Improved Proxy Signature Scheme without Bilinear Pairings

Sahadeo Padhye and Namita Tiwari

Department of Mathematics
Motilal Nehru National Institute of Technology
Allahabad-211004, India
{sahadeomathrsu,namita.mnnit}@gmail.com

Abstract. Proxy signature is an active research area in cryptography. In order to save the running time and the size of the signature, recently a provable secure proxy signature scheme without bilinear pairings has been proposed which is based on elliptic curve discrete log problem (ECDLP). In this paper, we point out some forgery attacks and security issues on this scheme. Furthermore, we also improve the scheme to make it secure against these forgeries. Our scheme is as efficient as previous proposed scheme.

Keywords: Digital signature, Proxy signature, Bilinear pairings, Elliptic curve discrete log problem.

1 Introduction

Digital signature offers source authentication in cryptography. To handle the situations arisen in digital world related to authentication, different types of digital signatures have been developed e.g a manager want to delegate his secretaries to sign documents without giving his private signing key, while he is on vacation. Proxy signature is the solution of such problem and firstly introduced by Mambo et al [11] in 1996. Proxy signature schemes can also be used in electronic transactions and mobile agent environment [10]. Since the proxy signature appears, it attracts many researcher's great attention. Using bilinear pairings, people proposed many proxy signature schemes [6,7,9,15,16,17]. All the above schemes are very practical, but they are based on bilinear pairings and the pairing is regarded as one of the expensive cryptography primitive. Therefore, to save the running time and to reduce the size of the signature, recently a provable secure proxy signature scheme without bilinear pairings [14] has been proposed which is based on ECDLP. In this paper, we point out some forgery attacks and security issues on this scheme. We show that scheme [14] does not satisfy prevention of misuse property. It has some other drawbacks also. Furthermore, we improve the scheme against these forgeries. Our improved scheme is as efficient as previous proposed scheme [14].

K. Singh, A.K. Awasthi, and R. Mishra (Eds.): QSHINE 2013, LNICST 115, pp. 682–688, 2013.

Roadmap: The rest of this paper is organized as follows. Some preliminary works are given in the section *2*. A brief review of scheme [14] is presented in section *3*. We discuss the forgeries on scheme [14] and present it's improved version in section *4* and *5* respectively. Section *6* presents the comparative analysis. Finally, conclusions are given in Section *7*.

2 Preliminaries

2.1 Background of Elliptic Curve Group

Let the symbol E/F_p denote an elliptic curve E over a prime finite field F_p, defined by an equation

$$y^2 = x^3 + ax + b,\ a, b \in F_p, \quad \text{and}$$

discriminant $\Delta = 4a^3 + 27b^2 \neq 0$

The points on E/F_p together with an extra point O called the point at infinity form a group $G = \{(x, y) : x, y \in F_p, E(x, y) = 0\} \cup \{O\}$.

Let the order of G be n. G is a cyclic additive group under the point addition "+" defined as follows: Let $P, Q \in G$, l be the line containing P and Q (tangent line to E/F_p if $P = Q$), and R, the third point of intersection of l with E/F_p. Let l' be the line connecting R and O. Then $P + Q$ is the point such that l' intersects E/F_p at R and O and $P + Q$.

Scalar multiplication over E/F_p can be computed as follows:

$$tP = \underbrace{P + P + ... + P}(t\ times).$$

2.2 Complexity Assumption

The following problem defined over G are assumed to be intractable within polynomial time.

Elliptic curve discrete logarithm problem (ECDLP): For $x \in_R {Z_n}^*$ and P the generator of G , given $Q = x.P$ compute x.

3 Brief Review of Scheme [14]

- Setup: Takes a security parameter k, and returns system parameters
  $\Omega = \{F_p, E/F_p, G, P, H_1, H_2, H_3\}$ as defined in Section 2.
  $H_1 : \{0, 1\}^* \to Z_n^*$, $H_2 : \{0, 1\}^* \times G \to Z_p^*$ and $H_3 : \{0, 1\}^* \to Z_p^*$ are three cryptographic secure hash functions.
- Extract: Each signer picks at random $sk_i \in Z_n^*$ and computes $pk_i = sk_iP$. Thus $(sk_i, pk_i), i \in \{o, p\}$ is private-public key pair.
- DelGen: This algorithm takes O's secret key sk_o and a warrant m_w as input, and outputs the delegation $W_{O\to P}$ as follows:
  a. Generates a random $a \in Z_n^*$ and computes $K = aP$.
  b. Computes $h_1 = H_2(m_w, pk_p)$ and $\sigma = h_1 sk_o + a \bmod n$.
  O sends the delegation $W_{O\to P} = \{pk_o, m_w, K, \sigma\}$ to proxy signer P.

- DelVerif: To verify the delegation $W_{O \to P}$ and message warrant m_w, proxy signer P first computes
  $h_1 = H_2(m_w, p_{k_p})$, then checks whether
  $\sigma P = h_1 pk_o + K$.
- PKGen: If P accepts the delegation $W_{O \to P}$, he computes the proxy signing key sk_{pr} as:
  $sk_{pr} = \sigma h_2 + sk_p \bmod n$, where $h_2 = H_3(m_w)$.
- PSign: Takes system parameters, the proxy signing key sk_{pr} and a message m as inputs, returns a signature of the message m. The user P does as follows.
  a. Chooses at random $b \in Z_n^*$ and computes $R = hbP$, where $h = H_1(m)$.
  b. Computes $s = hb + sk_{pr} \bmod n$.
  The resulting signature is $(pk_o, pk_p, m_w, K, m, R, s)$.
- PSVerif: To check whether the signature $(pk_o, pk_p, m_w, K, m, R, s)$ is a valid proxy signature on message m under warrant m_w, verifier V first checks if the proxy signer and the message conform to m_w and computes $h_1 = H_2(m_w, pk_p), h_2 = H_3(m_w), h = H_1(m)$ then verify whether the following equation holds.
  $sP = R + [(h_1 pk_o + K)h_2 + pk_p]$.

4 Security Analysis of Scheme [14]

In this section, we will demonstrate that the scheme [14] has some drawbacks. As the first drawback, a forgery is given by malicious signer who is not designated as a proxy signer by the original signer. However, the malicious signer can forge a valid proxy signature on any message.

4.1 Forgery by Proxy Signer

After having the delegation σ on warrant message m_w, proxy signer makes the following forgery as follows:

- Chooses another warrant $m_w{}'$, computes $h_1{}' = H_2(m_w{}', pk_p)$.
- Computes $\sigma' = (\frac{h_1{}'}{h_1})\sigma$ s.t. $\sigma' = h_1' s_{k_o} + ah_1' h_1^{-1}$ $(mod\ n)$.
- This generated σ' satisfies $\sigma' P = h_1{}' pk_o + K'$ where $K' = ah_1' h_1^{-1} P$. So $(pk_o, m_w{}', K', \sigma')$ is a valid delegation on new warrant $m_w{}'$. Using this delegation, proxy signer can sign any message of it's own choice.

Thus proxy signer can misuse the right of delegation. Our attack is possible only because public parameter K lonely exist in the delegation verification equation in the form of bases. Similarly, parameter R is also used in the proxy signature verification equation in the form of bases. As a result, some other forgeries also may be possible. To avoid such attacks, verification equations would be so complicated, that no such attacks would be possible. As a modification, we will hash K with (m_w, pk_p) as hash query $H(m_w, pk_p, K)$ and R with m as hash query

$H(m, R)$ in the improved scheme. One more thing is to observe that message m is not used in the verification equation so given proxy signature is a valid proxy signature for any chosen message. We will remove also this flaw of the scheme [14] in the improved version.

5 Improved Proxy Signature Scheme

In this section, we present the improvements on provable secure proxy signature scheme without using pairings [14].

- Setup: System parameters are generated in the same manner as in scheme [14] only with a slight change in hash functions $H_1, H_2 : \{0,1\}^* \times G \to Z_n^*$ and $H_3 : \{0,1\}^* \to Z_n^*$.
- Extract: Private-public key pair are generated in the same way as in the scheme [14].
- DelGen: This algorithm takes O's secret key sk_o and a warrant m_w as inputs, and outputs the delegation $W_{O\to P}$ as follows:
  a. Generates a random $a \in Z_n^*$ and computes $K = aP$.
  b. Computes $h_1 = H_2(m_w, pk_p, K)$ and $\sigma = (h_1 sk_o + a) \bmod n$.
  O sends the delegation $W_{O\to P} = \{pk_o, m_w, K, \sigma\}$ to proxy signer P.
- DelVerif: To verify the delegation $W_{O\to P}$ and message warrant m_w, proxy signer P first computes
  $h_1 = H_2(m_w, pk_p, K)$, then checks whether
  $\sigma P = h_1 pk_o + K$.
  Accepts if it is equal, otherwise rejects.
- PKGen: If P accepts the delegation $W_{O\to P}$, he computes the proxy signing key $sk_{pr} = (\sigma h_2 + sk_p) \bmod n$, where $h_2 = H_3(m_w)$.
- PSign: Takes system parameters, the proxy signing key sk_{pr} and a message m as inputs, returns a signature of the message m. The user P does as follows.
  a. Chooses at random $b \in Z_n^*$ and computes $R = bP$.
  b. Computes $s = hb + sk_{pr} \pmod n$, where $h = H_1(m, R)$.
  The resulting signature is $(pk_o, pk_p, m_w, K, m, R, s)$.
- PSVerif: To check whether the signature $(pk_o, pk_p, m_w, K, m, R, s)$ is a valid proxy signature on message m under warrant m_w, verifier V first checks if the proxy signer and the message conform to m_w and computes $h_1 = H_2(m_w, pk_p, K), h_2 = H_3(m_w), h = H_1(m, R)$ then verify whether the following equation holds.
  $sP = hR + [(h_1 pk_o + K)h_2 + pk_p]$.
  If the equality holds, Verifier V accepts the signature, otherwise rejects it.

Correctness:
Since $R = bP$, $s = (hb + sk_{pr}) \bmod n$, we have

$$\begin{aligned} sP &= (hb + sk_{pr})P \\ &= hR + [(\sigma P)h_2 + sk_p P] \\ &= hR + [(K + h_1 pk_o)h_2 + pk_p]. \end{aligned}$$

5.1 Security Analysis

We analyze the security of our scheme as follows.

Distinguishability. The proposed proxy signature $(pk_o, pk_p, m_w, K, m, R, s)$ contains the warrant m_w while the normal signature does not, so both are different in the form. Also in the verification equation, public keys pk_o, pk_p and warrant m_w are used. So anyone can distinguish the proxy signature from normal signature easily.

Verifiability. The verifier of proxy signature can check easily that the verification equation $sP = hR + [(h_1 pk_o + K)h_2 + pk_p]$ holds. In addition, this equation involves original signer's public key pk_o and warrant m_w, so anyone can be convinced of the original signer's agreement on the proxy signer.

Unforgeability. In our scheme only the designated proxy signer can create a valid proxy signature, since proxy private key $sk_{pr} = (\sigma h_2 + sk_p) \bmod n$ includes the private key sk_p of proxy signer and to compute sk_p is equivalent to solve ECDLP.

Nonrepudiation. As in the verification equation warrant m_w and public keys pk_o, pk_p are used. Also generation of proxy signature needs original and proxy signer's private key sk_o, sk_p respectively. It is already proved that neither the original signer nor the proxy signer can sign in place of other party. So the original signer can not deny his delegation and proxy signer can not deny having signed the message m on behalf of original signer to other party.

Identifiability. In the proposed scheme, it can be checked who is original signer and who is proxy signer from warrant m_w. Also seeing from the verification equation $sP = hR + [(h_1 pk_o + K)h_2 + pk_p] \bmod n$, the public keys pk_o, pk_p are asymmetrical in position. So anyone can distinguish the identity of proxy signer from proxy signature.

Prevention of Misuse. Original signer generates the delegation (pk_o, m_w, K, σ) using its private key and sends to P. So the delegation can not be modified or forged. Also it is not possible for proxy signer P to transfer his proxy power to other party unless he provides proxy private key sk_p. In addition, warrant m_w contains the limit of delegated signing capability. So it is not possible to sign the messages that have not been authorized by original signer.

6 Efficiency Comparison

Here, we compare the efficiency of our scheme with similar signature scheme [15] and show that our scheme is more efficient in computational and timing (total operation time) sense than existing scheme. We compare the total number of bilinear pairings, map-to-point hash functions (H), pairing-based scalar multiplications, elliptic curve-based scalar multiplications and consequently the total operation time in overall signature process. We also note that the operation time for one pairing computation is 20.04 milliseconds, one map-to-point hash function is 3.04 milliseconds, one pairing-based scalar multiplication 6.38 milliseconds and one ECC-based scalar multiplication 2.21 milliseconds [8]. In the following tables, we have omitted the operation time due to a general hash function, as it takes ≤ 0.001 milliseconds [8]. For the computation of operation time, we refer [8] where the operation time for various cryptographic operations have been obtained using MIRACAL [13], a standard cryptographic library, and the hardware platform is a PIV 3 GHZ processor with 512 M bytes memory and the Windows XP operating system. For the pairing-based scheme, to achieve the 1,024-bit RSA level security, Tate pairing defined over the supersingular elliptic curve $E = F_p : y^2 = x^3 + x$ with embedding degree 2 has been used, where q is a 160-bit Solinas prime $q = 2^{159} + 2^{17} + 1$ and p a 512-bit prime satisfying $p + 1 = 12qr$. For the ECC-based schemes, to achieve the same security level, the parameter secp160r1 [12], recommended by the Certicom Corporation has been employed, where $p = 2^{160} - 2^{31} - 1$.

Table 1. Computational Cost Comparison

Scheme	Extract	DelGen	DelVerif	PKgen
Scheme [15]	$1M_P$	$1M_P + 1H_M$	$1H_M + 2O_P$	$1M_p$
Our scheme	$1M_E$	$1M_E$	$2M_E$	$0M_E$

Scheme	PSign	PSVerif	Total
Scheme [15]	$3M_P$	$1M_p + 1H_M + 3O_P$	$7M_P + 3H_M + 5O_P$
Our scheme	$1M_E$	$3M_E$	$8M_E$

Table 2. Running Time Comparison(in ms)

Scheme	Extract	DelGen	DelVerif	PKGen	PSign	PSVerif	Total
Scheme [15]	6.38	9.42	43.12	6.38	19.14	69.54	153.98
Our scheme	2.21	2.21	4.41	≈ 0	2.21	6.63	17.68

According to these running time computations, the running time of our proxy signature algorithm is 11.54% of scheme [15]'s algorithm and total running time of our scheme is 11.48% of the scheme [15].

If we use the running time computation results obtained by Cao and Kou [2] in different environment then efficiency of our scheme can be improved as given in the following table.

Table 3. Running Time Comparison(in ms)

Scheme	Extract	DelGen	DelVerif	PKgen	PSign	PSVerif	Total
Scheme [15]	6.38	9.42	43.12	6.38	19.14	69.54	153.98
Our scheme	0.83	0.83	1.66	≈ 0	0.83	2.49	6.64

According to these running time computations, the running time of our proxy signature algorithm is 4.33% of scheme [15]'s algorithm and total running time of our scheme is 4.31% of the scheme [15].

7 Conclusion

In this paper, we demonstrated that previously proposed scheme [14] has some security flaws. Furthermore, we presented an improved proxy signature scheme without pairing which removes these flaws. Our improved scheme is as efficient as [14].

References

1. Chen, L., Cheng, Z., Smart, N.: Identity-based key agreement protocols from pairings. Int. J. Inf. Secur. (6), 213–241 (2007)
2. Cao, X., Kou, W.: A Pairing-free Identity-based Authenticated Key Agreement Protocol with Minimal Message Exchanges. Information Sciences (2010), doi:10.1016/j.ins.2010.04.002
3. David, P., Jacque, S.: Security arguments for digital signatures and blind signatures. J. Cryptol. 13(3), 361–396 (2000)
4. Goldwasser, S., Micali, S., Rivest, R.: A digital signature scheme secure against adaptive chosenmessage attacks. SIAM J. Comput. 17(2), 281–308 (1988)
5. Granger, R., Page, D., Smart, N.P.: High security pairing-based cryptography revisited. In: Algorit. Numb. Theo. Sympo. VII, pp. 480–494 (2006)
6. Gu, C., Zhu, Y.: Provable security of ID-based proxy signature schemes. In: Lu, X., Zhao, W. (eds.) ICCNMC 2005. LNCS, vol. 3619, pp. 1277–1286. Springer, Heidelberg (2005)
7. Gu, C., Zhu, Y.: An efficient ID-based proxy signature scheme from pairings. In: Pei, D., Yung, M., Lin, D., Wu, C. (eds.) Inscrypt 2007. LNCS, vol. 4990, pp. 40–50. Springer, Heidelberg (2008)
8. He, D., Chen, J., Hu, J.: An ID-Based proxy signature schemes without bilinear pairings. Anna Telicom (2011), doi:10.1007/s12243-011-0244-0
9. Ji, H., Han, W., Zhao, L., et al.: An identity-based proxy signature from bilinear pairings. In: 2009 WASE International Conference on Information Engineering, pp. 14–17 (2009)
10. Kim, H., Baek, J., Lee, B., Kim, K.: Secret computation with secrets for mobile agent using one-time proxy signature. In: Cryptog. and Infor. Secur., Canada, pp. 307–312 (2001)
11. Mambo, M., Usuda, K., Okamoto, E.: Proxy signatures: Delegation of the power to sign message. IEICE Transactions Fundamentals E79-A(9), 1338–1353 (1996)
12. The Certicom Corporation, SEC 2:Recommended Elliptic Curve Domain Parameters, `http://www.secg.org/collateral/sec2_final.pdf`
13. Shamus Software Ltd., Miracl library, `http://www.shamus.ie/index.php?page=home`
14. Tiwari, N., Padhye, S.: Provable secure proxy signature scheme without bilinear pairings. Int. J. Commun. Syst. (2011), doi:10.1002/dac.1367
15. Wang, A., Li, J., Wang, Z.: A provably secure proxy signature scheme from bilinear pairings. J. Electro. (china)1 27(3) (2010)
16. Wu, W., Mu, Y., Susilo, W., Seberry, J., Huang, X.: Identity-based proxy signature from pairings. In: Xiao, B., Yang, L.T., Ma, J., Muller-Schloer, C., Hua, Y. (eds.) ATC 2007. LNCS, vol. 4610, pp. 22–31. Springer, Heidelberg (2007)
17. Zhang, J., Zou, W.: Another ID-based proxy signature scheme and its extension. Wuhan Univ. J. Nat. Sci. 12, 133–136 (2007)