An Efficient Batch Verifying Scheme for Detecting Illegal Signatures

Yanli Ren[1], Shuozhong Wang[1], Xinpeng Zhang[1], Min-Shiang Hwang[2,3]

(Corresponding author: Min-Shiang Hwang)

School of Communication and Information Engineering, Shanghai University, Shanghai 200072, China[1]

Department of Computer Science and Information Engineering, Asia University[2]

No. 500, Lioufeng Rd., Wufeng, Taichung 41354, Taiwan

(Email: mshwang@asia.edu.tw)

Department of Medical Research, China Medical University Hospital, China Medical University[3]

No. 91, Hsueh-Shih Road, Taichung 40402, Taiwan

(Received Dec. 25, 2012; revised and accepted May 14 & July 20, 2014)

Abstract

In a batch verifying scheme, multiple RSA digital signatures can be verified simultaneously in just one exponential operation time. Currently, the verifier could not easily detect where the signature-verification fault was located in most schemes, if the batch verification fails. In this article, we proposed a new batch verifying multiple RSA digital signatures scheme based on a cube. The scheme can detect accurately where the signature-verification fault is located. Moreover, once the total number of signatures is fixed, the size of exponentiation operations is independent from the number of illegal signatures in our scheme. Therefore, our scheme has a better performance and higher efficiency than the previous ones. Finally, we described an extend batch verifying scheme under the condition of n-dimension.

Keywords: Batch verifying, digital signature, multiple signatures

1 Introduction

RSA is a well-known public key cryptosystem where each user has a public key e for encryption (verification) and a private key d for decryption (signature) [17,18]. It protects the transaction information over the network [16], and satisfies the requirement of user authentication and communication security on networking environments [15]. In a RSA signature scheme, the signer uses the personal private key d to sign document M and obtains the signature $S = M^d$, and then the receiver verifies whether $M = S^e$ using the signer's public key e. If there are t documents and signatures $(M_i, S_i)(i = 1, \cdots, t)$, the receiver then needs to verify these signatures one by one and fully executes t exponential computations [11]. This will reduce the computer host's processing ability and the efficiency of RSA signature scheme. Therefore, the concept of batch verifying signatures has been introduced to efficiently improve the performance of verifying multiple RSA signatures [2,6,8,13,19,20].

Harn proposed a batch verifying multiple RSA signature scheme [5] in 1998, where multiple signatures could be verified simultaneously in just one exponential operation time. Such method is considered to be more efficient than the previous signature schemes where the signer must repeatedly verify each signature [3,12]. However, Hwang et al. showed that the Harn's scheme could not resist two kinds of attacks [7,10]. In addition, the verifier could not detect where the signature-verification fault was located if the batch verification fails in Harn's scheme. Since the verifier must re-verify each of the signatures and then confirms where the signature-verification fault is located, it is inefficient to detect the illegal signatures. There are many batch verifying multiple RSA signature schemes have been proposed [1,4,12,21].

Recently, Li et al. proposed a matrix-based solution to quickly find out where the signature-verification faults are located without re-verifying each of the signatures [14]. In their scheme, the performance would be at its best when both numbers of row and column square roots are equal to the message's numbers. Let's assume there are 25 signatures, the scheme is the most efficient one and the verifier needs to execute 10 exponential computations when the matrix has 5 rows and 5 columns. If there is one illegal signature, the verifier needs to execute 10 exponential computations. If there are two illegal signatures, the verifier needs to execute 14 exponential computations to detect the illegal signatures.

In this paper, we present a new batch verifying scheme which is especially efficient when there are illegal signatures. When the verifier receives t signatures, it generates a cube of side length n and fills these t signatures in the

$n \times n \times n$ cube, where n is the smallest integer which satisfies $n^3 \geq t$. Let's assume there are 25 signatures, the verifier generates a cube of side length 3 and executes $3 + 3 + 3 = 9$ exponential computations since 3 is the smallest integer which satisfies $3^3 \geq 25$. Moreover, the verification time would not increase as the number of the illegal signatures increases.

The paper is organized as follows: In Section 2, we review two batch verifying schemes including Harn's and Li's scheme. Then we present the proposed scheme and compare its performance with that of previous schemes in Sections 3 and 4, respectively. In Section 5, an extended batch verification scheme is described. Finally, we conclude our paper in Section 6.

2 Two Batch Verifying Scheme

We review two batch verifying schemes before presenting the proposed one.

2.1 Harn's Scheme

In this section, we first introduce Harn's batch verifying scheme [5]. Let's assume p and q are two prime numbers, and $N = pq$. e and d are presented as the signer's public key and private key respectively, which satisfies $ed \equiv 1 \bmod \varphi(N)$, and $\varphi(\cdot)$ is the Euler function. $h(\cdot)$ is a public one-way hash function.

We suppose Alice sends the messages M_0, M_1, $\cdots$, M_{t-1} and signatures S_0, S_1, $\cdots$, S_{t-1} to Bob, where $S_i = h(M_i)^d \bmod N$, $(i = 0, 1, \cdots, t-1)$. Bob can verify these signatures using Alice's public key e by the following equation:

$$(\prod_{i=0}^{t-1} S_i)^e \stackrel{?}{=} \prod_{i=0}^{t-1} h(M_i). \qquad (1)$$

If Equation (1) holds, $(S_0, S_1, \cdots, S_{t-1})$ are valid signatures of M_0, M_1, $\cdots$, M_{t-1}, respectively. In Harn's scheme, these signatures can be verified simultaneously in one exponential operation time.

2.2 Li et al.'s Scheme

The scheme was proposed by Li et al. recently [14]. When the verifier receives the messages (M_1, S_1), (M_2, S_2), $\cdots$, (M_t, S_t) from the signer, the verifier will generate an $m \times n$ matrix (where $m \times n \geq t$) and t random numbers r_i, $i = 1, 2, \ldots, t$, where $r_i \in \{1, 2, \ldots, t\}$. He then randomly fills these t messages into the $m \times n$ matrix using the following equation (see Table 1):

$$S(m,n) = \begin{cases} S(\lceil r_i/n \rceil, n), & \text{if } r_i \bmod n = 0 \\ S(\lceil r_i/n \rceil, r_i \bmod n), & \text{otherwise.} \end{cases} \qquad (2)$$

After filling these messages in the $m \times n$ matrix, the verifier could batch verify each of the rows and the columns, respectively. The complete batch verifying process is divided into two verifications: row verification and column verification. The details of row and column verification are shown as follows.

Table 1: An $m \times n$ matrix

S(1,1)	S(1,2)	...	S(1,n-1)	S(1,n)
S(2,1)	S(2,2)	...	S(2,n-1)	S(2,n)
⋮	⋮	⋮	⋮	⋮
S(m-1,1)	S(m-1,2)	...	S(m-1,n-1)	S(m-1,n)
S(m,1)	S(m,2)	...	S(m,n-1)	S(m,n)

- Row verification:

$$\text{First row: } (\textstyle\prod_{i=1}^{n} S_{(1,i)})^e \stackrel{?}{=} \prod_{i=1}^{n} h(M_{(1,i)}) \bmod N,$$
$$\text{Second row: } (\textstyle\prod_{i=1}^{n} S_{(2,i)})^e \stackrel{?}{=} \prod_{i=1}^{n} h(M_{(2,i)}) \bmod N,$$
$$\vdots$$
$$m\text{-th row: } (\textstyle\prod_{i=1}^{n} S_{(m,i)})^e \stackrel{?}{=} \prod_{i=1}^{n} h(M_{(m,i)}) \bmod N.$$

- Column verification:

$$\text{First column: } (\textstyle\prod_{i=1}^{m} S_{(i,1)})^e \stackrel{?}{=} \prod_{i=1}^{m} h(M_{(i,1)}) \bmod N,$$
$$\text{Second column: } (\textstyle\prod_{i=1}^{m} S_{(i,2)})^e \stackrel{?}{=} \prod_{i=1}^{m} h(M_{(i,2)}) \bmod N,$$
$$\vdots$$
$$n\text{-th column: } (\textstyle\prod_{i=1}^{m} S_{(i,n)})^e \stackrel{?}{=} \prod_{i=1}^{m} h(M_{(i,n)}) \bmod N.$$

If there are some signature-verification faults in the matrix, we could find out where these signature-verification faults are located by finding the matrix positions of row and column overlaps.

Table 2: An 5×5 matrix

S(1,1)	S(1,2)	S(1,3)	S(1,4)	S(1,5)
S(2,1)	S(2,2)	S(2,3)	S(2,4)	S(2,5)
S(3,1)	S(3,2)	S(3,3)	S(3,4)	S(3,5)
S(4,1)	S(4,2)	S(4,3)	S(4,4)	S(4,5)
S(5,1)	S(5,2)	S(5,3)	S(5,4)	S(5,5)

Suppose Alice sends 25 messages to Bob, then Bob will generate 25 random numbers and a 5×5 matrix shown as Table 2. After batch verifying each of the rows and the columns, Bob could easily realize there was a signature-verification fault occurring and precisely detects where the signature-verification fault is located. Assume there was one signature-verification fault in the position $S(3,3)$ of matrix, there would occur two verification fails and these two fails would occur in the third row and the third column, respectively. According to the verification fails of the third row and the third column overlaps, the signature-verification fault could be precisely detected in the position $S(3,3)$ of matrix. However, it is possible for the verifier to execute additional operations if two illegal signatures occur. As shown in [14], the number of total verification is 10 if two signature-verification faults are occurring on the same row or on the same column, and the number is 14 if two illegal signatures are occurring on adjacent diagonal or not occurring on the same row or not on the same column. Please refer to [14] for more details.

3 The Proposed Scheme

We now present a batch verifying multiple signatures scheme which is more efficient than the previous ones, especially when the illegal signature occurs. The details of our scheme are described as follows.

First, the verifier generates a cube with side length m when he receives some pairs of message and signature $(M_0, S_0), (M_1, S_1), \cdots, (M_{t-1}, S_{t-1})$ from the signer, where m is the smallest integer which satisfies $m^3 \geq t$.

Next, the verifier chooses t random numbers r_i, where $r_i \in \{0, 1, \cdots, m^3 - 1\}$, $i = 0, 1, \cdots, t-1$, and fills these t signatures in the $m \times m \times m$ cube according to coordinate figure (x, y, z), where

$$r_i = xm^2 + ym + z, \quad \text{and} \quad x, y, z \in \{0, 1, \cdots, m-1\}. \quad (3)$$

Finally, the verifier could then batch verify each plane according to the three coordinate axes. The details are shown as follows.

- x-axis plane:

$$x = 0\text{: } \left(\prod_{i=0}^{m-1} \prod_{j=0}^{m-1} S_{(0,i,j)}\right)^e \overset{?}{=} \prod_{i=0}^{m-1} \prod_{j=0}^{m-1} h(M_{(0,i,j)}),$$
$$x = 1\text{: } \left(\prod_{i=0}^{m-1} \prod_{j=0}^{m-1} S_{(1,i,j)}\right)^e \overset{?}{=} \prod_{i=0}^{m-1} \prod_{j=0}^{m-1} h(M_{(1,i,j)}),$$
$$\vdots$$
$$x = m-1\text{:}$$
$$\left(\prod_{i=0}^{m-1} \prod_{j=0}^{m-1} S_{(m-1,i,j)}\right)^e \overset{?}{=} \prod_{i=0}^{m-1} \prod_{j=0}^{m-1} h(M_{(m-1,i,j)}).$$

- y-axis plane:

$$y = 0\text{: } \left(\prod_{i=0}^{m-1} \prod_{j=0}^{m-1} S_{(i,0,j)}\right)^e \overset{?}{=} \prod_{i=0}^{m-1} \prod_{j=0}^{m-1} h(M_{(i,0,j)}),$$
$$y = 1\text{: } \left(\prod_{i=0}^{m-1} \prod_{j=0}^{m-1} S_{(i,1,j)}\right)^e \overset{?}{=} \prod_{i=0}^{m-1} \prod_{j=0}^{m-1} h(M_{(i,1,j)}),$$
$$\vdots$$
$$y = m-1\text{:}$$
$$\left(\prod_{i=0}^{m-1} \prod_{j=0}^{m-1} S_{(i,m-1,j)}\right)^e \overset{?}{=} \prod_{i=0}^{m-1} \prod_{j=0}^{m-1} h(M_{(i,m-1,j)}).$$

- z-axis plane:

$$z = 0\text{: } \left(\prod_{i=0}^{m-1} \prod_{j=0}^{m-1} S_{(i,j,0)}\right)^e \overset{?}{=} \prod_{i=0}^{m-1} \prod_{j=0}^{m-1} h(M_{(i,j,0)}),$$
$$z = 1\text{: } \left(\prod_{i=0}^{m-1} \prod_{j=0}^{m-1} S_{(i,j,1)}\right)^e \overset{?}{=} \prod_{i=0}^{m-1} \prod_{j=0}^{m-1} h(M_{(i,j,1)}),$$
$$\vdots$$
$$z = m-1\text{:}$$
$$\left(\prod_{i=0}^{m-1} \prod_{j=0}^{m-1} S_{(i,j,m-1)}\right)^e \overset{?}{=} \prod_{i=0}^{m-1} \prod_{j=0}^{m-1} h(M_{(i,j,m-1)}).$$

If there are some signature-verification faults in the cube, we could find out where these faults are located by finding the point of intersection of three kinds of plane. As shown in Figure 1, there is a signature-verification fault in the position (a, b, c) of the cube if three verifications fail in the $x = a$, $y = b$, and $z = c$ plane, respectively.

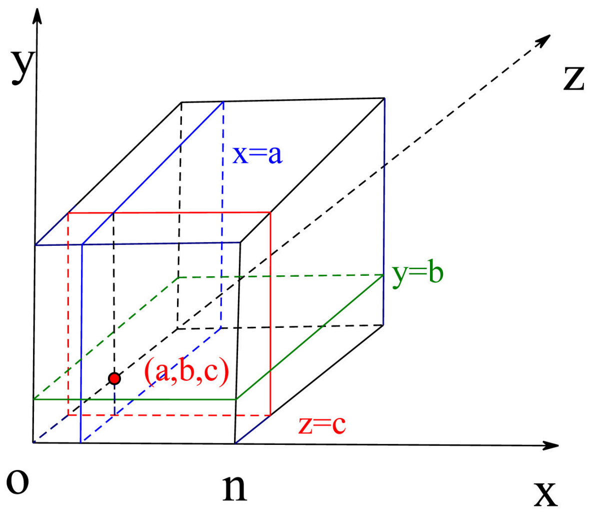

Figure 1: An $m \times m \times m$ cube

We will now give a simple example to show the correctness of our scheme. Let's suppose Alice have sent 64 messages to Bob, then Bob will choose 64 random numbers and generate a $4 \times 4 \times 4$ cube as shown in Figure 2. If $r_0 = 22$, the pair (M_0, S_0) would be filling in the position $(1, 1, 2)$ of the cube since $22 = 1 \cdot 4^2 + 1 \cdot 4 + 2$. If $r_1 = 45$, the pair (M_1, S_1) would be filling in the position $(2, 3, 1)$ of the cube because $45 = 2 \cdot 4^2 + 3 \cdot 4 + 1$. The rest can be deduced similarly by Equation (2). After filling 64 signatures in the cube, Bob could then batch verify three kinds of plane by the method described above.

- x-axis plane:

$$x = 0\text{: } \left(\prod_{i=0}^{3} \prod_{j=0}^{3} S_{(0,i,j)}\right)^e \overset{?}{=} \prod_{i=0}^{3} \prod_{j=0}^{3} h(M_{(0,i,j)}),$$
$$x = 1\text{: } \left(\prod_{i=0}^{3} \prod_{j=0}^{3} S_{(1,i,j)}\right)^e \overset{?}{=} \prod_{i=0}^{3} \prod_{j=0}^{3} h(M_{(1,i,j)}),$$
$$x = 2\text{: } \left(\prod_{i=0}^{3} \prod_{j=0}^{3} S_{(2,i,j)}\right)^e \overset{?}{=} \prod_{i=0}^{3} \prod_{j=0}^{3} h(M_{(2,i,j)}),$$
$$x = 3\text{: } \left(\prod_{i=0}^{3} \prod_{j=0}^{3} S_{(3,i,j)}\right)^e \overset{?}{=} \prod_{i=0}^{3} \prod_{j=0}^{3} h(M_{(3,i,j)}).$$

- y-axis plane:

$$y = 0\text{: } \left(\prod_{i=0}^{3} \prod_{j=0}^{3} S_{(i,0,j)}\right)^e \overset{?}{=} \prod_{i=0}^{3} \prod_{j=0}^{3} h(M_{(i,0,j)}),$$
$$y = 1\text{: } \left(\prod_{i=0}^{3} \prod_{j=0}^{3} S_{(i,1,j)}\right)^e \overset{?}{=} \prod_{i=0}^{3} \prod_{j=0}^{3} h(M_{(i,1,j)}),$$
$$y = 2\text{: } \left(\prod_{i=0}^{3} \prod_{j=0}^{3} S_{(i,2,j)}\right)^e \overset{?}{=} \prod_{i=0}^{3} \prod_{j=0}^{3} h(M_{(i,2,j)}),$$
$$y = 3\text{: } \left(\prod_{i=0}^{3} \prod_{j=0}^{3} S_{(i,3,j)}\right)^e \overset{?}{=} \prod_{i=0}^{3} \prod_{j=0}^{3} h(M_{(i,3,j)}).$$

- z-axis plane:

$$z = 0\text{: } \left(\prod_{i=0}^{3} \prod_{j=0}^{3} S_{(i,j,0)}\right)^e \overset{?}{=} \prod_{i=0}^{3} \prod_{j=0}^{3} h(M_{(i,j,0)}),$$
$$z = 1\text{: } \left(\prod_{i=0}^{3} \prod_{j=0}^{3} S_{(i,j,1)}\right)^e \overset{?}{=} \prod_{i=0}^{3} \prod_{j=0}^{3} h(M_{(i,j,1)}),$$
$$z = 2\text{: } \left(\prod_{i=0}^{3} \prod_{j=0}^{3} S_{(i,j,2)}\right)^e \overset{?}{=} \prod_{i=0}^{3} \prod_{j=0}^{3} h(M_{(i,j,2)}),$$
$$z = 3\text{: } \left(\prod_{i=0}^{3} \prod_{j=0}^{3} S_{(i,j,3)}\right)^e \overset{?}{=} \prod_{i=0}^{3} \prod_{j=0}^{3} h(M_{(i,j,3)}).$$

Table 3: Experiment results of RSA, Harn, Li, and our schemes

Method	Message	Size of Row (Column)	Size of x (y, z)-axis	Size of exponentiation
RSA [18]	25	—	—	25
Harn [5]	25	—	—	1
Li [14]	25	5	—	10
Ours	25	—	3	9
Method	Message	Size of Row (Column)	Size of x (y, z)-axis	Size of exponentiation
RSA [18]	100	—	—	100
Harn [5]	100	—	—	1
Li [14]	100	10	—	20
Ours	100	—	5	15
Method	Message	Size of Row (Column)	Size of x (y, z)-axis	Size of exponentiation
RSA [18]	256	—	—	256
Harn [5]	256	—	—	1
Li [14]	256	16	—	32
Ours	256	—	7	21

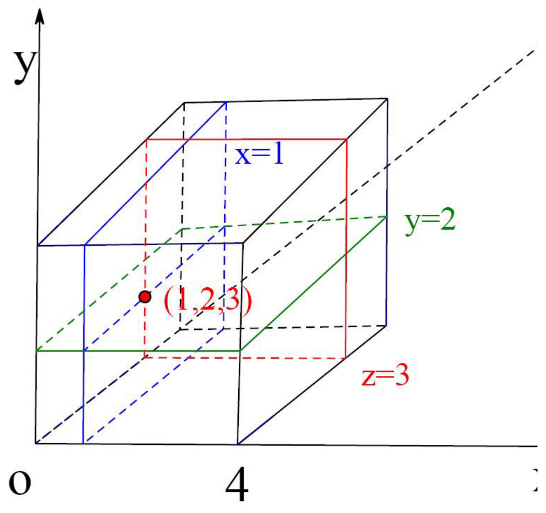

Figure 2: An $4 \times 4 \times 4$ cube

After batch verifying each plane, Bob is now confirmed whether the signature-verification fault is occurring or not. We suppose there was one signature-verification fault in the position $(1, 2, 3)$ of the cube, so Bob could realize there was a signature-verification fault occurring and precisely detects where the signature-verification fault is located. From the method described above, there would occur three verification that fails in the $x = 1$, $y = 2$, and $z = 3$ plane, respectively. According to the point of intersection of three kinds of plane, the signature-verification fault could be precisely detected in the position $(1, 2, 3)$ of the cube.

4 Implementation and Result Analysis

4.1 Experimental Results

In this section, we implement RSA, Harn's scheme, Li's scheme, and ours, with the experimental results of four schemes shown in Table 3. From Table 3, we have concluded that Harn's scheme is better when the batch verification of multiple signatures have succeeded. However, it must re-verify all signatures if there are illegal ones as presented in Section 1. Thus, the performance of Harn's scheme is worse than Li's and ours when illegal signatures occur. In addition, our scheme needs less exponentiation operations for the same number of messages. Therefore, the performance of our scheme is best in most situations.

In Table 3, we compared the sizes of exponentiation operations for different number of signatures in all four schemes. In RSA scheme, the verifier needs to verify the signatures one by one, so the sizes of exponentiation operations are 25, 100, and 256, respectively. As described in Section 2, the verifier can determine the correctness of signature by using one exponential operation in Harn's scheme. Therefore, we only need to show the size of exponentiation operations in Li's and our scheme. In Li's scheme, the verifier can obtain $5 \times 5, 10 \times 10$, and 16×16 matrixes and executes $5 + 5 = 10, 10 + 10 = 20$, and $16 + 16 = 32$ exponentiation operations for $25, 100$, and 256 signatures since $5, 10$ and 16 are the square root of 25, 100, and 256, respectively. In our scheme, the verifier can generate $3 \times 3 \times 3, 5 \times 5 \times 5$, and $7 \times 7 \times 7$ cubes and executes $3+3+3 = 9$, $5+5+5 = 15$, and $7+7+7 = 21$ exponentiation operations since 3, 5, 7 are the smallest integer which satisfies $3^3 \geq 25$, $5^3 \geq 100$, and $7^3 \geq 256$,

Table 4: Comparisons for detecting illegal signatures among RSA, Harn, Li, and our schemes

Method	Message	Size of exponentiation (one illegal signature)	Size of exponentiation (two illegal signatures)
RSA [18]	25	25	25
Harn [5]	25	26	26
Li [14]	25	10	14
Ours	25	9	9
Method	Message	Size of exponentiation (one illegal signature)	Size of exponentiation (two illegal signatures)
RSA [18]	100	100	100
Harn [5]	100	101	101
Li [14]	100	20	24
Ours	100	15	15
Method	Message	Size of exponentiation (one illegal signature)	Size of exponentiation (two illegal signatures)
RSA [18]	256	256	256
Harn [5]	256	257	257
Li [14]	256	32	36
Ours	256	21	21

respectively.

4.2 Analysis of Illegal Signature Detection

In this section, we now present the size of exponentiation operations in four schemes when illegal signatures occur. As described in Section 1, the performance of illegal signatures detection is regarded as the position it is located in. From Table 4, we know that our scheme is better than Harn's and Li's for determining the situation where illegal signatures are located. Once the total number of signatures is fixed, the size of exponentiation operations is independent from the number of illegal signatures in our scheme.

In Table 4, we compared the sizes of exponentiation for detecting one and two illegal signatures among RSA, Harn, Li, and our schemes. In RSA scheme, the verifier needs to verify the signatures one by one, so the sizes of exponentiation operations are 25, 100, and 256 respectively. In Harn's scheme, the verifier must re-verify each signatures if there are illegal ones, so he needs to execute 26, 101, and 257 exponentiation operations, respectively. As shown in Section 2.2, in Li's scheme, one illegal signature can be detected accurately after batch verification finished, and the verifier must add 4 exponentiation operations if there are two illegal signatures. From Table 3, we know that 10, 20, and 32 operations are needed for 25, 100, and 256 signatures respectively in Li's scheme. Thus, the verifier executes $10, 20,$ and 32 operations when one illegal signature occurs; and $14, 24,$ and 36 operations if there are two illegal signatures in Li's scheme. In our scheme, the position of illegal ones can be determined accurately once batch verification is finished and the size of operations are independent from the number of signatures. From Table 3, we know that $9, 15,$ and 21 operations are needed for $25, 100,$ and 256 signatures respectively in our scheme. Therefore, the sizes of exponentiation operations are $9, 15,$ and 21 in our scheme whether one or two faults occurred.

5 The Extension of the Scheme

The proposed scheme is based on a cube, and we can extend it to the condition of n-dimension. First, the verifier generates an n-dimension object with side length m when he receives some pairs of message and signature $(M_0, S_0), (M_1, S_1), \cdots, (M_{t-1}, S_{t-1})$ from the signer, where m is the smallest integer which satisfies $m^n \geq t$.

Next, the verifier chooses t random numbers r_i, where $r_i \in \{0, 1, \cdots, m^n - 1\}$, $i = 0, 1, \cdots, t-1$, and fills these t messages in the m^n object according to coordinate figure $(a_{n-1}, a_{n-2}, \cdots, a_1, a_0)$, where $a_{n-1}, \cdots, a_1, a_0 \in \{0, 1, \cdots, m-1\}$ and

$$r_i = a_{n-1}m^{n-1} + a_{n-2}m^{n-2} + \cdots + a_1 m + a_0.$$

Finally, the verifier could then batch verify each plane according to n-dimension coordinate axis. The details are described as follows.

1) a_{n-1}-axis plane:

a. $a_{n-1} = 0$:

$$\left(\prod_{a_{n-2}=0}^{m-1} \cdots \prod_{a_0=0}^{m-1} S_{(0,a_{n-2},\cdots,a_0)}\right)^e \overset{?}{=} \prod_{a_{n-2}=0}^{m-1} \cdots \prod_{a_0=0}^{m-1} h(M_{(0,a_{n-2},\cdots,a_0)}).$$

b. $a_{n-1}=1$:

$$(\prod_{a_{n-2}=0}^{m-1}\cdots\prod_{a_0=0}^{m-1}S_{(1,a_{n-2},\cdots,a_0)})^e \stackrel{?}{=} \prod_{a_{n-2}=0}^{m-1}\cdots\prod_{a_0=0}^{m-1}h(M_{(1,a_{n-2},\cdots,a_0)}).$$

$$\vdots \qquad\qquad \vdots$$

c. $a_{n-1}=m-1$:

$$(\prod_{a_{n-2}=0}^{m-1}\cdots\prod_{a_0=0}^{m-1}S_{(m-1,a_{n-2},\cdots,a_0)})^e \stackrel{?}{=} \prod_{a_{n-2}=0}^{m-1}\cdots\prod_{a_0=0}^{m-1}h(M_{(m-1,a_{n-2},\cdots,a_0)}).$$

2) a_{n-2}-axis plane:

a. $a_{n-2}=0$:

$$(\prod_{a_{n-1}=0}^{m-1}\prod_{a_{n-3}=0}^{m-1}\cdots\prod_{a_0=0}^{m-1}S_{(a_{n-1},0,\cdots,a_0)})^e \stackrel{?}{=} \prod_{a_{n-1}=0}^{m-1}\prod_{a_{n-3}=0}^{m-1}\cdots\prod_{a_0=0}^{m-1}h(M_{(a_{n-1},0,\cdots,a_0)}).$$

b. $a_{n-2}=1$:

$$(\prod_{a_{n-1}=0}^{m-1}\prod_{a_{n-3}=0}^{m-1}\cdots\prod_{a_0=0}^{m-1}S_{(a_{n-1},1,\cdots,a_0)})^e \stackrel{?}{=} \prod_{a_{n-1}=0}^{m-1}\prod_{a_{n-3}=0}^{m-1}\cdots\prod_{a_0=0}^{m-1}h(M_{(a_{n-1},1,\cdots,a_0)})$$

$$\vdots \qquad\qquad \vdots$$

c. $a_{n-2}=m-1$:

$$(\prod_{a_{n-1}=0}^{m-1}\prod_{a_{n-3}=0}^{m-1}\cdots\prod_{a_0=0}^{m-1}S_{(a_{n-1},m-1,\cdots,a_0)})^e \stackrel{?}{=} \prod_{a_{n-1}=0}^{m-1}\prod_{a_{n-3}=0}^{m-1}\cdots\prod_{a_0=0}^{m-1}h(M_{(a_{n-1},m-1,\cdots,a_0)}).$$

$$\vdots$$

3) a_0-axis plane:

a. $a_0=0$:

$$(\prod_{a_{n-1}=0}^{m-1}\cdots\prod_{a_1=0}^{m-1}S_{(a_{n-1},\cdots,a_1,0)})^e \stackrel{?}{=} \prod_{a_{n-1}=0}^{m-1}\cdots\prod_{a_1=0}^{m-1}h(M_{(a_{n-1},\cdots,a_1,0)}).$$

b. $a_0=1$:

$$(\prod_{a_{n-1}=0}^{m-1}\cdots\prod_{a_1=0}^{m-1}S_{(a_{n-1},\cdots,a_1,1)})^e \stackrel{?}{=} \prod_{a_{n-1}=0}^{m-1}\cdots\prod_{a_1=0}^{m-1}h(M_{(a_{n-1},\cdots,a_1,1)}).$$

$$\vdots \qquad\qquad \vdots$$

c. $a_0=m-1$:

$$(\prod_{a_{n-1}=0}^{m-1}\cdots\prod_{a_1=0}^{m-1}S_{(a_{n-1},\cdots,a_1,m-1)})^e \stackrel{?}{=} \prod_{a_{n-1}=0}^{m-1}\cdots\prod_{a_1=0}^{m-1}h(M_{(a_{n-1},\cdots,a_1,m-1)}).$$

Therefore, the total number of exponentiation operations is mn in the extended batch verification scheme. If there are some signature-verification faults in the n-dimension object, we could find out where these faults are located by finding the point of intersection of n kinds of plane. For example, there is a signature-verification fault in the position $(0,1,\cdots,m-1)$ of the n-dimension object, if n verifications failed in the $a_{n-1}=0$ plane, $a_{n-2}=1$ plane, $\cdots$ and $a_0=m-1$ plane, respectively.

6 Conclusions

We presented a new batch verification multiple RSA signatures scheme which fills the signatures into a cube. It can detect accurately where the illegal signatures are located without additional re-verify operations. Moreover, the verification time would not increase as the number of the illegal signatures increases in one batch verification. Experiment shows our scheme is more efficient than the previous schemes, especially when the number of the signatures is very large. We then extended this scheme to the condition of n-dimension.

Acknowledgements

The work described in this paper was supported by the Natural Science Foundation of China (Grant No. 61202367, 61472235), Natural Science Foundation of Shanghai (Grant No. 12ZR1443700), and the Innovation Program of Shanghai Municipal Education Commission (Grant No. 14YZ020).

References

[1] F. Bao, C. C. Lee, M. S. Hwang, "Cryptanalysis and improvement on batch verifying multiple RSA digital signatures," *Applied Mathematics and Computation*, vol. 172, no. 2, pp. 1195-1200, 2006.

[2] T. Cao, D. Lin, and R. Xue, "Security analysis of some batch verifying signatures from pairings," *International Journal of Network Security*, vol. 3, no. 2, pp. 138-143, 2006.

[3] S. W. Changchien, M. S. Hwang, "A batch verifying and detecting multiple RSA digital signatures," *International Journal of Computational and Numerical Analysis and Applications*, vol. 2, no. 3, pp. 303-307, Oct. 2002.

[4] T. Y. Chang, M. S. Hwang, W. P. Yang, and K. C. Tsou, "A modified Ohta-Okamoto digital signature for batch verification and its multi-signature version," *International Journal of Engineering and Industries*, vol. 3, no. 3, pp. 75-83, Sep. 2012.

[5] L. Harn, "Batch verifying multiple RSA digital signatures," *Electronics Letters*, vol. 34, no. 12, pp. 1219-1220, 1998.

[6] M. S. Hwang, K. F. Hwang, I. C. Lin, "Cryptanalysis of the batch verifying multiple RSA digital signatures," *Informatica*, vol. 11, no. 1, pp. 1-4, Jan. 2000.

[7] M. S. Hwang, I. C. Lin, and K. F. Hwang, "Cryptanalysis of the batch verifying multiple RSA digital signatures," *Informatica*, vol. 11, no. 1, pp. 15-19, 2000.

[8] M. S. Hwang and C. C. Lee, "Research issues and challenges for multiple digital signatures," *International Journal of Network Security*, vol. 1, no. 1, pp. 1-7, July 2005.

[9] M. S. Hwang, C. C. Lee, Y. C. Lai, "Traceability on RSA-based partially signature with low computation," *Applied Mathematics and Computation*, vol. 145, no. 2-3, pp. 465-468, Dec. 2003.

[10] M. S. Hwang, C. C. Lee, E. J. L. Lu, "Cryptanalysis of the batch verifying multiple DSA-type digital signatures," *Pakistan Journal of Applied Sciences*, vol. 1, no. 3, pp. 287-288, July 2001.

[11] M. S. Hwang, E. J. L. Lu, I. C. Lin, "Practical (t, n) threshold proxy signature scheme based on the RSA cryptosystem," *IEEE Transactions on Knowledge and Data Engineering*, vol. 15, no. 6, pp. 1552-1560, Nov./Dec. 2003.

[12] S. J. Hwang, M. S. Hwang, and S. F. Tzeng, "A new digital multisignature scheme with distinguished signing authorities," *Journal of Information Science and Engineering*, vol. 19, no. 5, pp. 881-887, Sep. 2003.

[13] K. Kim, I. Yie, S. Lim, and D. Nyang, "Batch verification and finding invalid signatures in a group signature scheme," *International Journal of Network Security*, vol. 13, no. 2, pp. 61-70, 2011.

[14] C. T. Li, M. S. Hwang, "A batch verifying and detecting the illegal signatures," *International Journal of Innovative Computing, Information and Control*, vol. 6, no. 12, pp. 5311-5320, 2010.

[15] C. T. Li, M. S. Hwang, and Y. P. Chu, "An efficient sensor-to-sensor authenticated path-key establishment scheme for secure communications in wireless sensor networks," *International Journal of Innovative Computing, Information and Control*, vol. 5, no. 8, pp. 2107-2124, 2009.

[16] C. T. Li, M. S. Hwang, and C. Y. Liu, "An electronic voting protocol with deniable authentication for mobile ad hoc networks," *Computer Communications*, vol. 31, no. 10, pp. 2534-2540, 2008.

[17] N. Ojha and S. Padhye, "Cryptanalysis of multi prime RSA with secret key greater than public key," *International Journal of Network Security*, vol. 16, no. 1, pp. 53-57, 2014.

[18] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120-126, 1978.

[19] M. Stanek, "Attacking LCCC batch verification of RSA signatures," *International Journal of Network Security*, vol. 6, no. 2, pp. 238-240, 2008.

[20] S. F. Tzeng, C. C. Lee, and M. S. Hwang, "A batch verification for multiple proxy signature," *Parallel Processing Letters*, vol. 21, no. 1, pp. 77-84, 2011.

[21] J. Zhang, M. Xu, and L. Liu, "On the security of a secure batch verification with group testing for VANET," *International Journal of Network Security*, vol. 16, no. 4, pp. 313-320, 2014.

Yanli Ren is an associate professor in School of Communication and Information Engineering at Shanghai University, China. She was awarded a MS degree in applied mathematics in 2005 from Shaanxi Normal University, China, and a Ph.D. degree in computer science and technology in 2009 from Shanghai Jiao Tong University, China. Her research interests include applied cryptography, secure outsourcing computing, and network security.

Shuozhong Wang received BS degree in 1966 from Peking University, P.R. China, and Ph.D. degree in 1982 from University of Birmingham, England. He was with Institute of Acoustics, Chinese Academy of Sciences, from 1983 to 1985 as a research fellow. He joined Shanghai University of Technology in October 1985 as an associate professor. He is now a professor of the School of Communication and Information Engineering, Shanghai University. Professor Wang was a visiting associate scientist at Department of Electrical Engineering and Computer Science, University of Michigan, USA, from March 1993 to August 1994. His research interests include acoustics, image processing, audio processing, and information security.

Xinpeng Zhang received the B.S. degree in computational mathematics from Jilin University, China, in 1995, and the M.E. and Ph.D. degrees in communication and information system from Shanghai University, China, in 2001 and 2004, respectively. Since 2004, he has been with the faculty of the School of Communication and Information Engineering, Shanghai University, where he is currently a Professor. He was with the State University of New York at Binghamton as a visiting scholar from January 2010 to January 2011, and Konstanz University as an experienced researcher sponsored by the Alexander von Humboldt Foundation from March 2011 to May 2012. His research interests include multimedia security, image processing, and digital forensics. He has published more than 170 papers in these areas.

Min-Shiang Hwang received the B.S. in Electronic Engineering from National Taipei Institute of Technology, Taipei, Taiwan, Republic of China, in 1980; the M.S. in Industrial Engineering from National Tsing Hua University, Taiwan, in 1988; and the Ph.D. in Computer and Information Science from National Chiao Tung University, Taiwan, in 1995. He also studied Applied Mathematics at National Cheng Kung University, Taiwan, from 1984-1986. Dr. Hwang passed the National Higher Examination in field "Electronic Engineer" in 1988. He also passed the National Telecommunication Special Examination in field "Information Engineering", qualified as advanced technician the first class in 1990. From 1988 to 1991, he was the leader of the Computer Center at Telecommunication Laboratories (TL), Ministry of Transportation and Communications, ROC. He was also a project leader for research in computer security at TL in July 1990. He obtained the 1997, 1998, and 1999 Distinguished Research Awards of the National Science Council of the Republic of China. He is a member of IEEE, ACM, and Chinese Information Security Association. His current research interests include database and data security, cryptography, image compression, and mobile communications.