@Copyright GFCR
TRANSACTION ON CRYPTOLOGY Volume 2- Issue 1(2005) Pages: 5 - 11

Proxy Blind Signature Scheme

"Revised Version"

Amit K Awasthi
Hindustan College of Sc. & Tech.,
Farah Mathura, INDIA
Email: awasthi_hcst@yahoo.com

Sunder Lal
Institute of Basic Science,
Dr. B. R. A. University, Agra, INDIA
Email: sunderlal_2@rediffmail.com

Abstract

Blind signature is the concept to ensure anonymity of e-coins. Untracebility and unlinkability are two main properties of real coins, which require mimicking electronically. Whenever a user is permitted to spend an e-coin, he is in need to fulfill above requirements of blind signature. In this paper a proxy blind signature scheme is given with which a proxy is able to make proxy blind signature which verifier is able to verify in a way similar to proxy signature schemes.[1]

Keywords: *Proxy Signature, Blind Signature, Proxy-Blind, e-Coin*

[1]This research is partially supported by U. G. C. Grant No. 8 9/98 (SR-I)

1 Introduction

D. Chaum [3]introduced the concept of a blind signature scheme in 1982. Using this scheme a user A can obtain the signature of B on any given message, without revealing any in formation about the message or its signature. Apart from unforgeability, the scheme ensures untracebility and unlinkability. A lot of work has been done in field of blind signature schemes since Chaum. [3, 2, 6, 1]

In production of coins, the user makes the bank blindly sign a coin using blind signature schemes. The user is in possession of a valid coin such that the bank itself cannot recognize nor link with the user. Whenever a user goes through a valid branch to withdraw a coin, he needs the branch to make proxy blind signature on behalf of the signee bank. This application leads to the need of proxy blind signature schemes.

In 1996 Mambo et al [4] introduced the concept of proxy signature. In this scheme an original signer delegates his signing authority to another (*proxy*) signer in such a way that the proxy signer can sign any message on behalf of the original signer and the verifier can verify and distinguish between normal (original) signature and proxy signature. He also elaborated the two types of scheme: proxy unprotected (proxy and original signer both can generate a valid *proxy* signature) and proxy protected (only proxy can generate a valid *proxy* signature). These schemes ensures among other things, non-repudiation and unforgeability.

Recently Tan et al. [6] introduced a proxy blind signature scheme, which ensures security properties of the schemes, viz., the blind signature schemes and the proxy signature schemes. The scheme is based on Schnorr blind signature scheme. In this paper we introduce a new proxy blind signature scheme, which is based on Mambo et al., Our scheme is computationally more efficient than that of Tan et al. We also discuss a few attacks on the Tan et al scheme and show that these can be overcome using our proposed scheme.

2 The Scheme

In the proposed scheme the system parameters and some notations are
p : a large prime number
q : a large prime factor of $(p-1)$
g : an element of Z_p^* of order q
x_A: the secret key of the original signer A
y_A: the public key of the original signer A, where $y_A = g^{x_A} \bmod p$
$h(.)$: a secure one way hash function

2.1 Proxy unprotected case our protocol runs as follows

2.1.1 Proxy phase

1. (Proxy Generation) The original signer A randomly chooses $k \in Z_q^*, k \neq 1$ and computes

$$r = g^k \bmod p, \tag{1}$$

$$s = x_A + k \cdot r \bmod q, \tag{2}$$

and

$$y_p = g^s \bmod p \tag{3}$$

2. (Proxy Delivery) The original signer sends (s, r) to a proxy signer B in a secure way and makes y_p public.

3. (Proxy Verification)After receiving the secret key (s, r) the proxy signer B checks the validity of the secret key with the following congruence

$$y_p = g^s = y_A \cdot r^r \bmod p \tag{4}$$

If (s, r) satisfies this congruence, he accepts it as a valid proxy, otherwise rejects it. In the later case he either requests for another key, or simply stops the protocol.

2.1.2 Signing phase

1. B chooses a random number $K \in Z_q^*, K \neq 1$, computes R = $g^K \bmod p$ and sends it to the receiver C.

2. (a)C chooses randomly $\alpha, \beta \in Z_q^*$ and computes

$$r^{'} = R \cdot g^{-\alpha} \cdot {y_p}^{-\beta} \bmod p \tag{5}$$

If $r^{'} = 0$, he chooses another set of $\alpha, \beta \in Z_q^*$ and ; otherwise computes

$$e^{'} = h(r^{'}, m) \bmod q \tag{6}$$

$$e = e^{'} + \beta \bmod q \tag{7}$$

and C sends e to B.

3. After receiving e, B computes

$$s^{'} = K - s \cdot e \bmod q \tag{8}$$

and sends it to C.

4. Now C computes

$$S_p = s^{'} + \alpha \bmod q \tag{9}$$

The tuple $(m, S_p, e^{'})$is the proxy blind signature.

2.1.3 Verification Phase

The verifier or recipient of the proxy blind signature computes

$$e^{''} = h(g^{S_p} \cdot y_p^{e^{'}} \bmod p) \oplus m) \bmod p \tag{10}$$

where y_p is the public value of step 1 in (2.1.2) Here $e^{''} = e^{'}$, if and only if the tuple (m, S_p, e) is a valid proxy signature.

Table 1: Comparison of computational load of our scheme vs. Tan et al.

Scheme	Phase			Total
	Proxy Generation	Signature Generation	Signature Verification	
Tan et al	3E+2M	$8E + 7M + 4I$	$3E + 3M + I$	$14E + 12M + 5I$
Proposed Scheme	$4E + 2M$	$3E + 3M + 2I$	$2E + M$	$9E + 6M + 2I$

2.2 Proxy protected

If we want only proxy signer to generate a valid proxy signature, we modify the proxy phase (2.1.2) of the previous protocol as follows:

2.2.1 Proxy phase

1. (Proxy Generation) The original signer A randomly chooses $k \in Z_q^*, k \neq 1$ and computes r = g^k mod p, $\sigma = x_A + k \cdot r$ mod q, and $y_p = g^{y^B}$ mod p, where $y_B = g^{x_B}$ mod p, is the public key of B.

2. (Proxy Delivery) The original signer sends (σ, r) to a proxy signer B in a secure way and makes y_p public.

3. (Proxy verification and key alteration) After confirming the validity of the pair (σ, r) B alters the proxy key as $s = \sigma + x_B$ mod q

2.2.2 The Signing Phase and The Verification Phase

: Same as in the proxy unprotected case.

3 Efficiency

In this section we show the efficiency of our scheme over that of Tan et al. Let E, M and I respectively denote the computational load for exponentiation, multiplication and inversion. Then following table shows the comparison of computational load of our scheme vs. Tan et al.

Each phase in our scheme has less computational load except in proxy generation phase, where it is one exponential computation more than tan et al. This computational load may be adjusted with compromise that some computational load in verification phase increases. In some applications digital information is signed once but verified more than once. In such situation the efficiency of our scheme increases with the number of times verification is done. Further the total computation cost in our scheme is $9E + 6M + 2I$ as compared to Tan et al which is $14E + 12M + 5I$. Thus, our scheme has computational advantage over that of Tan et al.

4 Security Analysis

1. In signature verification phase we use different congruence to check the validity of the original signatures and the proxy signatures. So the original signature is distinguishable from the proxy signature.
2. To put a valid proxy signature s (in case proxy protected x_B too) is required. It is impossible to create a valid signature with out knowing x_B or s or both. Thus proxy signature cannot be forged. Furthermore, though original signer creates s, also have no knowledge about x_B in case of proxy protected. Thus the proxy signer cannot deny the proxy signature that he has created.
3. The public key y_p is computed from the original signer's public key y_A. thus the original signer cannot deny his agreement. Proxy signer's public key is also involved in the public key (in case proxy protected). Therefore the proxy signer can be identified from the signature.

Some security attacks that work in Tan's scheme, have also been removed in our scheme. These attacks are as follows:–

The verification equation in Tan et al's scheme is

$$e = h(g^S \cdot y_B^{-e} \cdot y_A^e \cdot u \parallel m) \bmod q \tag{11}$$

which ensure the participation of both the signers A and B and hence the tuple (m, u, s, e) [6, Section 3.4] is a valid proxy blind signature by B on behalf of A. Here involvement of both signers public key is the only way to recognize that it is a proxy blind signature of signer B for A. Here a forgery by R may be possible in Tan et al scheme.

- The receiver R may prove that (m, u, s, e) is a valid proxy blind signature of some other signer F although F might have not given his signing authority to any one. It may happen as follows When the receiver R interacts with 'B'. he computes

  $$u' = (\bar{r} \cdot y_A^{\bar{r}})^{-e+b} \cdot y_F^{-e} \bmod q \tag{12}$$

  instead of

  $$u = (\bar{r} \cdot y_A^{\bar{r}})^{-e+b} \cdot y_A^{-e} \bmod q \tag{13}$$

  No other equation would be affected by this forgery. Now the receiver may prove that the tuple (m, u, s, e) is a valid proxy blind signature of signer F by the similar verification equation as (11)

  $$e = h(g^S \cdot y_B^{-e} \cdot y_F^e \cdot u \parallel m) \bmod q \tag{14}$$

  which ensure the participation of the signer F in that blind signature.

- In Second case, the receiver may prove that a signer D had produced a valid proxy blind signature on behalf of A during verification. For this he computes

  $$u' = (\bar{r} \cdot y_A^{\bar{r}})^{-e+b} \cdot y_A^{-e} \cdot y_B^{-e} \cdot y_D^{-} \bmod q \tag{15}$$

  instead of eq (13) and thus verification equation eq (11) changes as

  $$e = h(g^S \cdot y_D^{-e} \cdot y_A^e \cdot u \parallel m) \bmod q \tag{16}$$

  which ensure that tuple (m, u, s, e) is a valid proxy blind signature by the signer D on behalf of signer A, although A never delegated his proxy key to D.

To overcome these forgeries either freeness of the u should be restricted or it should be removed from the computation altogether. In our equation u does not appear and hence our scheme is secure for these types of forgeries.

5 Conclusion

In this paper we propose a proxy blind signature scheme with which a proxy user is able to make proxy blind signature and verifier may verify it very similar to proxy signature schemes. Our scheme is based on Mambo et al's protocols. Its computational load is less than that of a recent scheme by Tan et al. In this paper, we also had discussed some possible attacks on Tan's scheme and which our proposed scheme is free from.

6 Further Remarks

Recently Sun and Hsieh [5] showed that our above said scheme has some security flaws. According to their view

6.1 On the Unlikability

For the proxy signer, in order to identify the relationship between the revealed messages and the blind information, the proxy signer records all messages he owned, such as R, e, s'. After a signature (m, s, e') is revealed the proxy signer computes $\alpha' = s' - s, \beta' = e - e'$ and $r' = g^s y_p^{e'} \bmod p$ for some $s' \in < s' >$ and $e \in < e >$. Finally, the proxy signer checks the equation $r' = Rg^{-\alpha'} y_p^{-\beta'} \bmod p$ for some $t \in < t >$. If he find a corresponding t such that $r' = Rg^{-\alpha'} y_p^{-\beta'} \bmod p$, therefore, the proxy signer knows that (t, e, s) is the related blind information corresponding to the revealed message m. Thus our scheme does not posses the unlinkability. But we are not agree on this point as whatever message tuple intruder has recorded, the verification equation holds for all.

6.2 On the Publishing

In general, in order to verify a proxy signature, the proxy public key is obtained by computing, while not retrieving from the public keys from the original signers. The computed proxy public key has the meaning of confirming the relationship between a original signer and a proxy signer. According to Sun and Hsieh in our scheme such publishing enables an adversary who obtained the proxy public key to republish it again. Finally, the adversary claims that he is the original signer. Therefore, the publishing of proxy public key suffers from the security flaw that the original signer is unable to be authenticated exactly.

6.2.1 Our View

The observation of the above section is partially correct and may be revised. Whenever an adversary republishes the proxy public key y_p, Proxy delegation protocol returns verification failure and hence that change is caught. Another chance is that the proxy signer may help to adversary. This is only if proxy signer is agree to change the proxy secret key, which shows that final proxy signature is not on behalf of original signer and may be denied easily be the real original signer. Also, we showed in section 3, that proxy signature verification may be done with original signer's public key y_o with compromise of 1 multiplication and 1 exponential cost.

References

[1] J. L. Camenisch, J.-M. Piveteau, and M. A. Stadler, *Blind signatures based on the discrete logarithm problem*, Lecture Notes in Computer Science **950** (1995), 428–432.

[2] D. Chaum, *Untraceable electronic mail, return addresses, and digital pseudonyms*, Communications of the Association for Computing Machinery **24** (1981), no. 2, 84–88.

[3] ———, *Blind signatures for untraceable payments*, Advances in Cryptology Crypto 82 **Plenum Press** (1982), 199–203.

[4] M. Mambo, K. Usuda, and E. Okamoto, *Proxy signatures: Delegation of the power to sign messages*, IEICE Trans. Fundamentals **E79-A**, no. 9.

[5] Hung-Min Sun and Bin-Tsan Hsieh, *On the security of some proxy blind signature schemes*, CRPIT '32: Proceedings of the second workshop on Australasian information security, Data Mining and Web Intelligence, and Software Internationalisation, Australian Computer Society, Inc., 2004, pp. 75–78.

[6] Z. Tan, Z. Liu, and C. Tang, *Digital proxy blind signature schemes based on dlp and ecdlp*, MM Research Preprints, MMRC, AMSS, Academia, Sinica, Beijing (2002), no. No. 21, 212–217.