to avoid its re-computation during the on-line phase.
  - *Online phase*: Given a message m', the signer finds a collision of the trapdoor hash such that $h_{HK}(m',s') = h_{HK}(m,s)$. The signature of message m', is $\langle S_{SK}(h_{HK}(m,s)), s', h_{HK}(m,s)\rangle$.
- *Verification algorithm VERF*: First verify $\langle S_{SK}(h_{HK}(m,s))\rangle$ using VK and $h_{HK}(m,s)$, and then compute $h_{HK}(m',s')$ to verify if $h_{HK}(m,s) = h_{HK}(m',s')$.

4 Our proxy signature scheme

We modify the online/offline scheme reviewed in Section 3 to use it as an efficient proxy signature scheme.

4.1 Signature scheme

Unlike regular signature schemes, proxy signature schemes have two signers: original signer O and proxy signer P. In our proxy signature scheme, the key generation operation, delegation operation, and partial signing operation are performed offline; actual signature generation and verification are performed online.

Let $h_{HK}(m,s)$ be a trapdoor hash function, HK be the hash key, TK be the associated trapdoor key, (SK_O, VK_O) and (SK_P, VK_P) be the pairs of signing and verification keys for any regular digital signature scheme for O and P, respectively. The following describes our proxy signature scheme:

- *Key generation algorithm PRO–GEN*: O and P generate corresponding pairs (SK_O, VK_O) and (SK_P, VK_P) using a public-key generation algorithm. P generates a pair (TK, HK) using algorithm $\mathcal{I}$. The proxy signing key is (SK_P, TK, HK) and the verification key is (VK_O, HK).
- *Delegation algorithm PRO–DEL*: P randomly selects (m,s) and computes $h_{HK}(m,s)$. P then uses SK_P to sign $h_{HK}(m,s)$ along with HK to obtain signature $\langle S_{SK_P}(h_{HK}(m,s), HK)\rangle$. P presents this signature to O. O stores a copy of the signature and prepares a delegation warrant (W) containing $h_{HK}(m,s), HK$, other algorithm specific public parameters, and signature restrictions (expiration time, liability amount, etc.). That is, $W = \langle h_{HK}(m,s), HK$, parameters, restrictions$\rangle$. O then signs using SK_O to produce $\langle S_{SK_O}(W)\rangle$, and then passes $\langle S_{SK_O}(W)\rangle$ to P as a certificate of delegation warrant W.
- *Proxy-signing algorithm PRO–SIGN*: For a presented message m', P uses TK to compute s' such that $h_{HK}(m,s) = h_{HK}(m',s')$. P then presents $\langle S_{SK_O}(W), s', W\rangle$ to V as a proxy signature for message m'.
- *Verification algorithm PRO–VERF*: V first verifies $\langle S_{SK_O}(W)\rangle$ using VK_O and then extracts HK and $h_{HK}(m,s)$ from W. V then computes $h_{HK}(m',s')$. Signature $\langle S_{SK_O}(W), s', W\rangle$ is valid on message m' if and only if $h_{HK}(m,s) = h_{HK}(m',s')$.

4.2 Internal dispute resolution

We discuss two scenarios for internal dispute between signers O and P.

4.2.1 Dispute 1: In this dispute, P repudiates generation of valid signature $\langle S_{SK_O}(W), s'', W\rangle$ for message m''.

Case 1: P is the actual signer: From the presented signature, O extracts HK. O then presents to judge J, the stored signature $\langle S_{SK_P}(h_{HK}(m,s), HK)\rangle$ of P on HK, collected during the delegation phase. This signature is presented to J to prove that HK was selected by P as a hash key during the delegation phase for trapdoor hash value $h_{HK}(m,s)$. Therefore the dispute resolves in favour of O.

Case 2: O is the actual signer: In this case, O cannot present P's signature $\langle S_{SK_P}(h_{HK}(m,s), HK)\rangle$ mentioned in case 1 under cryptographic assumption of the signature algorithm in use. Therefore, the dispute resolves in favour of P.

4.2.2 Dispute 2: In this dispute, O claims that P violated the one-timeness of delegation right by signing multiple messages using the same warrant. Let m' and m'' be the presented messages claimed to be signed by P using warrant W. Let $\langle S_{SK_O}(W), s', W\rangle$ and $\langle S_{SK_O}(W), s'', W\rangle$ be the signatures on messages m' and m'', respectively.

Case 1: P has actually signed multiple messages: Since both m' and m'' are signed under warrant W, the hash value $h(\cdot)$ and hash key HK are the same for both the signatures. O presents pairs (m', s') and (m'', s'') along with HK to J. Since $h_{HK}(m', s') = h_{HK}(m'', s'')$, J can resolve the dispute in favour of O by deriving TK corresponding to HK in one of the following ways depending on the trapdoor hash function in use.

- *Case a: DL-based trapdoor hash function (Section 3.1.1)*

From $h_{HK}(m', s') = h_{HK}(m'', s'')$, we have, $g^{m'} \cdot y^{s'} = g^{m''} \cdot y^{s''} \bmod p$. This gives linear equation $m' + xs' = m'' + xs'' \bmod q$. Since the only unknown quantity is x, it can be derived by solving this linear equation.

- *Case b: Factoring-based trapdoor hash function (Section 3.1.2)*

From $h_{HK}(m', s') = h_{HK}(m'', s'')$, we have, $g^{m'\|s'} = g^{m''\|s''} \bmod n$. Let $x = (m' \parallel s') - (m'' \parallel s'') \bmod \lambda(n)$. Since $m' \neq m'', x \neq 0$. Therefore $\lambda(n)$ divides x. Thus $\phi(n) = (p-1)(q-1) = 4p'q' = 2\lambda(n)$ divides $2x$. In essence, there exists a probabilistic polynomial time algorithm such that given input HK, it outputs a multiple of $\phi(n)$ for the trapdoor function. In [27] Miller shows that the factorisation of n, the trapdoor key $TK(p,q)$, can be computed from any multiple of $\phi(n)$.

Case 2: P has not signed multiple messages: Due to the collision resistance property [6] of trapdoor hash functions in use, given a message m' and P's signature $\langle S_{SK_O}(W), s', W\rangle$ on it, it is computationally infeasible for O to compute pair (m'', s'') such that $h_{HK}(m', s') = h_{HK}(m'', s'')$. Therefore O cannot provide sufficient information to enable J to derive secret key TK corresponding to hash key HK used by P to generate proxy signature. Thus the dispute resolves in favour of P.

5 Security analysis

We prove that our proposed signature scheme satisfies all of the security requirements listed in Section 1.

- *Strong unforgeability:* Since the key pair (TK, HK) is generated by P, only P, not even O, knows trapdoor secret TK. In order to generate a proxy signature on message m', P uses TK to find collision such that $h_{HK}(m,s) = h_{HK}(m', s')$. Under collision resistance property of the trapdoor function in use, it is computationally infeasible for anybody to find collision without knowing trapdoor secret TK.

- *Verifiability:* From the verification algorithm *PROVERF*, V validates delegation by using O's public key, VK_O, to verify O's signature on warrant W. Since W contains other public parameters such as $h_{HK}(m,s)$, HK, etc. for proxy signature, V uses these parameters to compare the hash values and validate the proxy signature. Since the entire verification operation uses only public parameters, our scheme achieves this property.

- *Signature indistinguishability:* The warrant of delegation in our scheme does not include identity of the proxy signer. Instead, the warrant includes hash key HK and hash value $h_{HK}(m,s)$ which are independent of the actual identity of the proxy signer. Since the keys to generate and verify the signature (TK, HK) are independent of the identity of the signer, V can use the same verification algorithm regardless of the actual identity of the signer.

- *Signature unlinkability:* As mentioned, V can use the same verification algorithm regardless of the actual identity of the signer. Further, if a signer holds multiple warrants for multiple proxy signatures, the hash key HK and hash value $h_{HK}(m,s)$ for each warrant are independent of the actual identity of the signer. Therefore no third party, including V, can link two signatures from the same actual signer.

- *External undeniability:* For a given message, if the V can present a signature that passes the verification under regular signature assumption, O cannot falsely deny the generation of the signature either by himself or by P.

- *Internal undeniability:* As discussed under dispute 1 in Section 4.2.1, for a given valid signature, the actual identity of the signer can be determined.

- *One-timeness:* As discussed under dispute 2 in Section 4.2.2, for every delegation, if P generates more than one proxy signature, the trapdoor key TK can be derived. That is, for a delegation, P cannot securely generate more than one proxy signatures.

6 Conclusions

In practice the delegation of signing ability need not be a publicly known operation. Any party other than the signers need not be concerned about the actual identity of the signer as long as a valid signature is presented for verification. However, in case of repudiation, the internal disputes between the signers need to be resolved. Further, it may be very important to restrict the signing ability of the proxy signer. Furthermore, true privacy is provided to the signer if the actual identity of the signer cannot be determined by a third party via any kind of traffic analysis. Most importantly, all these features need to be provided in an efficient manner.

Our proposed model provides all the mentioned features efficiently in that, rather than adopting consecutive signature operations for delegation and proxy signature generation, the computational load is split between online and offline phases.

7 References

1 Lee, B., Kim, H., and Kim, K.: 'Strong proxy signature and its applications'. Proc. 2001 Symp. on Cryptography and Information Security (SCIS'01), January 2001, Vol. 2/2, pp. 603–608

2 Foster, I.T., Kesselman, C., Tsudik, G., and Tuecke, S.: 'A security architecture for computational grids'. Proc. ACM Conf. on Computer and Communications Security, 1998, pp. 83–92

3 Bakker, A., van Steen, M., and Tanenbaum, A.: 'A law-abiding peer-to-peer network for free-software distribution', 2002

4 Park, H.-U., and Lee, I.-Y.: 'A digital nominative proxy signature scheme for mobile communication'. Proc. 3rd Int. Conf. on Information and Communications Security, (Springer-Verlag, 2001), pp. 451–455
5 Mambo, M., Usuda, K., and Okamoto, E.: 'Proxy signatures for delegating signing operation'. Proc. 3rd ACM Conf. on Computer and Communications Security, (ACM Press, 1996), pp. 48–57
6 Shamir, A., and Tauman, A.: 'Improved online/offline signature schemes'. Proc. 21st Annual Int. Cryptology Conf. on Advances in Cryptology, (Springer-Verlag, 2001), pp. 355–367
7 Kim, S., Park, S., and Won, D.: 'Proxy signatures, revisited'. Proc. Conf. on Information and Communications Security (ICICS97), *Lect. Notes Comput. Sci.*, 1997, Vol. 1334, pp. 223–232
8 Shum, K., and Wei, V.-K.: 'A strong proxy signature scheme with proxy signer privacy protection'. Presented at 11th IEEE Int. Workshop on Enabling Technologies: Infrastucture for Collaborative Enterprises (WETICE'02)
9 Sun, H.-M., and Hsieh, B.T.: 'Cryptanalysis of a strong proxy signature scheme with proxy signer privacy protection'. Presented at IEEE Int. Carnahan Conf. on Security Technology, 2003
10 Lee, B., Kim, H., and Kim, K.: 'Secure mobile agent using strong non-designated proxy signature'. Proc. of ACISP2001, *Lect. Notes Comput. Sci.*, 2001, Vol. 2119, pp. 474–486
11 Kim, H., Baek, J., Lee, B., and Kim, K.: 'Secret computation with secrets for mobile agent using one-time proxy signature'. Presented at Symp. on Cryptography and Information Security (SCIS), January 2001
12 Wang, H., and Pieprzyk, J.: 'Efficient one-time proxy signatures'. Proc. ASIACRYPT 2003, *Lect. Notes Comput. Sci.*, 2003, Vol. 2894, pp. 507–522
13 Al-Ibrahim, M., and Cerny, A.: 'Proxy and threshold one-time signatures'. Proc. Conf. on Applied Cryptography and Network Security (ACNS'03), *Lect. Notes Comput. Sci.*, 2003, Vol. 2846, pp. 123–136
14 Hwang, J., Lee, D., and Lim, J.: 'Digital signature scheme with restriction on signing capability'. Proc. ACISP, *Lect. Notes Comput. Sci.*, 2003, Vol. 2727, pp. 324–335
15 Choi, C.-J., Kim, Z., and Kim, K.: 'Schnorr signature scheme with restricted signing capability'. Proc. Computer Security Symp. 2003, pp. 385–390
16 Zhang, K.: 'Threshold proxy signature schemes'. Proc. Information Security (ISW97), *Lect. Notes Comput. Sci.*, 1997, Vol. 1396, pp. 282–290
17 Zhang, K.: 'Nonrepudiable proxy signature schemes', 1997, [online], Available: citeseer.ist.psu.edu/zhang97nonrepudiable.html
18 Lee, N.-Y., Hwang, T., and Wang, C.-H.: 'On Zhang's nonrepudiable proxy signature schemes'. Proc. 3rd Australasian Conf. on Information Security and Privacy, (Springer-Verlag, 1998), pp. 415–422
19 Okamoto, T., Tada, M., and Okamoto, E.: 'Extended proxy signatures for smart cards'. Proc. 2nd Int. Workshop on Information Security, (Springer-Verlag, 1999), pp. 247–258
20 Ghodosi, H., and Pieprzyk, J.: 'Repudiation of cheating and non-repudiation of zhang's proxy signature schemes'. Proc. 4th Australasian Conf. on Information Security and Privacy, (Springer-Verlag, 1999), pp. 129–134
21 Shao, Z.: 'Proxy signature schemes based on factoring', *Inf. Process. Lett.*, 2003, **85**, (3), pp. 137–143
22 Lee, J., Cheon, J., and Kim, S.: 'An analysis of proxy signatures: Is a secure channel necessary?'. Proc. CT-RSA Conf. 2003. *Lect. Notes Comput. Sci.*, 2003, Vol. 2612, pp. 68–79
23 Boldyreva, A., Palacio, A., and Warinschi, B.: 'Secure proxy signature schemes for delegation of signing rights'. Cryptology ePrint Archive Report 2003/096, 2003, (http://eprint.iacr.org/)
24 Bamasak, O., and Zhang, N.: 'A secure method for signature delegation to mobile agents'. Proc. 2004 ACM Symp. on Applied Computing, (ACM Press, 2004), pp. 813–818
25 Krawczyk, H., and Rabin, T.: 'Chameleon signatures'. Proc. Symp. on Network and Distributed Systems Security (NDSS00), (Internet Society, February 2000), pp. 143–154
26 Fischlin, M.: 'Trapdoor commitment schemes and their applications'. PhD Thesis, Johann Wolfgang Goethe-Universität, Frankfurt am Main, 2001
27 Miller, G.L.: 'Riemann's hypothesis and tests for primality', *J. Comput. Syst. Sci.*, 1976, **13**, pp. 300–317