Comments on a Threshold Proxy Signature Scheme Based on the RSA Cryptosystem

Guilin Wang, Feng Bao, Jianying Zhou, and Robert H. Deng

Infocomm Security Department (ICSD)
Institute for Infocomm Research (I^{2}R)
21 Heng Mui Keng Terrace, Singapore 119613
http://www.i2r.a-star.edu.sg/icsd/
{glwang,baofeng,jyzhou,deng}@i2r.a-star.edu.sg

Abstract. In a (t, n) proxy signature scheme, the original signer can delegate his/her signing capability to n proxy signers such that any t or more proxy singers can sign messages on behalf of the former, but $t-1$ or less of them cannot do the same thing. Such schemes have been suggested for use in a number of applications, particularly in distributed computing where delegation of rights is quite common. Based on the RSA cryptosystem, Hwang et al. recently proposed an efficient (t, n) threshold proxy signature scheme. In this paper we identify several security weaknesses in their scheme and show that their scheme is insecure.

Keywords: proxy signature, digital signature, public key cryptosystem, data security.

1. Introduction

Proxy signatures are first introduced by Mambo, Usuda, and Okamoto in [12, 13]. Such a scheme allows one user, called *original signer*, to delegate his/her signing capability to another user, called *proxy signer*. After that, the proxy signer can sign messages on behalf of the original signer. Upon receiving a proxy signature on some message, a verifier can validate its correctness by following a given verification procedure, and then is convinced of the original signer's agreement on the signed message if the validation is positive. Proxy signature schemes have been suggested for use in a number of applications, including e-cash systems, mobile agents, mobile communications, grid computing, global distribution networks, and distributed shared object systems etc [1, 24].

Based on the ideas of secret sharing [21, 15, 16] and threshold cryptosystems [2], Zhang and Kim et al. independently constructed the first threshold proxy signatures in [25] and [8], respectively. In a (t, n) threshold proxy signature scheme, the original signer's signing power is delegated to a group of n proxy singers such that t or more of them can generate proxy signatures cooperatively, but $t-1$ or less of them cannot do the same thing. This technology not only allows the original signer to delegate the proxy signing power to a group of proxy signers instead of one single proxy signer, but also lets the original signer to set the threshold value t freely ($1 \leq t \leq n$). Therefore, the threshold proxy signature approach is more practical, flexible and secure than standard proxy signature schemes.

A practical and secure (t, n) threshold proxy signature scheme should satisfy the following six requirements [7]:

- *Secrecy.* The original singer's private key cannot be derived from any information, such as the shares of the proxy signing key, proxy signatures etc. Particularly, even all proxy signers collude together, they cannot derive the original signer's private key.
- *Proxy Protection.* Only the delegated proxy signer can generate valid partial proxy signatures. Even the original signer cannot masquerade as a proxy signer to create partial signatures.
- *Unforgeability.* A valid proxy signature can only be cooperatively generated by t or more proxy signers. This means that valid proxy signatures cannot be created by $(t-1)$ or less proxy signers, or any third parties who are not designated as proxy signers.
- *Nonrepudiation.* Any valid proxy signature must be generated by t or more proxy signers. Therefore, proxy signers cannot deny that they have signed the message. In addition, the original signer cannot deny having delegated the power of signing messages to the proxy signers.
- *Time Constraint.* The proxy signing keys can be used only during the delegated period. Once they expire, the proxy signatures generated by using those keys become invalid.
- *Known signers.* For internal auditing purposes, the system is able to identify the actual signers of a given threshold proxy signature.

Following the first threshold proxy signatures in [8, 25], a number of improvements and new schemes have been proposed [18, 19, 6, 5, 23]. However, most of them do not meet all the above security requirements. At the same time, all these schemes are based on the discrete logarithm crytosystems [4, 20]. The main reason is that it is difficult to share the private key of the RSA cryptosystem [17] among multiple parties. The RSA cryptosystem is now the de facto industrial standard and is widely used in many applications; therefore, it is highly desirable to construct threshold proxy signature schemes based on the RSA cryptosystem. Hwang et al. recently proposed an RSA based (t, n) threshold proxy signature scheme in [7]. This scheme is dramatically more efficient than previous schemes in both computation and communication by not using the distributed random number generation protocol in [15, 16]. For example, after a detailed performance analysis, Hwang et al. concluded that their scheme only requires 5 percent computation overheads and 8 percent communication overheads of Kim et al.'s scheme [8]. Unfortunately, as we will show shortly, the scheme is not secure. Though Hwang et al. claimed that their scheme satisfies all the security requirements listed above, our analysis indicates that the scheme fails to satisfy all the requirements except the one on time constraint. For simplicity, we call their scheme the HLL scheme hereafter. The rest of this paper is organized as follows. We review the HLL scheme in Section 2 and present our analysis in Section 3. We conclude the paper in Section 4.

2. Review of the HLL Scheme

In the HLL scheme [7], the following notations are used. The original signer is denoted by P_0 and the n proxy signers are denoted by P_i $(i = 1, 2, \cdots, n)$. For $i = 0, 1, \cdots, n$,

N_i denotes P_i's RSA modulus, and (e_i, d_i) represents P_i's public/private key pair, where N_i is the product of two large primes p_i and q_i, $d_i e_i = 1 \bmod \phi(N_i)$, and $\phi(N_i) = (p_i - 1)(q_i - 1)$. In addition, the message m_w stands for a *warrant*, which specifies the validity period of the proxy key, the identities of the original signer and proxy signers, and the kind of messages being delegated etc.

2.1 The Proxy Sharing Phase

(1) *Proxy Generation.* The original signer P_0 generates the group proxy signing key D and the corresponding proxy verification key E as follows:

$$D = d_0^{m_w} \bmod \phi(N_0), \tag{1}$$

$$E = e_0^{m_w} \bmod \phi(N_0). \tag{2}$$

Then, P_0 publishes $\{m_w, E, [m_w||E]^{d_0} \bmod N_0\}$.

(2) *Proxy Sharing.* P_0 selects a random secret polynomial $f(X)$ of degree $(t-1)$ in the form

$$f(X) = D + a_1 X + + \cdots + a_{t-1} X^{t-1} \bmod \phi(N_0), \tag{3}$$

where $a_1, a_2, \cdots, a_{t-1}$ are random numbers. Then, P_0 computes P_i's partial proxy signing key $k_i = f(i)$, and sends $[[k_i]^{d_0} \bmod N_0||k_i]^{e_i} \bmod N_i$ to the proxy signer P_i, where i represents P_i's identity.

(3) *Proxy Share Generation.* When each proxy signer P_i receives $[[k_i]^{d_0} \bmod N_0||k_i]^{e_i} \bmod N_i$, he decrypts the ciphertext to obtain $\{[k_i]^{d_0} \bmod N_0, k_i\}$. Then, the proxy signer P_i confirms the validity of k_i and keeps it secret.

2.2 The Proxy Signature Issuing Phase

Let T denote a subset of t or more proxy signers who want to cooperatively generate a proxy signature on a message M on behalf of P_0. For this sake, they perform the following operations.

(1) With the partial proxy signing key k_i, each P_i, $i \in T$, generates his partial proxy signature s_i on message M as

$$s_i = M^{L_i \cdot k_i} \bmod N_0, \tag{4}$$

where the Lagrange interpolating coefficient L_i is given by

$$L_i = \prod_{i,j \in T, j \neq i} \frac{-j}{i-j}. \tag{5}$$

Then, each P_i $(i \in T)$ sends $\{[s_i]^{d_i} \bmod N_i, s_i\}$ to the combiner.

(2) The combiner verifies s_i using the public key of P_i and stores $[s_i]^{d_i} \bmod N_i$. If all the partial proxy signatures are valid, the combiner generates the proxy signature S on message M using the following equation:

$$\begin{aligned} S &= \textstyle\prod_{i \in T} s_i \bmod N_0 \\ &= M^{\sum_{i \in T} L_i \cdot f(i)} \bmod N_0 \\ &= M^{f(0)} \bmod N_0 \\ &= M^D \bmod N_0. \end{aligned} \tag{6}$$

2.3 The Proxy Signature Verification Phase

Using the publicly known parameters N_i, e_i, m_w and E, any receiver can validate a proxy signature S by the following the verification procedure below.

(1) The verifier first computes m_w and E with the original signer's public key. Then, the stipulated delegation period is checked. If the period has expired, the proxy verification key is invalid.
(2) The verifier then checks the validity of proxy signature S by the following equation:

$$\begin{aligned} S^E \bmod N_0 &= (M^D)^E \bmod N_0 \\ &= M^{d_0^{m_w} \cdot e_0^{m_w}} \bmod N_0 \\ &= M \bmod N_0. \end{aligned} \tag{7}$$

(3) For internal auditing purpose, the original signer can differentiate the actual signers from the signatures $[s_i]^{d_i} \bmod N_i$ on message s_i, where $i \in T$.

3. Our Comments on the HLL Scheme

In [7], Hwang et al. provided a detailed security discussion on the scheme and claimed that their scheme meet all the security requirements listed in Section 1. However, we discover that the claim is not true. In this section, we present some security weaknesses and flaws in the HLL scheme.

3.1 Security

(1) **Secrecy**. Hwang et al. claimed that their scheme meet secrecy, i.e., even t or all proxy signers collude together, they cannot recover the original signer's private key. They argued that though t or more proxy signers can get the proxy signing key D, they cannot derived the original signer's private key d_0 from the equation $D = d_0^{m_w} \bmod N_0$, since the values of d_0 and $\phi(N_0)$ are not known to them. However, this is not the fact. We explain the details as follows. Firstly, for any subset $T \subset \{1, 2, \cdots, n\}$ satisfying $|T| \geq t$, Eq. (6) implies that the following equation holds:

$$D = \sum_{i \in T} L_i \times k_i. \tag{8}$$

In other words, any t or more colluding proxy signers can get the proxy signing key D by Eq. (8). Secondly, since $E = e_0^{m_w} \bmod \phi(N_0)$ and $e_0 d_0 = 1 \bmod \phi(N_0)$, we have $ED = 1 \bmod \phi(N_0)$. This means that $ED-1$ is a non-zero multiple of $\phi(N_0)$. However, it is well known that knowing such multiple of $\phi(N_0)$ is equivalent to factoring N_0 (e.g., page 91 of [9]). Finally, with the factors of N_0, the proxy singers can compute the value of $\phi(N_0)$, and then get the value of d_0 easily from the equation $d_0 e_0 = 1 \bmod \phi(N_0)$ by using the extended Euclidean algorithm. Therefore, the result is that the original signer's private key is revealed to proxy signers[1]. Note that the well known fact that an RSA modulus cannot be shared among different users is also due to this algorithm.

[1] Note that in [3], a similar attack is mounted by Dodis et al. to break an RSA-based optimistic fair exchange protocol proposed in [14].

(2) **Proxy Protection**. Since the original signer P_0 knows all the partial proxy signing keys, i.e, k_i's, P_0 can generate partial proxy signatures on any message M by computing $s_i = M^{L_i \times k_i} \bmod N_0$. However, P_0 cannot generate the corresponding signature for s_i on behalf of P_i, since P_0 does not know P_i's private key d_i. Therefore, the authors of [7] concluded that in the HLL scheme even the original signer cannot generate valid proxy signatures to frame the proxy signers. We note that the combiner in the system is just a role for enhancing efficiency and internal audit. Furthermore, each proxy signer's signature $s_i^{d_i} \bmod N_i$ is not included in the proxy signature. Hence a verifier accepts a threshold proxy signature S just by checking whether it satisfies Eq. (7). This means that with the knowledge of the group proxy private key D, the original signer P_0 can generate valid threshold proxy signature on any message M of its choice by directly computing $S = M^D \bmod N_0$.

Moreover, we remark that the protocol of proxy sharing in the HLL scheme is different from the standard ones [12, 8, 10, 11]. In general, the following two properties should be satisfied by the proxy signing key generation protocol of (threshold) proxy signatures: (a) A valid proxy signing key can ONLY be generated JOINTLY by the proxy signer and proxy signers; and (b) Each proxy signer knows the (partial) signing key but the original singer does not know it. Based on these two properties, a proxy signature scheme can be guaranteed to meet the security requirement of proxy protection, unforgeability, and non-repudiation. In the HLL scheme, however, the partial proxy signing key k_i is just a share of D created by the original signer alone and not bound to the private key d_i of the proxy signer P_i . Therefore, the original signer can forge a new warrant m_w without the agreements of all the proxy signers, and then calculate the corresponding group proxy signing key D and verification key E. Then, as we mentioned above, using the value of D the original proxy signer can generate valid threshold proxy signatures in the names of proxy signers. Upon receiving such a proxy signature, a verifier will accept it and be convinced that it is signed by some proxy signers.

(3) **Non-Repudiation**. In fact, to generate a proxy signature any proxy signer can play the role of the combiner. This implies that if t proxy singers collude, they can generate a proxy signature in the same way but without the help of the designated combiner. At the same time, as we mentioned above, the original signer can also generate valid threshold proxy signatures directly by using the group proxy signing key D. Therefore, the combiner can be surpassed by the original signer and the proxy signers. The resulting problem is that when such valid threshold proxy signatures are presented in future disputes, who should take responsibility for them? Since in the database of the combiner there are no records of proxy signers' partital and individual signatures corresponding to them, either the original singer or the proxy signers can deny having signed them. So we conclude that the HLL scheme does not meet the security requirement of non-repudiation.

(4) **Known Signer**. Even for internal auditing, the combiner should be trusted by all proxy singers and the original signer; otherwise, a proxy singer's standard signature $s_i^{d_i} \bmod N_i$ on s_i can be altered, replaced, or deleted by the combiner. The point is that we cannot assume that the combiner is a trusted party because of the following

two reasons. On the one hand, maintaining such a trusted party is very costly in real word. On the other hand, such assumption is in contrary to the original motivation to reduce the original signer's trust on individual proxy signers. If the combiner is a trusted party, why not just treat the combiner as a single proxy signer instead of exploiting threshold scheme? Therefore, the combiner has only limited merit for supporting the security of the HLL scheme.

(5) **Unforgeability**. Moreover, using similar strategy as mentioned above, it is even possible for an outsider to mount a universal forgery attack. The reason is that E, e_0, m_w are publicly known values, so $e_0^{m_w}$ can be computed in the integer ring $\mathbb{Z}$ by anyone. Eq. (2) implies that $e_0^{m_w} - E$ is also a multiple of $\phi(N_0)$. So, an outsider can also factor the original signer's RSA modulus N_0. However, we have to note that this attack may be significantly time consuming than that by proxy signers because the integer $e_0^{m_w}$ is much larger than ED.

3.2 Correctness

We now point out some design errors in the HLL scheme. First of all, we notice that a proxy signer P_i cannot compute the coefficient L_i using Eq. (5). There are two reasons for this: (a) Since 4 is a factor of $\phi(N_0)$, the inverse of $(i-j)$ modulo $\phi(N_0)$ may not exist; and (b) Even if $(i-j)^{-1} \bmod \phi(N_0)$ exists, P_i cannot carry out its value because he does not know $\phi(N_0)$, which is a secret of the original signer. Therefore, the HLL scheme essentially does not work. We note that using the technology introduced by Shoup in [22], the HLL scheme can be adapted to satisfy the correctness. However, we do not provide details about this improvement since the above mentioned security weaknesses still remain.

In addition, when a verifier validates a threshold proxy signature, he does not need to compute m_w and E since they are public parameters. What he needs to do is to check whether (m_w, E) is a valid pair of warrant and proxy public key certified by the original signer. To this end, he checks whether $m_w||E \equiv ([m_w||E]^{d_0} \bmod N_0)^{e_0} \bmod N_0$. If this equality holds, the verifier accepts E as a valid proxy public key, and uses it to check the validity of a proxy signature according to Eq. (7).

4. Conclusion

In this paper, we presented a security analysis of the threshold proxy signature scheme proposed recently in [7]. Our results show that this scheme is insecure, which is contrary to the original authors' conclusion. We remark that it is still an open problem to construct efficient and secure RSA-based threshold proxy signature schemes.

References

1. A. Boldyreva, A. Palacio, and B. Warinschi, "Secure Proxy Signature Schemes for Delegation of Signing Rights," Available at `http://eprint.iacr.org/2003/096/`.
2. Y. Desmedt and Y. Frankel, "Threshold Cryptosystems," *Proc. Advance in Cryptology - CRYPTO'89*, LNCS 435, Springer-Verlag, pp. 307-315, 1989.

3. Y. Dodis and L. Reyzin, “Breaking and Repairing Optimistic Fair Exchange from PODC 2003,” *Proc. ACM Workshop on Digital Rights Management* (*DRM'03*), ACM press, pp. 47-54, 2003.
4. T. ElGamal, “A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms,” *IEEE Trans. Information Theory*, vol. 31, no. 4, pp. 469-472, July 1985.
5. C.-L. Hsu, T.-S. Wu, and T.-C. Wu, “New Nonrepudiable Threshold Proxy Signature Scheme With Known Signers,” *The Journal of Systems and Software*, vol. 58, no. 5, 119-124, 2001.
6. M.-S. Hwang, I.-C. Lin, and E. J.-L. Lu, “A Secure Nonrepudiable Threshold Proxy Signature Scheme with Known Signers”, *Intl J. Informatica*, vol. 11, no. 2, pp. 1-8, 2000.
7. M.-S. Hwang, E. J.-L. Lu, and I.-C. Lin, ”A Practical (t, n) Threshold Proxy Signature Scheme Based on the RSA Cryptosystem,” *IEEE Trans. Knowledge and Data Engineering*, vol. 15, no. 6, pp. 1552-1560, 2003.
8. S. Kim, S. Park, and D. Won, “Proxy Signatures, Revisited,” *Proc. Information and Communications Security* (*ICICS'97*), LNCS 1334, Springer-Verlag, pp. 223-232, 1997.
9. N. Koblitz, *A Course in Number Theory and Cryptography*, Springer- Verlag, 1994.
10. B. Lee, H. Kim, and K. Kim, “Secure Mobile Agent Using Strong Non-Designated Proxy Signature,” *Proc. Information Security and Privacy* (*ACISP'01*), LNCS 2119, Springer-Verlag, pp. 474-486, 2001.
11. J.-Y. Lee, J. H. Cheon, and S. Kim, “An Analysis of Proxy Signatures: Is a Secure Channel Necessary?” *Proc. Topics in Cryptology* (*CT-RSA 2003*), LNCS 2612, Springer-Verlag, pp. 68-79, 2003.
12. M. Mambo, K. Usuda, and E. Okamoto, “Proxy signature: Delegation of the Power to Sign Messages,” *IEICE Trans. Fundamentals*, vol. E79-A, no. 9, pp. 1338-1353, Sep. 1996.
13. M. Mambo, K. Usuda, E. Okamoto, “Proxy Signatures for Delegating Signing Operation',’. *Proc. 3rd ACM Conference on Computer and Communications Security* (*CCS'96*), ACM Press, pp. 48-57, 1996.
14. J. M. Park, E. Chong, H. Siegel, and I. Ray, “Constructing Fair Exchange Protocols for E-Commerce via Distributed Computation of RSA Signatures,” *Proc. 22th Annual ACM Symp. on Principles of Distributed Computing* (*PODC'03*), ACM Press, pp. 172181, 2003.
15. T. P. Pedersen, “Distributed Provers with Applications to Undeniable Signatures,” *Proc. Advance in Cryptology - EUROCRYPT'91*, LNCS 547, Springer-Verlag, pp. 221-242, 1991.
16. T.P. Pedersen, “A Threshold Cryptosystem without a Trusted Party,” *Proc. Advance in Cryptology - EUROCRYPT'91*, LNCS 547, Springer-Verlag, pp. 522-526, 1991.
17. R.L. Rivest, A. Shamir, and L.M. Adleman, “A Method for Obtaining Digital Signatures and Public-Key Cryptosystems,” *Communications of the ACM*, vol. 21, No. 2, pp. 120-126, Feb. 1978.
18. H.-M. Sun, “An Efficient Nonrepudiable Threshold Proxy Signature Scheme with Known Signers,” *Computer Communications*, vol. 22, no. 8, pp. 717-722, 1999.
19. H.-M. Sun, “Threshold Proxy Signatures,” *IEE Proc.-Computers & Digital Techniques*, vol. 146, no. 5, pp. 259-263, Sept. 1999.
20. C. Schnorr, “Efficient Signature Generation by Smart Cards,” *Journal of Cryptography*, vol. 4, no. 3, pp. 161-174, 1991.
21. A. Shamir, “How to Share a Secret,” *Communications of the ACM*, vol. 22, no. 11, pp. 612-613, 1979.
22. V. Shoup, “Practical Threshold Signatures,” *Proc. Advance in Cryptology - EUROCRYPT 2000*, LNCS 1807, Springer-Verlag, pp. 207-220, 2000.
23. C.-S. Tsai, S.-F. Tseng, and M.-S. Hwang, “Improved Non-Repudiable Threshold Proxy Signature Scheme With Known Signers,” *Intl J. Informatica*, vol. 14, no. 3, pp. 393-402, 2003.
24. G. Wang, F. Bao, J. Zhou, and R. H. Deng, “Security Analysis of Some Proxy Signatures,” *Proc. Information Security and Cryptology* (*ICISC'03*), Springer-Verlag, 2004 (to appear). Preliminary version is available at `http://eprint.iacr.org/2003/196.`
25. K. Zhang, “Threshold Proxy Signature Schemes,” *Proc. Information Security Workshop* (*ISW'97*), LNCS 1396, Springer-Verlag, pp. 282-290, 1997.