ISSN 1392 – 124X, ISSN 2335 – 884X (online) INFORMATION TECHNOLOGY AND CONTROL, 2013, Vol. 42, No. 4

Efficient Proxy Signature Scheme for Mobile Devices Using Bilinear Pairings

Jia-Lun Tsai[1], Nai-Wei Lo[1], Tzong-Chen Wu[1,2]

[1] Department of Information Management,
National Taiwan University of Science and Technology, Taipei 106, Taiwan.

[2] Taiwan Information Security Center (TWISC), National Taiwan University of Science and Technology, Taipei 106, Taiwan
e-mail: [1] crousekimo@yahoo.com.tw

crossref http://dx.doi.org/10.5755/j01.itc.42.4.1969

Abstract. A proxy signature scheme is useful and convenient because it allows a proxy signer to sign a message on behalf of an entity. This study proposes a novel efficient proxy signature scheme for mobile devices using bilinear pairings. The computational cost of the proposed signature scheme is extremely low, and the length of the proposed signature is limited. In addition, our scheme does not require a special hash function, namely the Map-To-Point hash function. We also show that the proposed scheme is secure against adaptive chosen message attacks under a random oracle.

Keywords: proxy signature; bilinear pairings; adaptive chosen-message attacks; random oracle.

1. Introduction

With the rapid development of electronic and wireless technology, mobile devices have gradually emerged and are widely used in everyday life. Mobile devices are extremely convenient tools that enable people to access sensitive information anywhere and at any time using unsecured wireless networks. However, the convenience of these devices has resulted in a number of problems. The main problem that should be addressed is the low computing capacity of typical mobile devices, which is insufficient to execute heavy cryptosystems. Thus, an efficient and strong cryptosystem suitable for low-computing mobile devices is urgently required.

Proxy signature schemes are useful and convenient because they allow an entity to delegate his/her signing power to a designated proxy signer. This proxy signer can then sign messages on behalf of the original signer. According to the delegation type, proxy signature schemes can be divided into the following three categories: full delegation, partial delegation, and delegation by warrant schemes. Full delegation means that the entity provides his/her private key to the proxy signer. Partial delegation means that the entity can generate and send a secret proxy key to the proxy signer; however, this key does not compromise the private key of the entity. Delegation by warrant means that the entity can allocate a warrant, which specifies the delegation information to the proxy signer. After delegation, the proxy signer can sign messages according to the warrant. Evidently, delegation by warrant is superior to the other delegation types. The concept of a proxy signature was proposed by Mambo et al. [1] in 1996. Subsequently, numerous proxy signature schemes have been proposed [2-10].

Bilinear pairings, namely the Weil pairing and the Tate pairing of algebraic curves, are used to evaluate discrete logarithm problems during the initial stage [11]. Numerous studies have determined that bilinear pairings can be used to construct new cryptosystems. Consequently, many cryptosystems have been proposed in recent years. The first signature scheme to employ bilinear pairings was proposed by Boneh et al. (known as the BLS scheme) [12]. The primary benefit of this scheme is that the length of the signature is approximately half that of a standard signature while providing a similar level of security. However, this scheme requires a special one-way hash function called a Map-To-Point hash function. This special one-way hash function is probabilistic and generally inefficient. Zhang et al. [13] proposed an efficient short signature scheme without the Map-To-Point function. This scheme has fewer pairing operations compared to the BLS scheme. They also proposed a ring signature scheme and a proxy signature scheme based on their proposed short signature scheme.

Subsequently, Lu et al. [14, 15] and Wang et al. [16] proposed a proxy signature scheme using pairings, respectively [14-16]. These two schemes are efficient, but both of them require the Map-To-Point hash function.

This study proposes an efficient short-proxy signature scheme for mobile devices. The computational costs of the proposed signature scheme are extremely low, and the signature length is limited. Thus, the proposed scheme is suitable for the mobile environment. We also verify the security of the proposed scheme against adaptive chosen message attacks under a random oracle.

In section 2, we review the bilinear pairings. Section 3 introduces our proposed scheme. Security analysis and a comparison are given in section 4 and section 5..

2. Bilinear pairings

Let G_1 and G_2 be a cyclic additive group and a cyclic multiplicative group, and let e: $G_1 \times G_1 \rightarrow G_2$ be a pairings function satisfying the following properties, where G_1 and G_2 have the same prime order q.

1. **Bilinearity:** For all P_1, P_2, Q_1, $Q_2 \in G_1$ and a, $b \in Z_q^*$, $e(aP_1, bQ_1) = e(P_1, Q_1)^{ab} = e(abP_1, Q_1) = e(P_1, abQ_1)$, $e(P_1 + P_2, Q_1) = e(P_1, Q_1) \bullet e(P_2, Q_1)$, $e(P_1, Q_1 + Q_2) = e(P_1, Q_1) \bullet e(P_1, Q_2)$.
2. **Non-degenerate:** There exists $P_1 \in G_1$, such that $e(P_1, P_1) \neq 1$.
3. **Computable:** For all P_1, $Q_1 \in G_1$, it is easy to compute $e(P, Q)$.

3. Our proposed scheme

This section introduces the proposed proxy signature scheme. The proposed scheme consists of six algorithms: system setup, key extract, delegation generation, delegation verification, proxy signature generation and proxy signature verification algorithms. Before presenting our scheme, we first describe the notations in Table 1.

Table 1. Notations

Notation	Descriptions
x_O, P_O	private key and public key of the original signer
x_p, P_p	private key and public key of the proxy signer
m	Message
m_w	Warrant
s_O, s_p	delegation and the proxy signature
e	bilinear pairings function
H_1, H_2	one-way hash functions

Details of each algorithm are described as follows.

System setup: Let G_1 and G_2 be the additive and multiplicative group of the same prime order q, where P is the generator of G_1. At the beginning, the system computes $e(P, P)$ and then chooses a bilinear pairings function $e: G_1 \times G_1 \rightarrow G_2$ and two hash functions $H_1: Z \rightarrow Z_q$ and $H_2: Z \times Z \times G_1 \rightarrow Z_q$. Next, the system parameters $\{G_1, G_2, e, q, e(P, P), P, H_1, H_2\}$ are published and can be gained by anyone.

Key extract: The original signer computes its corresponding public key $P_O = x_O P$, where $1 \leq x_O \leq q-1$ is his/her chosen private key. The proxy signer also generates his/her private/public key pair (x_p, $P_p = x_p P$), where $1 \leq x_p \leq q-1$. Both of them publish their public keys.

Delegation generation: When an original signer wants to delegate his/her signing power to the proxy signer, the original signer first makes a warrant m_w that is an explicit description of the delegation relation. The original signer checks whether $(H_1(m_w)+x_O)$ is the same as the order q. If they are identical, the original signer appends a redundancy bit to this warrant m_w. After that, the original signer computes

$$s_O = \frac{1}{H_1(m_w) + x_O} P \,. \tag{1}$$

Next, the original signer sends (m_w, s_O) to the proxy signer via a secure channel.

Delegation verification: Upon receiving (m_w, s_O), the proxy signer can verify its validity by checking the following equation:

$$e(s_O, H_1(m_w)P + P_O) ?= e(P, P) \,. \tag{2}$$

If it holds, s_O is accepted, so the proxy signer computes $R = x_p P_O$; otherwise, the proxy signer rejects s_O. After that, the proxy signer publishes R as his/her public parameter.

Proxy signature generation: When the proxy signer wants to sign the message m on behalf of the original signer, the proxy signer first checks whether $H_2(m, m_w, R)+x_p$ is the same as the prime order q. If they are the same, the proxy signer appends a redundancy bit to the message. After that, the proxy signer computes

$$s_p = \frac{1}{H_2(m, m_w, R) + x_p} s_O \,, \tag{3}$$

and then sends (m, m_w, s_p) to the verifier. Note that s_p is the proxy signature.

Proxy signature verification: After receiving (m, m_w, s_p), the verifier can verify the proxy signature s_p by computing the following equation:

$$e(s_p, R + H_1(m_w)P_p + H_2(m, m_w, R) \cdot \\ \cdot (H_1(m_w)P + P_O)) ?= e(P, P) \,. \tag{4}$$

If it holds, the verifier accepts the proxy signature s_p; otherwise, the proxy signature s_p is rejected by the verifier.

4. Security analyses

This section gives the security proof of the proposed proxy signature scheme under the random oracle. The proposed scheme is secure against the adaptive chosen message attack with the assumption of the Collusion Attack Algorithm with *k* traitors problem (k-CAA problem) [17, 18]. In an adaptive chosen message attack, an adversary can obtain many valid signatures for arbitrary messages of his/her choice. He/she then forges a valid signature using these signatures. In the following, we introduce the k-CAA problem.

Assumption 1. *The k-CAA (Collusion Attack Algorithm with k traitors) problem. Given {P, sP, e_1, e_2, …, e_n, $\frac{1}{s+e_1}P, \frac{1}{s+e_2}P, ..., \frac{1}{s+e_n}P$ }, it is hard to compute $\frac{1}{s+e}P$, where $e \notin \{e_1, e_2, ..., e_n\}$.*

Next, we define the proxy model for the proxy signature scheme. Derived from previous definitions related to the proxy signatures addressed in [19-26], we give the security model for the proxy signature scheme as follows. In this security model, the adversaries can be divided into three types.

Type I: The adversary only has the public key of the original and proxy signers.

Type II: In this type, the adversary has not only the public key of the original and proxy signers, but also the private key of the proxy signer. He/she wants to masquerade as the original signer to generate a delegation on his/her chosen warrant.

Type III: The adversary has not only the public key of the original and proxy signers, but also the private key of the original signer. Besides, he/she also owns the original signer's delegation. He/she wants to masquerade as a proxy signer to generate a signature on his/her chosen message.

It is obvious that if the proxy signature can withstand the type II and type III adversaries, our proposed scheme also can withstand the type I adversary. According to the type II and type III adversary, we define the following games to prove that the proposed proxy signature is secure against the adaptive chosen message attack.

Definition 1. Let A_{II} be the type II adversary. In the following game, A_{II} interacts with the Challenger *C*. We say that the proxy signature scheme can withstand the type II adversary if it has a negligible probability of success under the random oracle model.

Setup. Given the secure parameters, *C* runs this algorithm to obtain the public system parameters and the privacy and public key pairs (x_O, Y_O), (x_P, Y_P) of the original signer and the proxy signer.

***H_1* Hash queries.** In this query, A_{II} can request the hash value on any chosen warrant. Then, *C* responds to A_{II} with the hashed warrant.

Delegation signature queries. A_{II} can request a signature on his/her chosen warrant. After that, *C* returns a signature σ on this queried warrant.

Output. A_{II} outputs a valid w_m and its corresponding delegation σ' which has never been queried.

Definition 2. Let A_{III} be the type III adversary. In the following game, A_{III} interacts with the Challenger *C*. We say that the proxy signature scheme can withstand the type III adversary if it has a negligible probability of success under the random oracle model.

Setup. Given the secure parameters, *C* runs this algorithm to obtain the system public parameters and the privacy and public key pairs (x_O, Y_O), (x_P, Y_P) of the proxy signer and the public key of the original signer. Moreover, the original signer gives the type II adversary a valid delegation.

***H_2* Hash queries.** In this query, A_{III} can request the hash value on any chosen warrant and obtained message. Then, *C* returns it to A_{III}.

Proxy signature queries. A_{III} can request a proxy signature on his/her chosen message. After that, *C* returns a proxy signature σ on this queried message.

Output. A_{III} outputs a valid *m'* and its corresponding proxy signature σ' which has never been queried.

Now we have the following theorems to show that the proposed scheme is secure against the type II adversary and type III adversary.

Theorem 1. *Suppose that the type II adversary A_{II} can request at most q_{H1} H_1 hash queries and q_D delegation queries. If the type II adversary A_{II} -(t, ε) can break our scheme, there also exists an algorithm C-(t', ε ')that can solve the k-CAA problem, where t' = t and $\varepsilon' \geq (\frac{q_D}{q_{H_1}})^{q_D} \cdot \varepsilon$.*

▼**Proof.** We will show that the proposed scheme is secure against type II adversary A_{II} in this theorem. In this theorem, we construct an algorithm *C* that makes use of the type II adversary A_{II} to solve the k-CAA problem. Note that *C* maintains an empty hash list ***H_1*-list** to store the tuple (m_{wi}, $H(m_{wi})$) if the A_{II} asks the H_1 hash query.

***H_1* Hash queries**: H_1 Hash queries can generate and return the hashed warrant $H_1(m_{wi})$ to A_{II} if the A_{II} requests. For each request (m_{wi}), C first checks whether m_{wi} exists in the ***H_1*-list**. If it holds, *C* returns $H(m_{wi})$ to A_{II}. Otherwise, *C* randomly chooses $h_i \in Z_q^*$, and then adds a tuple (m_w, h_i) to ***H_1*-list**, set h_i = $H_1(m_{wi})$. Next, h_i is returned to the adversary A_{II}.

Delegation queries: In this query, A_{II} requests C to generate the delegation on his/her chosen warrant m_{wi} and then returns it to A_{II}. If m_{wi} exists in ***H*₁-list**, C returns $s_O = \frac{1}{H_1(m_w) + x_O} P$. Otherwise, C aborts the simulation and returns a "failure" message.

Output: After all the above queries, A_{II} outputs the delegation s_O on a warrant m_{wi}, which satisfies the following verification equation:

$$e(s_O, H_1(m_w)P + P_O) ? = e(P, P) . \quad (5)$$

Now, we analyze the advantage of C in this game. In the H_1 hash query, the success probability of A_{II} is $\frac{q_D}{q_{H_1}}$, so for all delegation oracle queries, the success probability of A_{II} is $\varepsilon' \geq (\frac{q_D}{q_{H_1}})^{q_D} \cdot \varepsilon$. ▲

Theorem 2. *Suppose that the type III adversary A_{III} can request at most q_{H2} H_2 hash queries and q_{PS} proxy signature queries. If the type III adversary A_{III} -(t, ε) can break the proposed scheme, there also exists an algorithm C-(t', ε') that can solve the k-CAA problem, where t' = t and* $\varepsilon' \geq (\frac{q_{PS}}{q_{H_2}})^{q_{PS}} \cdot \varepsilon$.

▼**Proof.** In the following, we prove that the proposed scheme is secure against type III adversary A_{III} if the k-CAA problem cannot be solved. Like theorem 1, we construct an algorithm C that makes use of the type III adversary A_{III} to solve the k-CAA problem. Note that the adversary C also maintains an empty hash-list ***H*₂-list** to store the tuple (m_i, $H_2(m_i, m_w, R)$) if the A_{III} asks the H_2 hash query.

***H*₂ Hash queries**: This query is used to generate and return the hashed triplet (m_i, m_w, R) to A_{III} if A_{III} makes H_2 hash queries on the message m_i. When A_{III} makes this query on the message m_i, C first checks whether m_i exists in the ***H*₂-list**. If it exists there, C sends $H_2(m_i, m_w, R)$ back A_{III}. Otherwise, C chooses a random $h_i \in Z_q^*$ and then adds a tuple (m_i, h_i) to the ***H*₂-list**, and sets $h_i = H_2(m_i, m_w, R)$. After that, C returns h_i to A_{III}.

Proxy signature queries: A_{III} makes a request C to generate the proxy signature on his/her chosen message m_i. If m_i exists in the ***H*₂-list**, C returns $s_p = \frac{1}{H_2(m, m_w, R) + x_p} s_O$. Otherwise, C aborts the simulation and returns a "failure" message.

Output: After all the above queries, A_{III} outputs the proxy signature on his/her chosen message m, which satisfies the following verification equation:

$$e(s_p, R + H_1(m_w)P_p + H_2(m, m_w, R) \cdot (H_1(m_w)P + P_O)) ? = e(P, P) \cdot$$

Now, we analyze the advantage of C in this game. In the H_2 hash query, the success probability of A_{III} is $\frac{q_{PS}}{q_{H_2}}$, so for all proxy signature queries, the success probability of A_{III} is $\varepsilon' \geq (\frac{q_{PS}}{q_{H_2}})^{q_{PS}} \cdot \varepsilon$. ▲

5. Comparison

In this section, we compare the proposed scheme with other related schemes regarding the delegation length, proxy signature length, pre-computation proxy signature length, computational costs of delegation, delegation verification, proxy signature generation, and proxy signature verification. For comparison, we included the schemes developed in the following studies: Zhang et al. [13], Lu et al. [14], and Wang et al. [16]. Let $|l|$ be the bit length of an element in G_1, and let n be the number of messages signed by the proxy signer. We first define the notations that are used to analyze efficiency.

T_m:	the time for performing a modular multiplication operation
T_b:	the time for performing a bilinear pairing operation
T_s:	the time for performing a point multiplication operation
T_h:	the time for performing a hash function
T_H:	the time for performing a Map-To-Point hash function

The inverse and the point addition operations [27–30] are much lower than the Map-To-Point hash function, the point multiplication operation and the pairings operation, so this study does not consider these two operations in our efficiency analysis. Table 2 shows the computational costs between point multiplication and bilinear pairing operations on a Philips HiPerSmart smartcard with a maximum clock speed of 36 MHz and a Pentium IV desktop computer with a maximum clock speed of 3GHz, which were reported in [30]. Based on Table 2, one can observe that the computational effort required to execute one pairing operation is approximately equal to three times of computational effort to execute one point multiplication operation.

Table 2. Computational costs of point multiplication and bilinear pairing operations on a 36 MHz Philips HiPerSmart smartcard and a 3GHz Pentium IV computer [30]

Device	Point Multiplication Operation	Bilinear Pairing Operation
36 MHz Smartcard	0.13 s	0.38 s
3GHz Computer	1.17 ms	3.16 ms

The computational complexity of the delegation is first discussed. In the proposed scheme, the original signer requires only T_s+T_h to generate the delegation. The delegation verification cost for the proxy signer is $T_s+T_b+T_h$. Thereafter, the proxy signer requires T_s to generate and publish $R = x_pP_{pub}$. However, $R = x_pP_{pub}$ only requires one computation and can be pre-computed. Next, consider the complexity of the proxy signature. In the proposed scheme, the proxy signer requires nT_s+nT_h to generate all proxy signatures on his/her chosen messages. After receiving the proxy signature, the verifier requires $T_b+3T_s+2T_h$ to verify the validity of each proxy signature. Comparisons are shown in Table 3. In terms of signature length usage in one session, one can discover that our signature scheme only requires $2|l|$, which is the shortest length among existing schemes as shown in item E4 of Table 3. In terms of performance efficiency, the comparison results in items E5-E9 of Table 3 indicate that the proposed proxy signature scheme is one of the most efficient solutions. In addition, the proposed scheme is secure and does not require the Map-To-Point hash function as shown in items E10-E11 of Table 3.

Table 3. The comparison among proxy signature schemes

	Zhang *et al.* [13]	**Lu *et al.* [14]**	**Wang *et al.* [16]**	**Our scheme**
E1	$\|l\|$	$2\|l\|$	$\|l\|$	$\|l\|$
E2	$2\|l\|$	$2\|l\|$	$2\|l\|$	$2\|l\|$
E3	$2\|l\|$	$\|l\|$	$2\|l\|$	$\|l\|$
E4	$3\|l\|$	$3\|l\|$	$3\|l\|$	$2\|l\|$
E5	T_s+T_h	$T_s+T_m+T_h$	T_s+T_H	T_s+T_h
E6	$2T_b+T_s+T_h$	$2T_s+T_h$	$2T_b+T_H$	$T_b+T_s+T_h$
E7	T_s	T_m	T_s	T_s
E8	$3nT_s+2nT_h$	nT_m+nT_H	$2nT_s+nT_h$	nT_s+nT_h
E9	$2T_b+2T_s+2T_h$	$2T_b+T_s+T_H+T_h$	$3T_b+T_s+T_H+T_h$	$T_b+3T_s+2T_h$
E10	Secure	Unsecure	Secure	Secure
E11	No	Yes	Yes	No

E1: the delegation length. E2: the proxy signature length. E3: the proxy signature length under pre-computation. E4: the total length required for one session with pre-computation technique. E5: the computation cost of the delegation generation. E6: the computation cost of the delegation verification. E7: the computation cost of the proxy key generation. E8: the computation cost of the proxy signature generation. E9: the computation cost of the proxy signature verification for each signature. E10: Security. E11: Map-To-Point hash function.

6. Conclusions

This study proposed a new efficient proxy signature scheme without the Map-To-Point hash function. The security of our scheme is based on bilinear pairings. We also verified the security of the proposed proxy signature scheme before comparing the proposed scheme with other related works. The comparison results showed that the proposed scheme was the most efficient solution; thus, the proposed scheme is more suitable for mobile devices with less computing power.

Acknowledgements

The authors gratefully acknowledge the support from Taiwan Information Security Center (TWISC) and National Science Council, Taiwan, under the Grants Numbers NSC 102-2218-E-011-012 and NSC 102-2218-E-011-013.

References

[1] **M. Mambo, K. Usuda, E. Okamoto.** Proxy signatures: delegation of the power to sign messages. *IEICE Trans. on Fundamentals of Electronics, Communications and Computer Sciences*, 1996, Vol. E79, No. A(9), 1338–1354.

[2] **M. Mambo, K. Usuda, E. Okamoto.** Proxy signatures for delegating signing operation. In: *Proceedings of Third ACM Conference on Computer and Communication Security*, January 1996, pp. 48–57.

[3] **H. M. Sun.** An efficient nonrepudiable threshold proxy signature scheme with known signers. *Computer Communications*, 1999, Vol. 22, No. 8, 717–722.

[4] **H. M. Sun, N. Y. Lee, T. Hwang.** Threshold proxy signatures. In: *IEE Proceedings Computers and Digital Techniques*, 1999, Vol. 146, No. 5, pp. 259–263.

[5] **L. Yi, G. Bai, G. Xiao.** Proxy multi-signature scheme: A new type of proxy signature scheme. *Electronic Letters*, 2000, Vol. 36, No. 6, 527–528.

[6] **S. J. Hwang, C. C. Chen.** A new proxy multi-signature scheme. In: *International Workshop on*

Cryptology and Network Security, Taipei, December 2000, pp. 134–138.

[7] **C. L. Hsu, T. S. Wu, T. C. Wu.** New nonrepudiable threshold proxy signature scheme with known signers. *Journal of Systems and Software*, 2001, Vol. 58, No. 2, 119–124.

[8] **Z. Shao.** Proxy signature schemes based on factoring, *Information Processing Letters,* 2003, Vol. 85, No. 3, 137–143.

[9] **R. Lu, Z. Cao, Y. Zhou.** Proxy blind multi-signature scheme without a secure channel. *Applied Mathematics and Computation*, 2005, Vol. 164, No. 1, 179–187.

[10] **R. Lu, Z. Cao, H. Zhu.** A robust (k, n) + 1 threshold proxy signature scheme based on factoring. *Appl Math Comput*, 2005, Vol. 166, No. 1, 35–45.

[11] **M. Alfred, V. Scott, O. Tatsuaki.** Reducing elliptic curve logarithms to logarithms in a finite field. In: *STOC '91: Proceedings of the twenty-third annual ACM symposium on Theory of Computing (New York, NY, USA)*, ACM Press, 1991, pp. 80–89.

[12] **D. Boneh, B. Lynn, H. Shacham.** Short signature from the Weil pairing. In: *Asia Crypto 2001*, 2001, pp. 514–532.

[13] **F. Zhang, S. N. Reihaneh, W. Susilo.** An efficient signature scheme from bilinear pairings and its applications. In: *Public Key Cryptography – PKC 2004*, 2004, pp. 277–290.

[14] **R. X. Lu, X. L. Dong, Z. F. Cao.** Designing efficient proxy signature schemes for mobile communication. *Science in China Series F: Information Sciences,* 2008, Vol. 51, No. 2, 183–195.

[15] **F. G. Li, M. Shirase, T. Takagi.** Cryptanalysis of efficient proxy signature schemes for mobile communication. *Science in China Series F: Information Sciences*, 2010, Vol. 53, No. 10, 2016–2021.

[16] **A. Wang, J. Li, Z. Wang.** A provably secure proxy signature scheme from bilinear pairings. *Journal of Electronics (China),* 2010, Vol. 27, No. 3, 298–304.

[17] **H. Du, Q. Wen.** An efficient identity-based short signature scheme from bilinear pairings. In: *2007 International Conference on Computational Intelligence and Security*, 2007, pp. 725–729.

[18] **S. Mitsunari, R. Sakai, M. Kasahara.** A new traitor tracing. *IEICE Trans. on Fundamentals of Electronics. Communications and Computer Sciences*, 2002, Vol. E85, No. A(2), 481–484.

[19] **F. Cao, Z. F. Cao.** Cryptanalysis on a Proxy multi-signature scheme. In: *Proceedings of the IMSCCS'06*, 2006, pp. 117–120.

[20] **Y. F. Chang, W. L. Tai, C. Y. Lin.** A verifiable proxy signature scheme based on bilinear pairings with identity-based cryptographic approaches. *Information Technology and Control*, 2012, Vol. 41, No. 1, 60–68.

[21] **B. Lee, H. Kim, K. Kim.** Strong proxy signature and its applications. *Symposium on Cryptography and Information Security (SCIS02001),* 2001, 603–608.

[22] **Y. C. Lin, T. C. Wu, J. L. Tsai.** ID-based aggregate proxy signature scheme realizing warrant-based delegation. *Journal of Information Science and Engineering*, 2013, Vol. 29 No. 3, 441–457.

[23] **X. Huang, Y. Mu, W. Susilo, W. Wu.** Proxy signature without random oracles. In: *Proceedings of MSN 2006*, 2006, pp. 473–484.

[24] **J. C. N. Schuldt, K. Matsuura, K. G. Paterson.** Proxy signatures secure against proxy key exposure. In: *Proceedings of the PKC 2008*, 2008, pp. 344–359.

[25] **Y. Sun, C. Xu, Y. Yu, B. Yang.** Improvement of a proxy multi-signature scheme without random oracles. *Computer Communications*, 2011, Vol. 34, No. 3, 257–263.

[26] **Q. Wang, Z. Cao.** Formal model of proxy multi-signature and a construction. C*hinese Journal of Computers*, 2006, Vol. 29, No. 9, 1628–1635.

[27] **P. S. L. M. Barreto, H. Y. Kim, B. Lynn, M. Scott.** Efficient algorithms for pairing-based cryptosystems. In: *Advances in Cryptology-Crypto 2002*, 2002, pp. 354–369.

[28] **P. S. L. M. Barreto, B. Lynn, M. Scott.** On the selection of pairing-friendly groups. In: *SAC 2003*, 2003, pp. 17-25.

[29] **S. D. Galbraith, K. Harrison, D. Soldera.** Implementing the Tate pairing. In: *ANTS 2002*, 2002, pp. 324–337.

[30] **M. Scott, N. Costigan, W. Abdulwahab.** Implementing cryptographic pairings on smartcards. In: *Proceedings of Cryptographic Hardware and Embedded Systems*, 2006, pp. 134–147.

Received June 2012.