A Secure Method for Signature Delegation to Mobile Agents

Omaima Bamasak and Ning Zhang
Department of Computer Science,
University of Manchester,
Oxford Rd, Manchester M13 9PL, UK
(+44) (0) 161 275 6117
{obamasak, nzhang}@cs.man.ac.uk

ABSTRACT
This paper presents a novel method that allows the delegation of signature power to one or more entities that jointly play the role of a proxy signer. This work is different from other related proxy signature schemes in that in addition to providing confidentiality protection to the proxy key, the method provides non-repudiation services to all the parties involved. In particular, it protects against repudiation of signature delegation by the original signer, repudiation of proxy signature generation by the proxy signer, and repudiation of receipt of the proxy signature by the signature recipient. This feature is attractive for signature delegation in agent-based paradigm in which proxy signers are mobile agents that are executed in remote untrustworthy hosts.

Categories and Subject Descriptors
K.4.4 [**Computer and Society**]: Electronic Commerce – *distributed commercial transactions, security, mobile code security.*

General Terms
Security.

Keywords
mobile agent security; signature delegation; non-repudiation

1. INTRODUCTION
Mobile software agents have emerged as a promising paradigm for a number of applications ranging from ubiquitous computing to mobile/electronic commerce (m-/e-commerce). One of the tasks a mobile agent is expected to perform is to sign a digital signature on behalf of its owner. In other words, a mobile agent may perform the role of a proxy signer signing signatures on behalf of the original signer (i.e. the owner of the agent) autonomously on remote hosts. Such signatures are sometimes referred to as proxy signatures. As the remote hosts are not trustworthy, or may be malicious, the challenges faced by signature delegation in mobile agent systems are great. Among these challenges are, firstly, the signature key carried by an agent (also called proxy key) may be abused by the agent. For example, the agent may use the key to sign messages on which the original signer has not authorized the agent to sign. Alternatively, the proxy key may be spied on by other agents or remote hosts on which the agent is executed. Consequently, the agents or hosts may forge this agent's proxy signature. Furthermore, the agent or the owner of the agent may falsely deny having signed a specific signature, i.e. repudiation of signing or repudiation of signature origin. Finally, a signature recipient, e.g. a remote host to which the agent has made an e-payment, may later falsely deny having received the signature (the payment), i.e. repudiation of receipt of a specific signature. To securely delegate the signing power to mobile agents in such applications as agent-based e/m-commerce systems or agent-based financial/banking systems, the signature delegation method must meet the following security requirements:

1. **Verifiability**: Validity of a proxy signature as well as the original signer's delegation on signature signing on a particular message can be verified using public parameters.
2. **Unforgeability**: It is difficult for any other entities than the original signer and the designated proxy signer to generate a valid proxy signature on a specific message. This requirement implies that the proxy signature key should be kept confidential as indicated by the confidentiality requirement next.
3. **Confidentiality of proxy key**: Only the original signer should have complete knowledge of the proxy signature key.
4. **Non-repudiation of proxy signing**: It is difficult for a proxy signer to falsely deny having signed its proxy signatures.
5. **Non-repudiation of signature delegation**: It is difficult for an original signer to falsely deny that it has delegated the signing power to a proxy signer.
6. **Non-repudiation of signature receipt**: It is difficult for a signature recipient to falsely deny that it has received a specific proxy signature if this signature is taken as the proof of a deal struck by the proxy signer and the recipient.

There have been various ideas and schemes [5, 6, 7, 8, 9, 10, 14, 15, 17] proposed in relation to signature delegation since the idea was first discussed by Mambo, et al [9, 10] in 1996. However, based upon our best knowledge, only the work in [6] has achieved all of the six requirements specified above. Here in this paper we will propose a new method for signature delegation. The novelty of this method lies in, firstly, it satisfies all the six security requirements mentioned above, and secondly, it is more efficient than the scheme proposed in [6] and places less computational requirement on the side of the original signer than that on the

SAC' 04, March 14-17, 2004, Nicosia, Cyprus.

remote host. As the result, our method is more suited to agent-based mobile applications for which mobile devices are expected to have less computational and storage capabilities.

The remainder of the paper is organized as follows. Section 2 summarizes the related works in the field of signature delegation and comments on their suitability to agent-based signature delegation applications. Section 3 presents the principles and philosophy on which our solution is based. Section 4 gives detailed coverage of our novel solution. Section 5 provides security and performance analysis of our work. Finally, Section 6 outlines our conclusions.

2. RELATED WORK

Proxy signatures and undetachable signatures are the two most representative approaches to delegating signing power to mobile agents.

2.1 Proxy Signatures

The Concept of a proxy signature was first introduced by Mambo et al. [9, 10]. They classified proxy signatures based on delegation type as full delegation (giving the original signer's private key itself), partial delegation (issuing a new key pair), and delegation by warrant (issuing a certificate stating the delegation information). The work gave various methods of constructing proxy signatures and the security analyses of these methods. However, Lee et al. [7] identified several security weaknesses in these schemes. For example, as a proxy signature generated using any of these schemes does not contain any authentic information of the proxy signer, the proxy signer may later repudiate his signature by claiming that the signature was generated by the original signer. In addition, it is also possible for the proxy signer to abuse the proxy key, e.g. he may use it to sign messages, which are not authorized by the original signer but for which the original signer will be held responsible. Therefore, these schemes do not satisfy requirements 3, 4 and 5.

Lee et al. [7] also introduced the concept of a strong proxy signature, which provides non-repudiation for both the original signer and the proxy signer. This is achieved by computing a proxy key pair from both the original signer's and the proxy signer's private keys. However, this scheme gives the proxy signer a complete control over the use of the proxy key, as the proxy signer, rather than the original signer, has complete knowledge of the proxy key. Therefore, the proxy signer can generate more than one valid signature on more than one valid message using a proxy key. Hence the scheme does not address requirement 3. To solve the problem, Kim et al. [6] introduced a one-time signature concept into the above scheme, thus allowing one proxy key to generate only one valid signature. If a proxy signer generates more than one signature with the same proxy key, he will risk having his private key disclosed. Although this scheme provides good protection against proxy key abuse, it requires the original signer to generate four delegation (EL-Gamal type [3]) key pairs and the proxy signer to generate four proxy key pairs. This means four computational expensive exponentiation operations are needed on each side. This is considered to be inefficient and resource consuming specially if the original signer uses a resource-limited mobile device.

2.2 Undetachable Signatures

An undetachable signature scheme is a concept aimed at protecting an original signer's private key that is delegated to an agent, i.e. for the case of full-delegation. The first such scheme was proposed by Sander and Tschudin [15]. The scheme is based on the idea of Computing with an Encrypted Function (CEF). In this scheme, an original signer creates an encrypted signature function, $s \circ f$, where f is an encryption function and s is the delegated private signature function. This encrypted signature function is then delegated to an agent that is dispatched to a remote host. The host can execute this function to generate an original signer's signature without having access to the private signature function s. The paper proposed an implementation of the idea using birational functions as introduced by Shamir [Sham97]. However, the scheme constructed using these functions is insecure and subject to the Coppersmith, Stem and Vaudenay attack [2].

More recently, Kotzanikolaou et al. [8] implemented a CEF-based undetachable signature scheme using the RSA algorithm [13]. In the scheme, the original signer signs his requirement information using the RSA signature method and constructs an encrypted signature function, and then gives the function to a mobile agent. The remote host that receives the agent can compute the function to regenerate an original signer's signature on the document that meets the requirements. This scheme is provably secure since it uses an exponential function as the encrypting function instead of the birational function.

Although this scheme can protect the original signer's private key and addresses requirements 1-3 and 5 mentioned in Section 1, it does not satisfy requirements 4 and 6. In other words, the undetachable signature represents only the original signer's signature and it can be computed by any remote host or more than once by a single remote host, therefore the remote host can later repudiate the signature it has generated. Let us use an example to illustrate this security weakness. Imaging Alice delegates her agent a flight ticket purchase task. After the agent completes its booking and payment processes with a remote host - the travel agent's host. The host can repudiate his signature signed on the deal and refuse to deliver the flight ticket. A simple solution to this problem may be to code the agent such that the agent asks for the host's signature on the deal before taking the signed deal back to its owner. However, as the mobile agent exposes its code and data to its execution environment, the remote host may maliciously alter the agent's code so that its execution skips the step where the agent asks for the host's signature on the deal.

3. DESIGN PRINCIPLES

The following measures have been taken in the design of our solution in order to achieve a better level of security in signature delegation and to satisfy all the requirements specified in Section1.

- **Measure 1**: The idea of partial delegation [9] is used. That is, a proxy key, rather than an original signer's private key, is used to generate a proxy signature, and the proxy key is generated from the original signer's private key. In this way, the proxy key can be made context-dependent for a specific signing mission. For example, the validity of a proxy signature can be restricted to certain context, price, agent/host identities, etc. Thus, any damage

caused by compromising the proxy key will be less than that caused by the original private key.

- **Measure 2**: The secret sharing scheme [16] is used to facilitate the idea of distributed signature generation to reduce the likelihood of proxy key being abused either by the key- carrying agent (i.e. the proxy agent) or any of the remote hosts on which the agent is executed. In detail, the scheme is used by the original signer (i.e. the owner of the proxy agent) to divide a proxy key into two shares: one is delegated to the agent that is dispatched into the network, and the other is sent to a trusted remote host, that is called trusted third party (*TTP*) hereafter. By dividing a proxy key into two shares and distributing them to two separate entities, the measure can prevent any single entity, i.e. the original signer, the proxy agent, the remote hosts (on which the agent will be executed), or the *TTP,* from abusing the proxy key without any collusion. In other words, this measure can give the proxy key a better level of protection in an agent-based environment.

- **Measure 3**: The digital signcryption scheme by Gamage, et al. [4] is used to protect the confidentiality of a proxy key share and to generate a partial signature using the proxy key share. A signcryption scheme is a cryptographic method that fulfills both the functions of secure encryption and digital signature in a logically single operation, but cost less than that by signature-then-encryption approach. It has been shown in [18] that for security parameters recommended for medium to long term security (i.e. size of public modulo = 1536 bits), the signcryption scheme costs on average 50% less in computation time and 91% less in message expansion than that by signature-then-encryption method using the RSA cryptosystem [13]. The notable works on the signcryption scheme are those by Bao and Deng [1] and by Gamage, et al. [4]. The scheme by [4] has one feature that can be exploited for us to design method to satisfy the security requirement of non-repudiation of signature receipt by remote hosts. In the Gamage et al. scheme, the verification of the proxy signature can be performed without using any secrets owned by remote hosts. Thus we can introduce measures to deprive the ability of signature verification by remote hosts so as to force these hosts to resort to a *TTP* for signature verification. In this way, the hosts will be forced to leave evidence of signature receipt thus protecting the original and proxy signers against repudiation of signature receipt by the remote hosts.

- **Measure 4**: A *TTP* has been introduced into our solution to facilitate an idea of distributed signature generation and to assist the provision of non-repudiation of signature delegation and non-repudiation of proxy signature generation and receipt. In detail, The *TTP* performs the following tasks. Firstly, it receives a proxy key share from an original signer and generates a partial proxy signature on a specified document using the Threshold Proxy Signcryption Scheme (TPSS) to be presented in Section 4 next. Secondly, it verifies the validity of a complete proxy signature once the signature is reconstructed and sent by a remote host. Finally, if the signature verification is positive, it generates signature verification token and returns the token back to the remote host as an evidence of signature receipt by the remote host. This can protect the host against repudiation of signature delegation by the original signer, repudiation of proxy signature generation by the mobile agent. In addition, if the host later denies the receipt of the proxy signature, the *TTP* will use the joint evidence - the signature verification request sent by the host plus the signature receipt token generated in response to the request - to thwart the false claim.

4. OUR SOLUTION IN DETAIL

4.1 Notations and Prerequisites

The following notation has been used in the remaining part of this paper to describe our solution.

A:	An agent owner.
B:	A remote host.
MA:	A mobile agent that is created by A and it performs specified tasks on behalf of A.
TTP:	An on-line Trusted Third Party. It is assumed that the TTP always executes the scheme in the solution correctly and does not conspire with either MA or B.
y_i/x_i:	A public/private ElGamal-type key pair of entity i, where $i \in \{A, B, TTP\}$.
PK_A:	A proxy key generated from the private key x_A of party A.
SH_i:	A proxy key share generated from the proxy key PK_A by A and given to party $i \in \{B, TTP\}$.
$H(m)$:	A one-way hash function applied on m, which produces a fixed-size output (digest) for a variable size input (m). Given $H(m)$, it is computationally infeasible to determine m. It is also collision free, i.e. it is computationally infeasible to find distinct m and n such that $H(m) = H(n)$. An example of such a hash function is SHA-1 [11].
(E_k, D_k):	Encryption and decryption operations using of a symmetric cipher such as Data Encryption Standard (DES) [12] and key k.
ID_i:	A unique identifier for entity i.
VT:	A proxy signature verification token generated and signed by TTP.
M:	A document to be signed by MA on behalf of its owner A.

SDSS1 [18] signature *(r, s)* on message m signed using *A*'s private key x_A:

$r = H(g^x \bmod p, m)$, where x is a random number $\in Z_q$.
$s = x / (r + x_A) \bmod q$, the signature is (s, r).

SDSS1 signature verification using *A*'s public key y_A:

$j = (y_A g^r)^s \bmod p$.
Check whether $H(j, m) = r$.

4.2 Our Solution Overview

The overview of our solution for proxy signature generation and verification is depicted in Figure 1. As can be seen, the solution is featured by three novel contributions.

- **The Proxy Key Generation Method**: Agent owner A generates a proxy key from his private key using this method to be detailed in Section 4.3.

- **The Threshold Proxy Signcryption Scheme (TPSS)**: Using the TPSS scheme, (1) the proxy key is divided into two shares: SH_{MA} carried by the MA and SH_{TTP}, securely sent to the TTP; (2) The proxy key shares are used by the

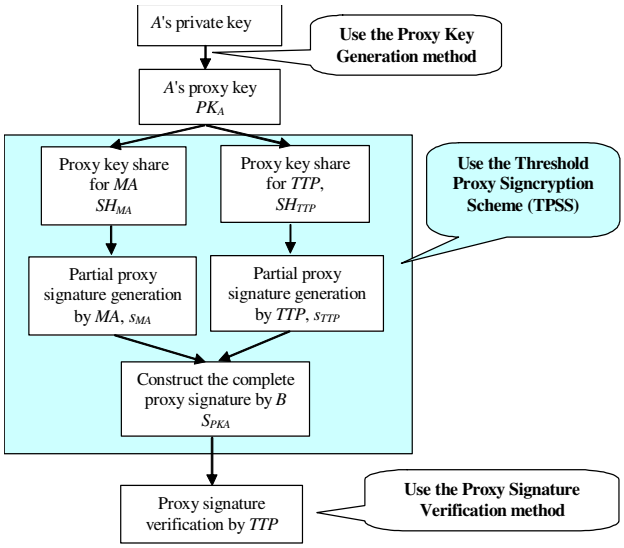

Figure 1. Our Solution Overview

MA and the *TTP* to generate two partial proxy signatures, s_{MA} and s_{TTP}, using the signcryption method, respectively; (3) the complete proxy signature is constructed using the two partial proxy signatures by host *B*. The scheme is detailed in Section 4.4 next.

- **The Proxy Signature Verification Method:** Once the proxy signature is constructed, (only) the *TTP* (can) verifies the correctness of the proxy signature by using this method to be detailed in Section 4.5 next.

In the following subsections, we give detailed coverage on the proxy key generation and proxy signature verification methods and the TPSS scheme.

4.3 Proxy Signature Key Generation Method

To generate a proxy key PK_A from his private key x_A, the owner *A* generates two random numbers $a, \beta \in Z_p$ and computes:

$$w = g^{\beta} \bmod p. \quad (1)$$

$$PK_A = x_A \alpha + \beta w \bmod p. \quad (2)$$

A then keeps β as a secret.

4.4 Threshold Proxy Signcryption Scheme (TPSS)

Once the proxy key PK_A is generated, it is divided into two shares, each of which is used to generate a partial proxy signature by two separate entities, i.e. the agent *MA* and the *TTP*. Upon the completion of a deal, a complete proxy signature is constructed using the two partial proxy signatures (by host *B*). All these tasks are fulfilled by the TPSS scheme. The scheme is designed by integrating a modified proxy signature generation method based upon the work by [9] with the Shamir's secret sharing scheme [16] and Gamage et al. signcryption scheme [4].

The TPSS uses a special case of Shamir-threshold scheme (t=n=2). To divide a proxy key into n=2 proxy key shares. *A* performs the following tasks.

- **Proxy key shares generation:** *A* generates a (n-1) degree polynomial:

$$f(x) = \sum_{i=1}^{n-1} a_i x^i + PK_A \bmod q, \quad (3)$$

where the coefficients a_i are chosen randomly from Z_q and n=2.*A* then computes two shares, SH_{MA} and $SH_{TTP.}$ For doing so, *A* needs to evaluate the polynomial $f(x)$ at $n(=2)$ different random points (x_i, $i \in \{MA, TTP\}$). These points will be made public after share generation process. Therefore, instead of generating additional public values, *A* can use values that are already public and distinct, e.g. shareholder's identities *ID*s, for this purpose. Thus, *A* computes shares $SH_i = f(ID_i)$, $i \in \{MA, TTP\}$ and securely transmits each share (SH_{MA}, ID_{MA}) and (SH_{TTP}, ID_{TTP}) to the corresponding shareholder (*MA* and *TTP*) together with a random number $R \in Z_q$. The random number *R* is kept the same for all shareholders for the signature generation purpose.

- **Partial proxy signature generation:** Each shareholder generates a partial proxy signature using signcryption on message *M* as follows:

$$k = H(y_B^R \bmod p) \quad (4)$$

$$y = g^R \bmod p \quad (5)$$

$$c = E_k(M) \quad (6)$$

$$r = H(y, c) \quad (7)$$

$$L_i = SH_i \prod_{j=1, j \neq i}^{n} \frac{-ID_j}{ID_i - ID_j} \bmod q, \quad (8)$$

$$s_i = R / (r + nL_i) \bmod q \quad (9)$$

where i $\in \{MA, TTP\}$.

Therefore, the partial proxy signature on *M* is (c, r, s_i), which is sent to *B*.

- **Proxy signature construction:** Upon receiving n individual partial signcryptions (c, r, s_i), *B* constructs the complete proxy signature using the following method.

$$S_{PK_A} = (n^{-1} \sum_{i=1}^{n} s_i^{-1})^{-1} \quad (10)$$

As *B* cannot verify the signature S_{PK_A}, it forwards it to *TTP*, where *Verification TTP* takes place, as explained in section 4.2.

Theorem: S_{PK_A} represents *A*'s valid proxy signature on *M*.

Proof: with the knowledge of n partial signcryptions (c, r, s_i), $1 \le i \le n$, *B* can generate a complete signature S_{PK_A} on *M*:

1. Computes the summation of the multiplicative inverses of s_i, $1 \le i \le n$

$$\sum_{i=1}^{n} s_i^{-1} = \sum_{i=1}^{n} \frac{r + nL_i}{R} \bmod q = \frac{nr + n\sum_{i=1}^{n} L_i}{R} \bmod q$$

$$= \frac{n(r + \sum_{i=1}^{n} SH_i \prod_{j=1, j \neq i}^{n} \frac{-ID_j}{ID_i - ID_j} \bmod q)}{R} \bmod q$$

$$= \frac{n(r + PK_A)}{R} \bmod q \quad (11)$$

2. Multiplies equation (11) by n^{-1}

$$(n^{-1}\sum_{i=1}^{n} s_i^{-1}) = n^{-1}\frac{n(r+PK_A)}{R} \bmod q$$

$$= \frac{r+PK_A}{R} \bmod q \qquad (12)$$

3. Computes the multiplicative inverse of equation (12)

$$(n^{-1}\sum_{i=1}^{n} s_i^{-1})^{-1} = \frac{R}{r+PK_A} \bmod q \qquad (13)$$

The result from equation (13) represents *A*'s proxy key signature on *M*.

4.5 Proxy Signature Verification Method

Once a complete proxy signature, S_{PK_A}, is constructed, it is verified by the *TTP* by using the proxy signature verification method. The procedure and the method are described below.

- **Signature verification request:** *B* sends the *TTP* a Verification Request that includes the items (M, S_{PK_A}, r).
- **Signature verification token (*VT*) generation:** Upon the receipt of the request, *TTP* carries out the following verifications:

Verification TTP: Checks the correctness of the signature S_{PK_A} *by the same checking operation as in the original SDSS1 signature scheme except for replacing* y_A *with* $y'_A = y_A^{\alpha} \times w^{w} \bmod p$ *, as shown in Table1.*

Table 1. Proxy signature S_{PK_A} verification method

The original SDSS1 verification method	Our proxy signature verification method
The signature is (s, r)	The signature is (S_{PK_A} , r`)
1. $u = (y_A \times g^{r})^{s} \bmod p$	1. $u' = (y'_A \times g^{r})^{S_{PK_A}} \bmod p$ (14)
2. Check whether $H(u, M) = r$	2. Checks whether $H(u', M) = r$

If the outcome of *Verification TTP* is positive, *TTP* generates a proxy signature verification token *VT* (*VT* is a signed token by the *TTP*, which typically includes a timestamp, the identities of all the entities involved, a hash value of the signed message). The token is then returned to *B*, which can protect *B* against false denial of signature delegation by *A* and protect both *A* and *MA* against repudiation of the recipient of *A*'s signature by *B*.

- ***VT* recipient response:** Upon the receipt of the *VT*, *B* verifies *TTP*'s signature on *VT*, and if it is valid, *B* signs *VT* using his private key x_B and submits the signed *VT* to *MA*. *MA* then returns back to its owner *A* carrying the signed *VT*, which represents the signature of both *A* and *B* on document *M*. Once the signed *VT* arrives, *A* confirms its validity by verify the signatures of both *TTP* and *B* on the *VT*.

4.6 Security Analysis

In this subsection, we demonstrate that the solution presented above satisfies all the security requirements stated in Section 1.

- When *TTP* performs the verification process *Verification TTP* by using *A*'s public key y_A, *TTP* is able to verify the proxy signature S_{PK_A} and will be convinced of *A*'s agreement on the signed message. Therefore, the Verifiability requirement is fulfilled.
- Since *A*'s private key x_A is used to generate the proxy key PK_A, no other party than *A* could have generated the proxy key PK_A. In addition, no one of the shareholders can generate a valid complete proxy signature on *M* by itself. This is because each shareholder has only a share of the proxy key and this share can only generate partial signatures. Furthermore, as *B* uses his private key x_B to sign the verification token *VT*, no party other than *B* could have generated this signature. Therefore, the Unforegability requirement is satisfied.
- It is difficult for one to compromise the proxy key PK_A due to the following reasons: (1) the proxy key is distributed in two shares, and (2) each share is protected by using the signcryption method. In other words, in order to compromise the proxy key, one has to intercept and hack the two signcrypted signature shares, unless any of the entities, *MA* or *B* colludes with the *TTP*, which contradicts our assumption made about the behavior of *TTP*. Hence, the Confidentiality requirement is met.
- The verification process *Verification TTP* performed by *TTP* ensures that the proxy signature S_{PK_A} on *M* is generated by using a proxy key that is generated from *A*'s private key. This is achieved by the inclusion of *A*'s public key in the verification process. Therefore, *A* cannot deny the fact the he has generated the proxy key, which satisfies the Non-repudiation of Signature Delegation requirement.
- The Non-repudiation of Signature Receipt is achieved by the *TTP* saves *B*'s request to verify the signature S_{PK_A} and the verification token *VT* signed by *TTP*, which confirms that *B* has actually received *A*'s signature on *M*. Therefore, *B* cannot deny later that has received *A*'s signature.
- To increase the level of security of the solution, the proxy key PK_A can be bound to some restrictions specified by the owner to the use of PK_A. In detail, PK_A can have a limited lifetime in which any signature generated after a specific time period will be considered as invalid. Moreover, the key PK_A can be made one-time only or transaction dependent.

5. COMPARISION WITH RELATED WORK

The designs of the methods and scheme presented in Section 4 have been geared towards mobile computing (M-computing) or ubiquitous computing (U-computing) environments. The devices in these environments are characterized by low bandwidth, expensive connectivity and limited computational and storage capabilities. Therefore, efficiency has been considered as one of the design focus in our solution. To demonstrate the efficiency of our solution in comparison with related work, we have chosen Kim's signature delegation scheme [6] as it is the most secure scheme seen in the literature. The efficiency is measured in terms of number of exponentiation operations since they are the most resource consuming cryptographic operations. Table 2 shows the number of exponentiation operations for each cryptographic operation used in our solution.

Table 2. Exponentiation operations in each cryptographic operation.

Cryptographic operation	Number of exp. Operation
Public key generation	1
ElGamal encryption	2
ElGamal decryption	1
SDSS1 signature generation	1
SDSS1 signature verification	2
Signcryption of message *m*	2
Unsigncryption of message *m*	2 for signature verification only 3 for signature verification and decryption of *m*

A comparison between our solution and Kim's scheme measured by the number of exponentiation operations performed by each entity in the solution is shown in table 3.

Table 3. Our solution vs. Kim's scheme

	Agent owner *A*	Remote host *B*	*TTP*	Total
Our solution	6	6	8	20
Kim's scheme	13	17	1	31

From the above table, it can be seen that our solution requires 35% less exponentiation operations in total than the Kim's scheme. If we look into the number of exponentiation operations performed in the agent's owner side (mobile device), it is clear that they are about 54% less in our solution than that in Kim's scheme. Therefore, we can state that our solution is far more efficient than Kim's scheme in M-/U-computing application arena.

6. CONCLUSION

In this paper, we have presented a novel solution for signature delegation to mobile agents. The novelty of our work is reflected by the two methods and one scheme: the proxy key generation method, the proxy signature verification method, and the TPSS scheme. These methods/scheme enable the integration of proxy signature concept, secret sharing scheme, and signcryption scheme to achieve a distributed signature delegation solution. This novel solution satisfies the six security requirements specified in Section 1, and makes it difficult for a single entity alone, be it the agent, the remote host or the *TTP*, to forge or abuse the proxy key or the proxy signature. Most importantly it can provide non-repudiation service to all of the participating entities, i.e. non-repudiation of signature delegation, non-repudiation of proxy signature generation and non-repudiation of receipt of proxy signatures. In comparison with the only other work [6] that can achieve all these security requirements, our method is more efficient.

For future work, we intend to apply our novel method to solving the problem of m-commerce, and implement the method in the context. We also plan to extend the protocol to provide agent's owner anonymity because it is a desirable feature for customers in most commercial transactions.

REFERENCES

[1] Bao, F., and Deng, R. A signcryption scheme with signature directly verifiable by public key. In *PKC' 98*, *LNCS 1431*, pages 55-59. Springer, 1998.

[2] Coppersmith, D., Stern, J., and Vaudenay, S. Attacks on the Birational Permutation Signature Schemes. In *Crypto'93*, *LNCS 773*, pages 435-443. Springer, 1993.

[3] ElGamal, T.A. Public Key Cryptosystem and Signature Scheme Based on Discrete Logarithms. *IEEE Transaction on Information Theory,* 31(4): 469-472, 1985.

[4] Gamage, C., Leiwo, J., and Zheng, Y. Encrypted message authentication by firewalls. In *PKC' 99, LNCS 1560*, pages 69-81. Springer, 1999.

[5] Kim, P., Park, S., and Won, D. Proxy Signature, Revisited. In *ICICS97*, *LNCS 1334*, pages 223-232. Springer, 1997.

[6] Kim, H., Baek, J., Lee, B., and Kim, K. Secret Computation with Secrets for Mobile Agent using One-Time Proxy Signature. In *SCIS2001*, pages 845-850. **publisher

[7] Lee, B., Kim, H., Baek, J., and Kim, K. Secure Mobile Agent Using Strong Non-designated Proxy Signature. In *ACISP 2001*, *LNCS 2119*, pages 474-486. Springer, 2001.

[8] Kotzanikolaou, P., Burmester, M., and Chrissikopoulos, V. Secure Transactions with Mobile Agents in Hostile Environments. In *ACISP'2000*, *LNCS 1841*, pages 289-297. Springer, 2000.

[9] Mambo, M., Usuda, K., and Okamoto, E. Proxy Signatures for Delegating Signing Operation. In *Proceedings of 3rd ACM Conference on Computer and Communication Security*, pages 48-57. ACM Press, 1996.

[10] Mambo, M., Usuda, K., and Okamoto, E. Proxy Signature: Delegation of the Power to Sign Messages. *IEICE Transactions Fundamentals*, E79-A(9):1338-1353, 1996.

[11] National Institute of Standard and Technology. Secure Hash Standard. *Federal Information Processing Standards Publication 180-1*, 1995.

[12] National Institute of Standard and Technology. Data Encryption Standard. *Federal Information Processing Standards Publication 46-3*, 1999.

[13] Rivest, R., Shamir, A., and Adleman, L. A Method for Obtaining Digital Signatures and Public Key Cryptosystems. *Communications of the ACM*, 21(2):120-126, 1978.

[14] Romao, A. and Silva, M. Secure mobile agent digital signatures with proxy certificates. *E-commerce Agents*, *LNCS 2033*, pages 206-220. Springer, 2001.

[15] Sander, T. and Tschudin, C. Protecting Mobile Agents against Malicious Hosts. *Mobile Agents and Security*, *LNCS 1419*, pages 44-60. Springer, 1998.

[16] Shamir, A. How to Share a Secret. *Communications of the ACM*, 22(11):612-613, 1979.

[17] Yi, L., Bai, G., and Xiao, G. Proxy Multi-signature Scheme: A New Type of Proxy Signature Scheme. *Electronic Letters*, 36(6):527-528, 2000.

[18] Zheng, Y. Digital signcryption or how to achieve cost (signature & encryption) << cost (signature + cost (encyption). In *CRYPTO'97*, *LNCS 1294*, pages 165-179. Springer, 1997.