Design and Application of Batch Verification Scheme on Internet E-Voting Systems

Wen-Shen Lai [1] Chu-Hsing Lin [2,*] Jen-Chieh Chang [3] Ruei-Hau Hsu [4] Mei-Chun Chou [5]

[1] Department of Information Management
Chienkuo Technology University
Changhua, Taiwan
WenShenLai@gmail.com

[2,3,4,5] Department of Computer Science and Information Engineering
Tunghai University
Taichung City, Taiwan
{[2]chlin, [3]g942817, [5]g96350011}@thu.edu.tw, [4]xyz.hsu@msa.hinet.net

Received 1 September 2007; Revised 15 November2007; Accepted 3 January 2008

Abstract. The digital signature has been applied extensively for message authentication and verification in e-cash, e-bank, e-voting, and e-document. The signing processes of the digital signature scheme and the traditional signature scheme are the same. But the verification processes of them are different. The traditional signature scheme verifies signatures one by one, whereas digital signatures can be verified altogether in a batch signature verification scheme. In general, signing a message does not spend too much time, which is contrary to the time of the verifying process addressed in this paper. Many batch signature papers have been proposed in the past. In this paper, we will adopt the architecture of Lin's method to implement an e-voting system.

Keywords: e-voting, batch verification, digital signature, authentication and verification.

1 Introduction

The digital signature has a wide range of applications in authentication and verification of e-cash, e-bank, e-document, and e-voting. The signing processes of the traditional and digital signature schemes are similar and the time cost of them is alike, too. But the verification processes of them can be quite different. One way of digital signature verification is to process the signature in a batch to accelerate the verification process, which normally cost much more time than the signing process.

Many signature verification schemes have been published in the literature. In this paper, we will investigate experimentally the effectiveness of a traditional signature verification scheme and a modified batch signature verification scheme. The costs of time measured from implementation of both signature verification schemes corroborate the theoretical conclusion that the batch signature verification process is faster when signatures are verified in suitable sized batch blocks.

The rest of this paper is organized as follows. In Section 2, we introduce the related background, and in Section 3, we implemented an e-voting system with Lin's method. In Section 4, we analyze experimentally efficiencies of traditional and batch signature verifications. Section 5 concludes this paper.

2 Relative background

In this paper, we will focus on comparisons related to traditional and batch signature verifications. In the following subsections, we will introduce the method of traditional signature verification, DSA [1] and the method of batch signature verification based on the DSA – Naccache, which is proposed by Naccache et al [2].

* Correspondence author

2.1 Digital Signature Algorithm, DSA

DSA is a variant integrated Schnorr [3,4] and ElGamal [5,6] signature algorithm. Parameters in DSA are defined as follows.

p: a large prime with bit length between 512 to1024 of the multiple of 64.
q: a large prime divisor of p – 1, and the bit length equal to 160.
g: an element in Z_p of order q.
x: a secret key belongs to Z_q.
y: a public key $y=g^x \bmod p$.
H(•): a secure hash algorithm,.
m: a message.

DSA's signing and verifying processes are as follows.

Signing process

Step 1: The signer chooses a random number k belongs to Z_q.
Step 2: The signer creates signature according to the following formulas:

$$r = (g^k \bmod p) \bmod q \ . \quad (1)$$

$$s = (k^{-1}(H(m) + xr)) \bmod q \ . \quad (2)$$

The signature pair (r, s) of message m will be sent to the verifier.

Verifying process

To verify the received signature pair (r, s), the verifier computes the following formulas:

$$w = s^{-1} \bmod q \ . \quad (3)$$

$$u_1 = (H(m) * w) \bmod q \ . \quad (4)$$

$$u_2 = (rw) \bmod q \ . \quad (5)$$

$$v = ((g^{u_1} y^{u_2}) \bmod p) \bmod q \ . \quad (6)$$

If the equality v = r establishes, then the signature is correct.

2.2 Naccache's batch signature verification method

Naccache et al. proposed a batch signature verification method based on the DSA algorithm. Parameters of this method are defined as follows.

p: a large prime.
q: a prime divisor of p – 1
g: an element of a order q in GF(p).
x: a secret key of signer.
y: a public key of signer, where $y = g^x \bmod p$.
k_i: a group of random value less than q, i = 1, 2, …, t.

Its signing and verifying processes are as follows.

Signing process

A signer is ready to sign t messages: $m_1, m_2, \ldots, m_t$ after respectively creating t signature pairs: $(r_1, s_1), (r_2, s_2), \ldots, (r_t, s_t)$**.** For each message m_i, the signature pair (r_i, s_i) is created according to the following formulas:

$$r_i = g^{k_i} \bmod p. \quad (7)$$

$$s_i = k_i^{-1}(m_i + xr_i) \bmod q. \quad (8)$$

Verifying process

After receiving all signature pairs from the signer, the verifier uses the following formulas to verify them in a batch.

$$\prod_{i=1}^{t} r_i \overset{?}{\equiv} g^{\sum_{i=1}^{t} m_i s_i^{-1} \bmod q} y^{\sum_{i=1}^{t} r_i s_i^{-1} \bmod q} (\bmod p). \quad (9)$$

3 System implementation

In this section, we will introduce the batch signature verification method proposed by Lin et al. [7,8]. The main idea of Lin's method is to design a framework that one can apply any batch signature verification method on it according to one's specific need. If necessary, it can replace the existing batch signature verification method by other methods. In the following subsection, we will present the Lin's method and implementation of the e-voting system.

3.1 Testing method of Batch signature entities

Suppose the batch signature x is composed of n batch signature entities (x_1,y_1), (x_2,y_2), ..., (x_n,y_n), where $n = 2^{k-1}$ and k is a positive integer. Then, we define H as a 2-dimensional verifying k-by-n array as follows:

$$H = \begin{bmatrix} h_{1,1} & h_{1,2} \cdots & h_{1,n} \\ \vdots & \ddots & \vdots \\ h_{k,1} & \cdots & h_{k,n} \end{bmatrix}, h_{i,j} \in \{0,1\}. \quad \textbf{(10)}$$

Where $h_{i,j}$ is a binary value for all indices i and j. Each column in this verifying array is different and its elements can not be all zero.

In addition, we define GT(x, n) as a normal batch signature verification method that is applied based on any known batch signature verifying method. Let x be a batch signature and n, the greatest number of this batch signature, then GT(x, n) is defined as follows:

$$\prod_{i=1}^{n} y_i \equiv g^{\sum_{i=1}^{n} x_i \bmod q} \bmod p. \quad \textbf{(11)}$$

When the equality is established, GT(x, n) = 1, or GT(x, n) = 0.

Next, we introduce the main idea of Lin's batch signature verification method.

The batch verifying steps are as follows.

Step 1

Create k entities from the batch signature x.

For example,

$$x_i = \{(m_j, s_j) \mid h_{i,j} = 1\}. \quad \textbf{(12)}$$

The entity x_i are from some part of batch signature x. Let x_i be subset of the batch signature x. According to the $h_{i,j}$ value, one decides whether to take the entity x_i from batch signature x or not, where i = 1, 2, …, k and j = 1, 2, …, n. When $h_{i,j} = 1$ is established, the j-th signature is taken from x as one of elements in the entity. Otherwise, the j-th signature is not taken. Thus, we finish building of k batch signature entities by above rule.

Step 2

We use the formula, $GT(x_i, 2^{k-1}) = \sigma_i$, to do the verifying process in each batch signature entities. If the verification is correct, then σ_i equals to zero or one. When $\sigma_i=1$ occurs, it means that the batch signature has some wrong

signature. We stop the verifying steps immediately. When all the batch signature entities passed the verifying process and all σ_i are zero, no wrong batch signature entities are detected. We conclude that there is no errors found in the batch signature verifying process.

Example of Batch Signature Verification:
Suppose there are 7 batch signature entities needed to be verified: (x_1, y_1), (x_2, y_2), …, and (x_7, y_7). We define a verifying array as follows.

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}.$$

Then we do the batch signature verifying process of the three batch signature entities as follows:

$$x_1 = \{(x_4, y_4), (x_5, y_5), (x_6, y_6), (x_7, y_7)\} .$$

$$x_2 = \{(x_2, y_2), (x_3, y_3), (x_6, y_6), (x_7, y_7)\} .$$

$$x_3 = \{(x_1, y_1), (x_3, y_3), (x_5, y_5), (x_7, y_7)\} .$$

In the process of verifying the three batch signature entities: $GT(x_1, 4)$, $GT(x_2, 4)$, $GT(x_3, 4)$, it stops when one of them does not pass the verifying. According to the proposed verifying array, it can be considered as a deterministic batch signature verification method.

3.2 E-voting system

Our e-voting system applies the essential methods mentioned in the last subsection. We send the voting messages of users to the e-voting certification authority system. Then, the e-voting certification authority system starts the batch signature verification process to verify the signed voting messages. Once the e-voting certification authority system receives the signed voting messages and detects any invalid signatures, it decides that the batch signature is faked and the batch signature will be abandoned.

3.3 The e-voting system environment

Fig. 1 shows an e-voting system built on a client-server model. An e-voting CA server can accept messages from multiple connecting clients at the same time, and can process batch signature verifications in parallel.

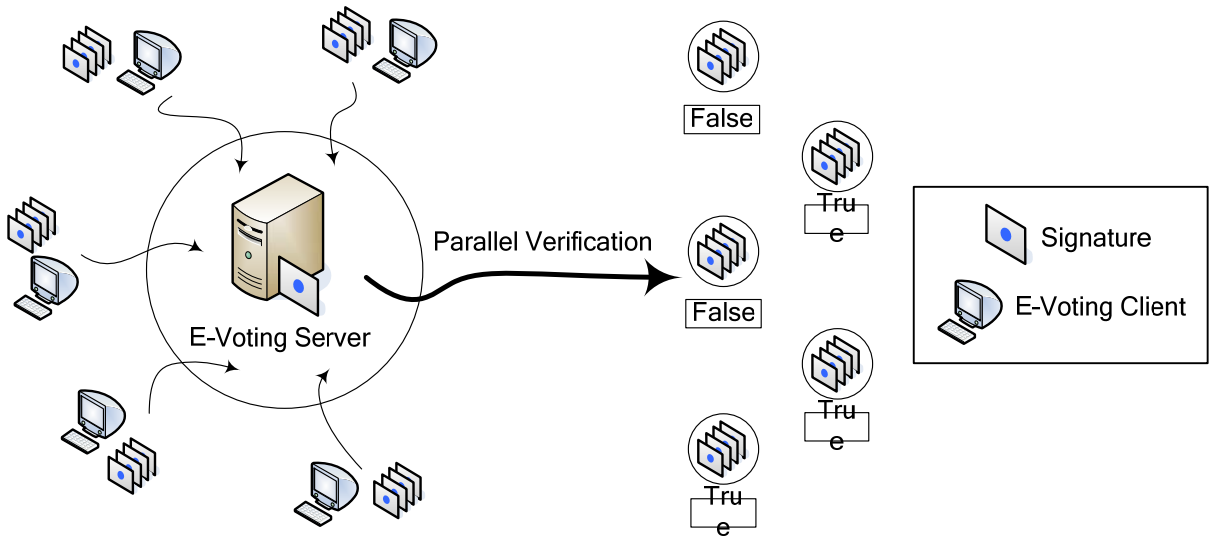

Fig. 1. Components of the e-voting system

2.4 The e-voting system elements

The main framework of the e-voting system is based on a client-server model, including the client and server system. Users vote by using the client system. After the client system collect the related voting messages, it will begin signing process on the messages to create the signature and send it to the server system immediately. In the server system, it always waits to receive any possible batch signature sent from client systems. When the batch signatures received, the server system will verify them and make sure that they are valid or not. Besides, because the server system implemented with a multithread mode, it can receive many batch signatures to verify in parallel. The diagram of the framework of the e-voting system in Fig. 2 shows the relationship of the link, communications and operations.

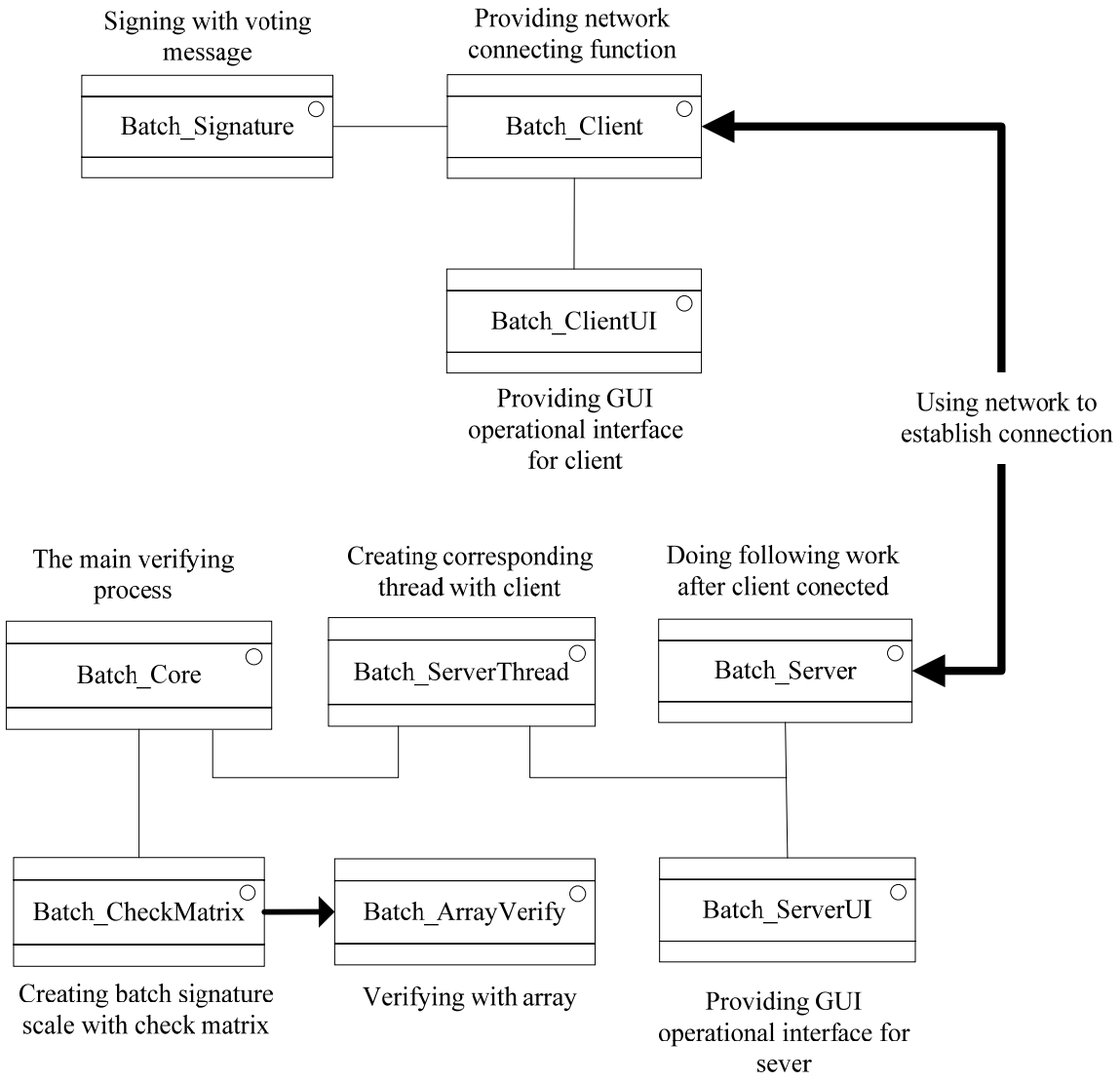

Fig. 2. The framework of the e–voting system

2.5 The batch signature signing and verifying procedures of e-voting system

The processes of the client and the server systems will be described in the following.

In the beginning, the client system starts the voting function and makes sure of network connection with the server, and then users do the voting action. After confirmation of voting action, the client will sign this voting message and collect it in the signature vector form. The signature vector can collect any number of signatures into batch signature and send it to the server system. Fig. 3 shows the main implementation process of the client system.

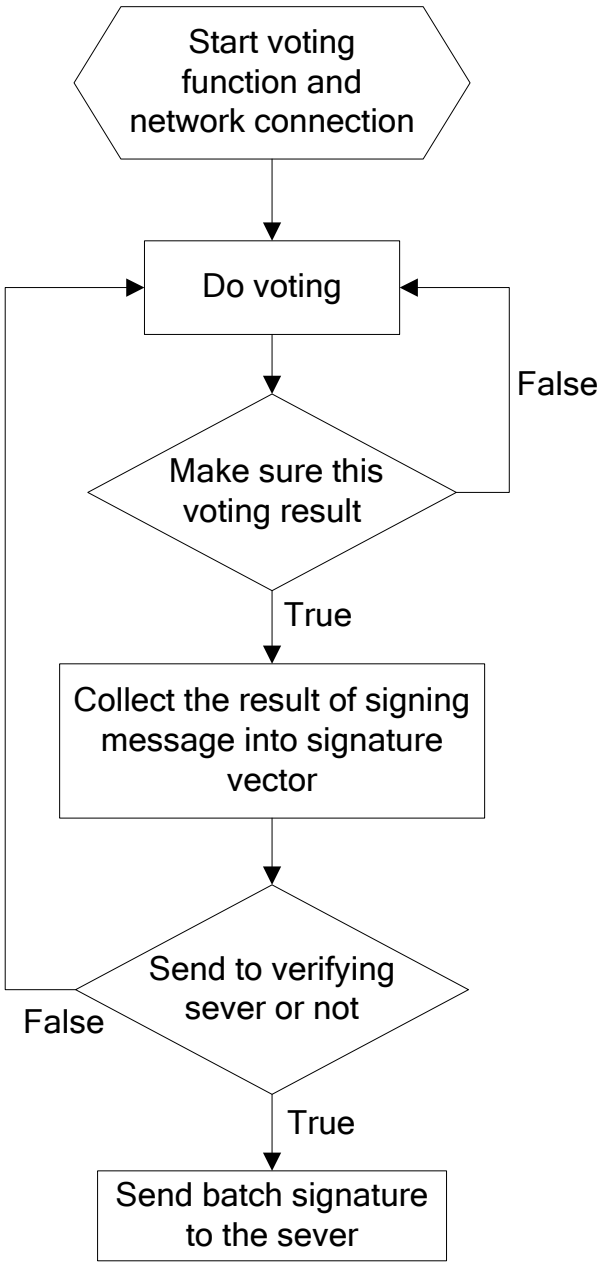

Fig. 3. The client of e-voting system implementation process

Secondly, we will introduce the process of the server. At beginning, the server starts the network connection and waits the client to connect. After a client connects to the server, the server will evoke corresponding thread to deal with the sent batch signature from corresponding client, and create the verifying array. After passing the batch signature and verifying array into the Batch Array Verify element, it will verify this batch signature according to the verifying array. If one entity of this batch signature can not pass the verifying process, the verifying thread will stop immediately and will not accept this batch signature message. The flow chart to implement the server is shown in Fig. 4.

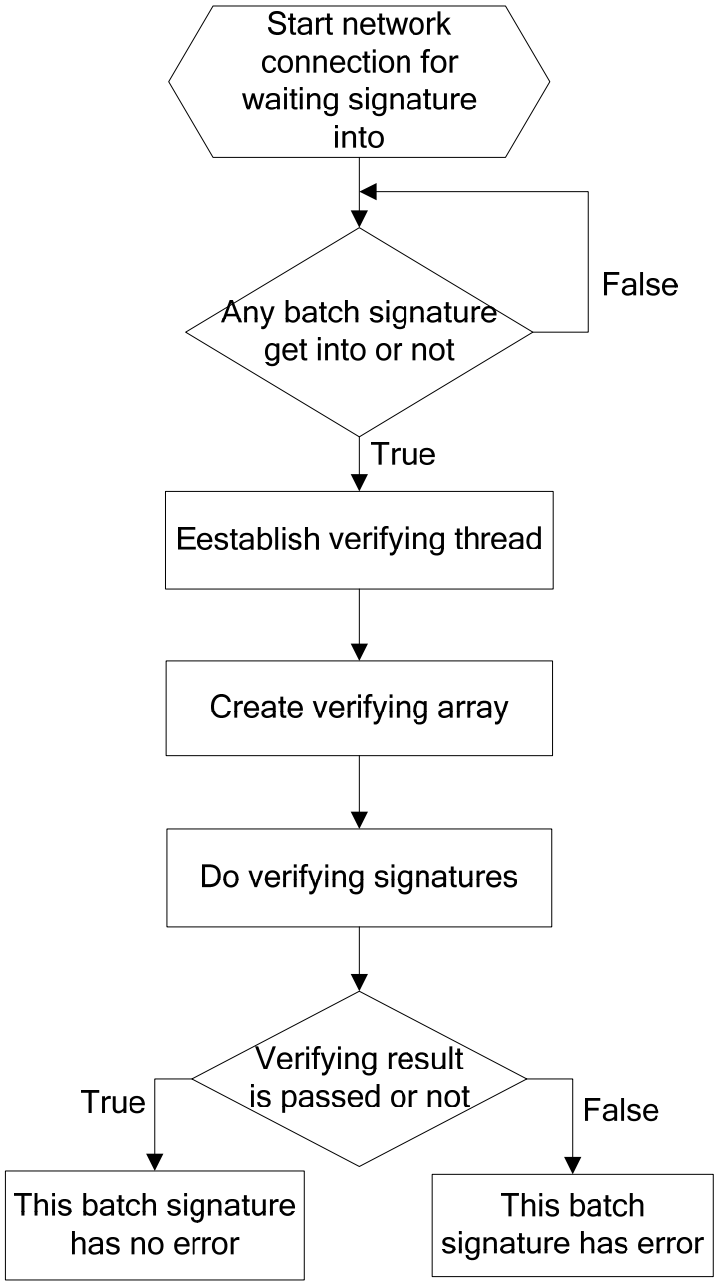

Fig. 4. Implementation of the server of e-voting system

We will show that the e-voting system with user interfaces on the client and server terminals and give a brief description of them in the following:

User interface on the terminal of the client

As shown in Fig. 5, after starting the voting function on the client terminal, the voter makes his voting choice and verifies it. Then the e-voting system verifies the voting message by creating a signature and adds it into signature vector. Then the voting message can be sent to the server by the user. If the server is off-line, the terminal of the client will show a message indicating that it can not connect the server, and hold this batch signature until connected; otherwise, if the server is on-line, the client will send batch signature to the server with a message in the message bar indicating that the sending action is succeeded, and then clear the rest signatures in the signature vector. Subsequently, the program will again wait for the user to restart voting function for the next voting process.

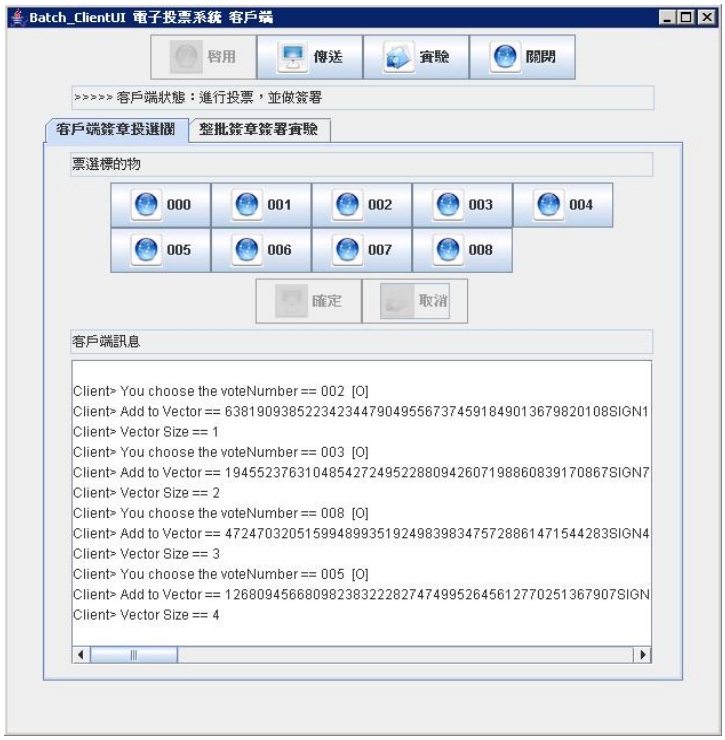

Fig. 5. User interface on the client terminal in e-voting system

User interface on the terminal of the server

The server must start network connection at first, and wait connection from the client. After receiving batch signature from the client, the server pre-processes this batch signature first and then sends it to the corresponding thread of the client. The thread will take over and do the verification of batch signature. Fig. 6 shows seven records of batch signatures sent from clients, and verification results on the server's message bar reported from corresponding threads. Since the results of the seven verification processes are correct, the sent batch signature has no errors and can be accepted.

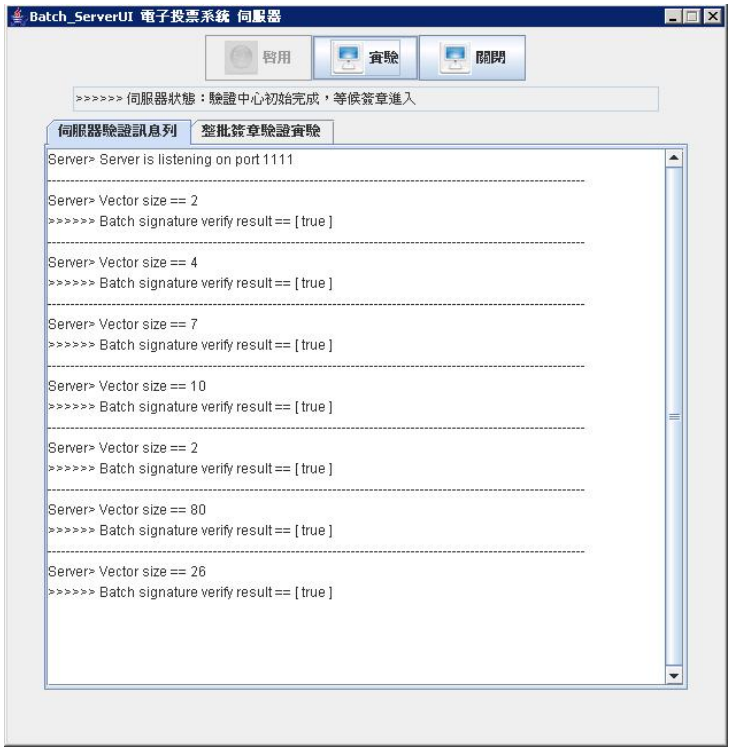

Fig. 6. User operation interface of the server terminal in e-voting system

4 Analysis and comparison of system performance

In this section, we will show results of the experiment to analyze the time cost of e-voting systems. In this experiment, the traditional signature adopts DSA signature verification method and will be called as the traditional signature. Lin's method which applied in the kernel framework with Naccache's batch signature verification method will be called as the batch signature.

Frameworks of traditional signature and proposed batch signature are both based on same theories and similar operations and both use the single signing process. The costs of time of their signing processes are similar as shown in Fig. 7. Therefore, we will not do further analyses on the signing processes.

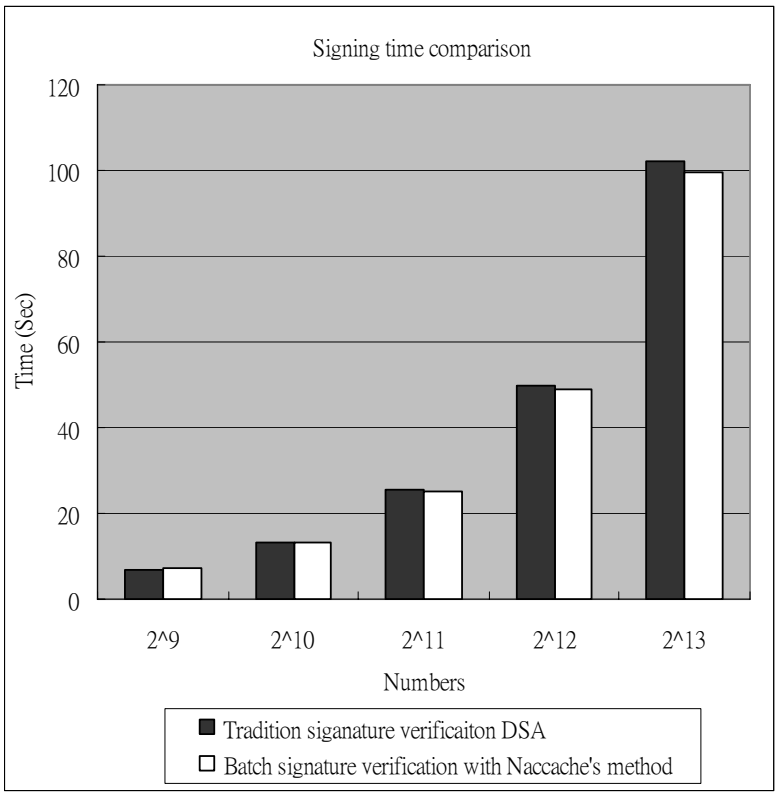

Fig. 7. The signing time of signatures

Because the traditional signature and the batch signature adopt different verification methods, the time cost analysis of verification processes is done. According to the experimental results of verification time, it reveals that when the amount of signature entities is smaller the verification time of the batch signature is less than that of the traditional signature. When the amount of signatures is larger than some specific value, the cost of time to verify the signature by using the batch signature method is higher than that by using the traditional signature verification. Explanation for this conflicting phenomenon comes from the restrictions of system resources. To verify the same amount of signatures, the traditional signature verification uses many single verifying threads, whereas the batch signature verification uses a single batch verifying thread. The time-sharing multitasking property of the system causes the batch signature verification to perform worse than the traditional signature verification. However, theoretically, the batch signature verification would perform better than the traditional signature verification with unlimited system resources. In reality, each of the single verifying process in traditional signature verification requires lesser system resource required by a single batch verifying process. In a time-sharing multitasking system, the batch signature verification might need longer processing time to complete the whole verification process of the signatures. Beside, the processing time grows exponentially with the amount of signatures as shown in Fig. 8.

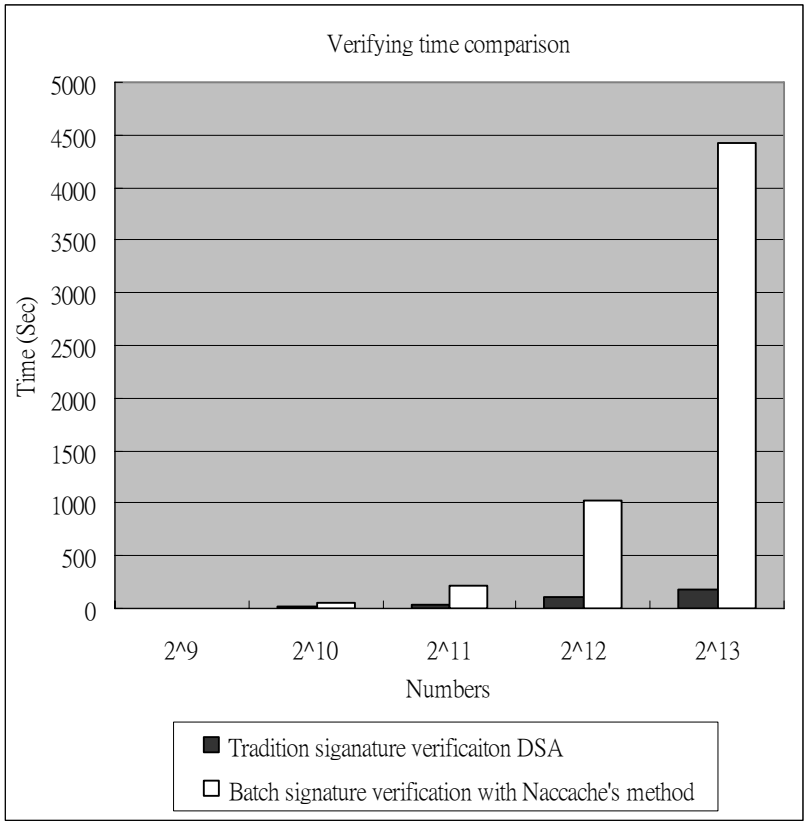

Fig. 8. The verifying time of signatures

Because of restrictions of accessible system resources, batch signature verification process need longer processing time to complete the work. To solve this problem, we first find experimentally that when the number of signatures is 2^7, the batch signature verification time is shorter than the traditional signature verification time. So we divide the batch signatures into blocks of 2^7, and verify the divided blocks of signatures in batches. Therefore, batch signature verification time would grow linearly with the total number of signatures. For instance, suppose that batch signatures of 2^7 require t seconds verifying time, and there has a batch of 2^{10} signatures. With each sub-block of 2^7, we divide the batch signature into 2^3 ($2^{10} / 2^7 = 2^3$) sections. Then, we do the verifying process of 2^3 sections of signature blocks one by one. The verifying time is calculated to be $2^3 \cdot t$ seconds.

It is worth mentioning that the optimal block value depends on the system resources. After our system is implemented, we find experimentally that block of 2^7 is optimal for our system. Fig. 9 shows the results of experiments to decide the optimal block value.

In Fig. 10, it shows the effect of the modified batch signature method by dividing signatures into blocks of 2^7 first. The time cost of modified batch signature method not only is better than that of the traditional signature, but also grows linearly instead of exponentially. Since the verifying time is increased linearly with the amount of signatures, its cost of time is less than that of the tradition signature verification.

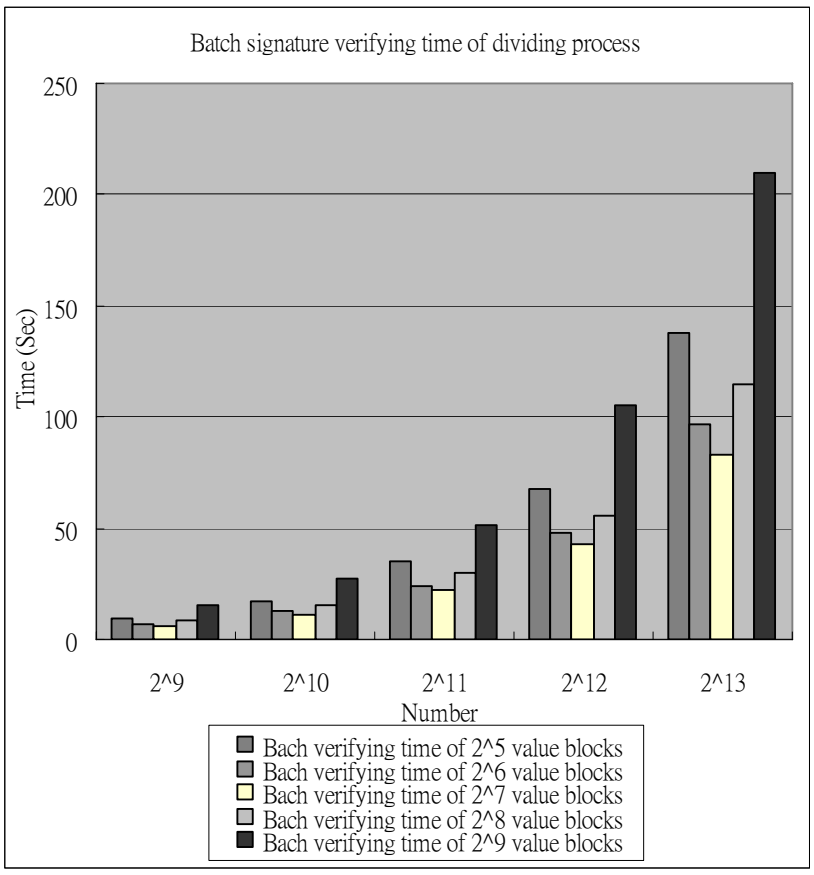

Fig. 9. The verifying time for signatures consists of blocks of different sizes

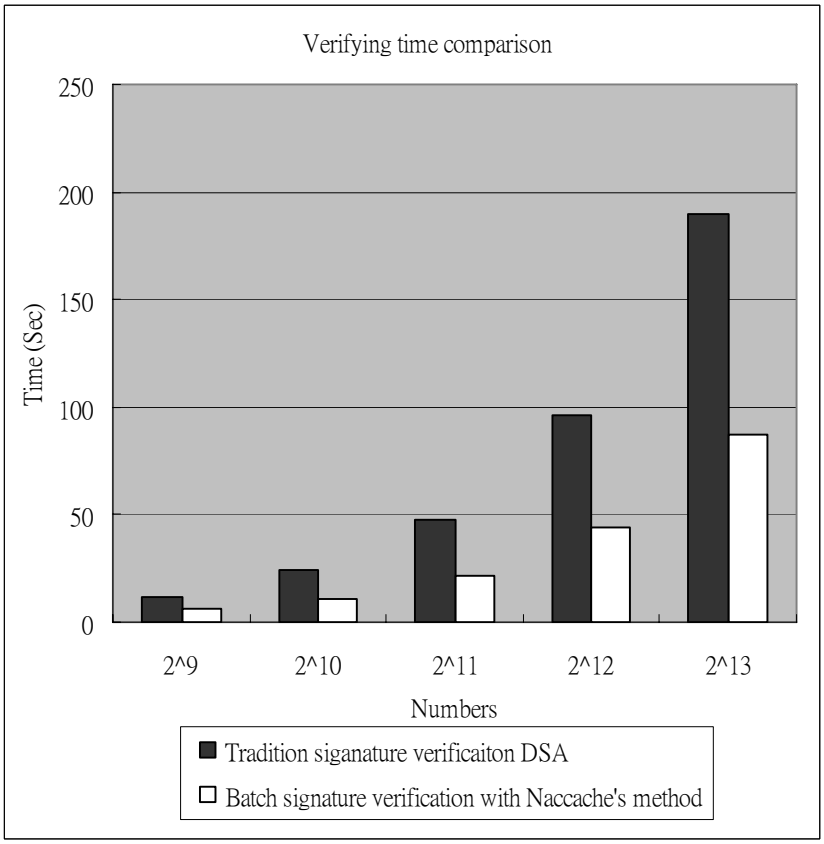

Fig. 10. Verifying time by using the traditional DSA method and the modified Naccache's batch method

5 Conclusions

In views of the feasibility and efficiency of Lin's method of batch signature verification, we choose to implement it in our framework of an e-voting on-line system. In this paper, we not only introduce the Lin's method, but also make a brief description and introduction of the system environment, entity function, and system process in an on-line e-voting system. Then, we demonstrate experimental data, identify the problem to implement the batch signature verification in our system, provide a modified batch signature scheme, and evaluate the effectiveness of the modified system. From the experimental results, we conclude that the modified batch signature verification scheme can be used to implement an effective on-line e-voting system.

Acknowledgement

This system implementation was supported in part by National Science Council under grants, Taiwan Information Security Center NSC94-2213-E-270-009, NSC-95-2218-E-001-001, NSC-95-2218-E-011-015, iCAST NSC96-3114-P -001-002-Y and NSC95-2221-E-029-020-MY3.

References

[1] Proposed Federal Information Processing Standard for Digital Signature Standard (DSS), Federal Register, Vol.56, No.169, pp.42980-42982, 1991.
[2] D. Naccache, D. M'Raihi,, D. Rapheali,, S. Vandenay, "Can DSA be Improved: Complexity Trade-offs with The Digital Signature Standard," *Proceedings of Advances in Cryptology – EUROCRYPT'94*, pp. 77-85, 1995.
[3] C. P. Schnorr, "Efficient Signature Generation for Smart Cards," *Proceedings of Advances in Cryptology – CRYPTO'89,* Springer Verlag, pp. 239-252, 1990.
[4] C. P. Schnorr, "Efficient Signature Generation for Smart Cards," *Journal of Cryptology*, Vol. 4, No. 3, pp. 161-174, 1991.
[5] T. ElGamal, "A Public-key Cryptosystem and a Signature Scheme based on Discrete Logarithms," *Proceedings of Advances in Cryptology – CRYPTO'84*, Springer Verlag, pp.10, 1985.
[6] T. ElGamal, "A Public-key Cryptosystem and a Signature Scheme based on Discrete Logarithms," *IEEE Transactions on Information Theory*, Vol.31, No. 4, pp. 469-472, 1985.
[7] C. H. Lin,, Y. I. Fung,, J. H. Hsu, "Batch Verification with Complete Detection of Bad Signatures," Private Communication.
[8] C. H. Lin, R. H. Hsu , L. Harn, "Improved DSA Variant for Batch Verification," *Applied Mathematics and Computation*, Vol. 169, No.1, pp. 75-81, 2005.