FORENSIC SCIENCE JOURNAL SINCE 2002

Forensic Science Journal
2004;3:33-44

The Novel Data Dispersal and Verification Framework for Keeping Digital Evidence

Wen-Chao Yang,[1] MBA; Che-Yen Wen,[2,*] Ph.D.

[1] *Forensic Sciences Laboratory, Central Police University, Taoyuan, Taiwan*
[2] *Department of Forensic Science, Central Police University, Taoyuan, Taiwan*

Received：December 08, 2003/Received in revised form：January 03, 2004/Accepted：February 25, 2004

ABSTRACT

With the progress of computer technologies, the machine "computer" plays an important role in the modern society. However, the criminal problems with computers become more serious. The term of "computer forensics" is the technology of dealing with digital evidence. There are many problems of computer forensics need to deal with, such as recovery, tedious computation, encryption, and etc. In this paper, we concentrate on the recovery problem.We provide a novel technology "the framework of keeping digital evidence" for the recovery process, based on the public-key cryptography, the hash function, and the information dispersal algorithm. The public-key cryptography and the hash function in our framework can authenticate and verify if the digital evidence, (such as the computer processing logs or login logs) is modified. The information dispersal algorithm can assure the transferred data cannot be modified and has the fault tolerance rate n/m. In the framework, we design a checking function to solve the tediously checking process. The function can decrease the mean of check times from $\frac{n+1}{2}$ to $\frac{2n+13}{8}$. We use a simple experiment to verify the accuracy of our framework.

Key words: Computer Crime, Computer Forensic, Digital Evidence, Recovery

The digital magnetic records include the computer log files, text-format files, machine codes, digital video and images files, and so on. If the digital magnetic records relating to crime scenes are presented accurately in court by an appropriate procedure, they are called "digital evidence".

"As more criminals utilize technology to achieve their goals and avoid apprehension, there is a developing need for individuals who can analyze and utilize evidence stored on and transmitted using computers." said by Eoghan Casey and Keith Seglem [1]. When the computer crimes and information security events increase, the computer forensic technology becomes more important.

The digital magnetic records in the computer system can be divided as two categories: one is "apparent data" that can be operated and observed directly; the other one is "latent data" (such as deleted data or swapped sections) that can be observed only with special tools [2].

The recovery process plays an important role in the computer forensic procedures, and it can recover all digital magnetic records (both apparent and latent data). A successful recovery process can help us get the clues for investigation and the valid evidence for a law court. However, we sometimes meet a difficult problem (named the recovery problem) in practice --- how to find out the clues or evidence from the wiped or forged digital records. This problem is unsolved in computer forensics currently.

In this paper, we provide a novel technology for solving the recovery problem. This technology contains a novel digital records dispersal and verification framework for keeping digital evidence. The framework is

* Corresponding author,e-mail: cwen@mail.cpu.edu.tw

based on the public-key cryptography, hash function and information dispersal techniques. RSA is a well-known algorithm for authenticating the digital records. RSA (for public-key cryptography) [3] and SECURE HASH STANDARD-1 (SHA-1; hash function)[4] can encrypt and verify the digital records without modification and illegal access; in other words, they can assure the admissibility of the digital evidence by Federal Rules of Evidence 803 [5] & 902 [6]. In this paper, we use RSA (with the public-key of the police department) and SHA-1 to convert the digital magnetic records into digital evidence and verify the evidence, respectively. The information dispersal algorithm (IDA) is used to backup the digital evidence for assuring the data available. It is also used to "secret sharing".

Methods

In this section, we will summarize techniques used in our framework, including the public-key cryptography, hash function, and secret sharing algorithm.

The framework of keeping digital evidence

Our framework is illustrated in Fig.1, and summarized as follows:

Assume low data is *Logi*,

The production procedure:

1. To use the SHA-1 hash function and XOR operator:

$$Log_i, H_{SHA-1}(Log_i), XOR(H_{SHA-1}(Log_{i-1}), H_{SHA-1}(Log_i)) \quad (1)$$

2. To use the RSA encrypt with the police department public key, Kpub-cid, we get Ed_i (Encrypt Data):

$$Ed_i = E_{K_{pub-cid}}(Log_i, H_{SHA-1}(Log_i)), XOR(H_{SHA-1}(Log_{i-1}), H_{SHA-1}(Log_i)) \quad (2)$$

3. To disperse with IDA (assume disperse to *n* pieces and get *m* can reconstruct, thus the vectors denotes), then get n Ceils:

$$\{time_i, vector_{ij}, IDA(Ed_i), H_{SHA-1}(time_i, vector_{ij}, IDA(Ed_i))\}, \quad (3)$$

The acquirement and testify procedure:

1. To get more than m Ceils and verify the Ceils if been modified?

   We compare the $NewHash_{ij}(time_i, vector_{ij}, IDA(Ed_i))$ with $H_{SHA-1}(time_i, vector_{ij}, IDA(Ed_i))$

2. To choose not been modified *m* Ceils to reconstruct Ed_i (IDA^{-1}), then we get Ed_i

$$Ed_i = E_{K_{pub-cid}}(Log_i, H_{SHA-1}(Log_i)), XOR(H_{SHA-1}(Log_{i-1}), H_{SHA-1}(Log_i)) \quad (4)$$

3. According the XOR value, forensic examiners can verify two Logs with Privacy-Key in a time XOR check process.
4. After find out the modified Log_k, forensic examiners can decrypt Ed_k with Privacy-Key and use the original Log_k to investigate.

We use Fig.2 as an example to explain the check process in our framework.

Assume we have the *n* rows log data and make sure the criminal offense has been recorded in one row, which has been forged (or wipe) as soon as (as Fig. 2). In the traditional way, we can just verify one row at each time (from the *n*th row to the first row), and the expected value of all check times is .

Our check process can verify two rows in each check time (from the *n*th row to the first row). The check times can be estimated in two cases:

(1) If the *n* is odd number, the expected value of all check times is

$$\left\{(0+2)\times\frac{1}{\left(\frac{n-1}{2}+1\right)}+(1+2)\times\frac{1}{\left(\frac{n-1}{2}+1\right)}+\ldots+\left[\frac{(n-1)}{2}+2\right]\times\frac{1}{\left(\frac{n-1}{2}+1\right)}\right\}$$

$$\Rightarrow\left\{\left[2+\frac{(n+3)}{2}\right]\times\frac{n+1}{2}\div 2\right\}\times\frac{1}{\frac{n+1}{2}}\Rightarrow\frac{n+7}{4}$$ .Since the "first check" (to check the *n*th row) and the "last check" (to find out which row is the forged row in the two rows, where we discover the error) must be executed, we add "2" at each item.

(2) If the *n* is even number, the expected value is

$$\left\{(0+2)\times\frac{1}{\left(\frac{n}{2}+1\right)}+(1+2)\times\frac{1}{\left(\frac{n}{2}+1\right)}+\ldots+\left(\frac{n-2}{2}+2\right)\times\frac{1}{\left(\frac{n}{2}+1\right)}+\left(\frac{n}{2}+1\right)\times\frac{1}{\left(\frac{n}{2}+1\right)}\right\}$$

$$\Rightarrow\left\{\left(2+\frac{n+2}{2}\right)\times\frac{n+2}{2}\div 2\right\}\times\frac{1}{\frac{n+2}{2}}\Rightarrow\frac{n+6}{4}$$

From (1) and (2), we obtain the expected value of all check times as .

$$\left(\frac{n+7}{4}+\frac{n+6}{4}\right)\div 2\Rightarrow\frac{2n+13}{8}$$

We can reduce the expected value of all check times from $\frac{n+1}{2}$ to $\frac{2n+13}{8}$ in the check process.

Public-key Cryptography

In the proposed framework, we apply the public-key cryptography with the public-key of the police department to convert the digital magnetic records into digital evidence. In the process, we can obtain the chain of custody and admission of the converted digital evidence.

The public-key cryptography includes two parts: encrypting data with the public key and decrypting the encrypted data with the pair privacy key. We can encrypt the system log with the public key of the police department or the third authentic institution, and be sure the data cannot be modified by anyone without the privacy key.

Up to now, the public-key cryptography has two most popular algorithms: RSA and Elliptic Curve Cryptography (ECC). They have been used in electronic commercial products for years. Since the RSA algorithm is mature, we choose it in our framework.

The RSA algorithm is based upon the Fermat's and Euler's theorems [3,7]:

(1) Fermat's theorem

If *p* is prime and *a* is a positive integer not divisible by *p*, then

$$a^{p-1}\equiv 1 \bmod p\,. \qquad (5)$$

(2) Euler's theorem

For every *a* and *n* that are relatively prime

$$a^{\phi(n)}\equiv 1 \bmod n\,. \qquad (6)$$

The RSA algorithm can be summarized as follows:

We first choose two primes, *p* and *q*, and compute

$$n = pq\,. \qquad (7)$$

We randomly choose the encryption key, *e*, such that *e* and (*p*-1)(*q*-1) are relatively primes. Finally, we use the Euclidean algorithm to compute the decryption key, *d*, such as

$$\text{.} \qquad (8)$$

The numbers *e* and *n* are the public keys; the numbers *e* and *d* are the privacy keys.

To encrypt a message *m*, we first divide it into numerical blocks (smaller than *n*). The encrypted message, *c*, will be made up of similar size messages blocks, c_i. The encryption formula is

$$c_i = m_i^e \bmod n\,. \qquad (9)$$

To decrypt an encrypted message, we take each encrypted block c_i and compute

$$m_i = c_i^d \bmod n \quad . \qquad (10)$$

By the Fermat's and Euler's theorems (all mod *n*), we can obtain

$$c_i^d = (m_i^e)^d = m_i^{ed} = m_i^{k(p-1)(q-1)+1}$$
$$= m_i m_i^{k(p-1)(q-1)} = m_i * 1 = m_i \qquad (11)$$

Hash Function

We use the hash function and XOR operator to make sure that the digital evidence is not modified, or we can quickly find out if they are modified.

The hash function is designed to be a one-way function, and used for authentication or error-detection. It accepts a variable-size message *m* as the input and produces a fix-size hash code *H(m)*, sometimes called a message digest, as the output. The hash code is a function with the bit message, and it provides a capability for error-detection: any change to any bit or bits in the message will result in a change to the hash code.

As the public-key cryptography, the hash function has various kinds of algorithms that have been applied in electronic commercial products. MD5, SHA-1, and RIPEMD-160, are most popular hash algorithms. Because MD5 is highly vulnerable to the brute-force attack and RIPEMD-160 needs tedious computation, we choose the SHA-1 hash function in our framework.

SHA was developed by the National Institute of Standards and Technology (NIST) and published as a federal information processing standard (FIPS PUB 180) in 1993. A revised version was issued as FIPS PUB 180-1 in 1995 and is generally referred to as SHA-1 [3,8].

SHA-1 takes a message with a maximum length of less than 2^{64} bits as input and a 160-bit message digest as output. The input is processed in 512-bit blocks. The overall processing of a message is shown in Fig. 3 The processing consists of the following steps [8]:

Step 1: Append padding bits.

The message is padding so that its length is congruent to 448 modulo 512 (448 mod 512). Therefore the numbers of padding bits is in the range of 1 to 512.

Step 2: Append length

A block of 64 bits, as an unsigned 64-bit integer, is appended to the message.

Step 3: Initialize MD buffer

A 160-bit buffer is used to hold intermediate and final results of the hash function. The buffer can be presented as five 32-bit registers (A, B, C, D, E). These registers are initialized to the following 32-bit integer (base 16):

A = 6 7 4 5 2 3 0 1
B = E F C D A B 8 9
C = 9 8 B A D C F E **Initial Value (IV)**
D = 1 0 3 2 5 4 7 6
E = C 3 D 2 E 1 F 0

Step 4: Process message in 512-bit blocks.

The kernel of the algorithm, illustrated in Fig.4, is a module that consists of four rounds of processing of 20 steps. The four rounds have a similar structure, but each one uses a different primitive logical function (f_1, f_2, f_3, and f_4 in Fig.4).

Each round also makes use of an additive constant K_t in hexadecimal and decimal:

Step Number	Hexadecimal	Take Integer Part of
$0 \le t \le 19$	K_t=5 A 8 2 7 9 9 9	$[2^{30}\times\sqrt{2}\,]$
$20 \le t \le 39$	K_t=6 E D 9 E B A 1	$[2^{30}\times\sqrt{3}\,]$
$40 \le t \le 59$	K_t=8 F 1 B B C D C	$[2^{30}\times\sqrt{5}\,]$
$60 \le t \le 79$	K_t=C A 6 2 C 1 D 6	$[2^{30}\times\sqrt{10}\,]$

Step 5: Output.

After all *L* 512-bit blocks have been processed, the output from the *L*th stage is the 160-bit message digest.

We can summarize the operator of SHA-1 as follows:

CV0 = IV

CVi+1 = SUM_ADD32 (CVi, ABCDEi)

MD = CVL

Where

IV = initial value of the ABCDE buffer, defined in step 3.

$ABCDE_i$ = the output of the last round of processing of the *i*th message block

L = the number of 512-bit blocks in the message

SUM_ADD32 = Addition modulo 232 performed separately on each word of the pair of inputs; the $ABCDE_i$ exchange formula as follow: (⊞is addition modulo 2^{32}, f_i is defined in Fig. 4, S^k means circular left shift (rotation) of the 32-bit argument by *k* bits, W_i is a 32-bit word defined in Fig.4, K_i is an additive constant, as defined previously)

$A \leftarrow E \boxplus f(i, B, C, D) \boxplus S^5(A) \boxplus W_i \boxplus K_i$

$B \leftarrow A$

$C \leftarrow S^{30}(B)$

$D \leftarrow C$

$E \leftarrow D$

MD = final message digest value

Information Dispersal Algorithm (IDA)

In practice, we use the information dispersal algorithm to backup the digital evidence in other computers or storage devices through the LAN. The IDA technique can not only backup the digital evidence but help us to reduce the network transfer rate. We summarize IDA as follows [9]:

IDA separate a file *F* (length *L* = |*F*|) into *n* pieces F_i (ength $|F_i| = \lceil L/m \rceil$), $1 \le i \le n$, and every *m* pieces suffice for reconstructing *F*. The sum of the lengths $|F_i|$ is $\lceil L/m \rceil * n$.

Choose an appropriate integer m so that $n = m + k$ satisfies $n/m \le 1+\varepsilon$ for a specified ε > 0. Choose n vectors $a_i = (a_{i1}, \ldots, a_{im})$ the finite field Z_p^m (p means a prime), $1 \le i \le n$, such that every subset of *m* different vectors are linearly independent. Alternatively, it is reasonable for us to assume that any randomly chosen subsets of m vectors in $\{a_i, \ldots, a_n\}$ are linearly independent .

The file *F* (length = *N*) is segmented into sequences (if there are *k* sequences) of length *m* (if the length of last sequence is less than *m*, we pad 0's to it),

$$F = (b_1,\ldots,b_m),(b_{m+1},\ldots,b_{2m}),\ldots,(b_{(k-1)m+1},\ldots,b_N). \quad (12)$$

Let $S_1 = (b_1,\ldots,b_m)$ and $F_i = c_{i1}, c_{i2},\ldots,c_{i\lceil N/m \rceil}$, for *i* = *1*,...,*n*,

where $c_{ik} = a_i \cdot S_k = a_{i1} \cdot b_{(k-1)m+1} + \ldots + a_{im} \cdot b_{km}$

, it follows that $|F_i| = \lceil N/m \rceil$.

If *m* pieces of *F*, $F_1,\ldots,F_m$, are given, we can reconstruct *F*. Let $A = (a_{ij})_{1 \le i,j \le m}$ be the $m \times m$ matrix which ith row is a_i, we can obtain

$$A \cdot \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix} = \begin{bmatrix} c_{11} \\ \vdots \\ c_{m1} \end{bmatrix}, \quad (13)$$

or

$$\begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix} = A^{-1} \cdot \begin{bmatrix} c_{11} \\ \vdots \\ c_{m1} \end{bmatrix}. \quad (14)$$

If the ith row of A^{-1} is $(\alpha_{i1},\ldots,\alpha_{im})$, then in general, for $1 \le k \le \lceil N/m \rceil$,

$b_j = \alpha_{i1}c_{1k} + \ldots + \alpha_{im}c_{mk}, 1 \le j \le N$,

where $i = j \bmod m$, $k = \lceil j/m \rceil$ (here we take the residues to be 1,...,*m*).

Experiment Results

In order to see the capability of the proposed framework, we use Fig.5 as an example. Fig.5 is an experiment without encryption and hash processes. We disperse the message to 5 ceils. An original message (192. 168.100.100 19/Sep/2003:02:28:23 GET /c/winnt/system32/cmd.exe?/c+dir HTTP/1.0 404 296) is the illegal access fictitious log of IIS, the restored message is the same one. From the experimental result, we only need any three ceils to restore the original message back.

Discussion

In this paper, we proposed a novel digital data dispersal and verification framework for keeping the digital evidence. The framework is design for solving the recovery and tedious computation problem, besides it has three advantages: security, reliability, and flexibility. The most important feature is that our framework doesn't need any new hardware devices or update any network topology.

With our framework, the digital data can become digital evidence and have $\lceil n/m \rceil$ fault-tolerance rate. Besides it can help forensic examiners to reduce the mean of check times from $\frac{n+1}{2}$ to $\frac{2n+13}{8}$. The computer system with our framework can assure the dispersal data integral and fault tolerance. By using the public-key encryption with the public key provided by the police department or the third authentic institution, the digital magnetic data have the admissibility in Federal law to be the digital evidence.

Although this method provides a good solution for the recovery problem, there are still some extended questions need to be dealt with, for example, how to reduce the overhead of the network traffic and the performance of storage devices? How can we combine this method with the general database systems (like MS SQL Server or Oracle Database)? We will work on them in the future work.

References

1. Eoghen C. Handbook of Computer Crime Investigation. Academic Press, 2002.
2. Eoghan C. Digital Evidence and Computer Crime. Academic Press, 2000.
3. Rivest R.L., Shamir A., and Adleman L. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. Communications of the ACM 1978; 21(2):120-6.
4. Federal Information Processing Standards. SECURE HASH STANDARD (SHA). FIPS PUB 180-1, 1993.
5. Federal Rules of Evidence 803, Hearsay Exceptions; Availability of Declarant Immaterial, Data from http://www.law.cornell.edu/rules/fre/C902.html, Date 2003/12/02.
6. Federal Rules of Evidence 902, Self-authentication, Data from http://www.law.cornell.edu/rules/fre/C902.html, Date 2003/12/02.
7. Bruce S. Applied Cryptography. 2nd ed. John Wiley & Sons, 1996.
8. William S. Cryptography and Network Security: Principles and Practices. 2nd ed. Prentice Hall International, 1999.
9. Michael O.R. Efficient Dispersal of Information for Security, Load Balancing, and Fault Tolerance. Journal of ACM 1989:36(2):335-48.

Figure Caption

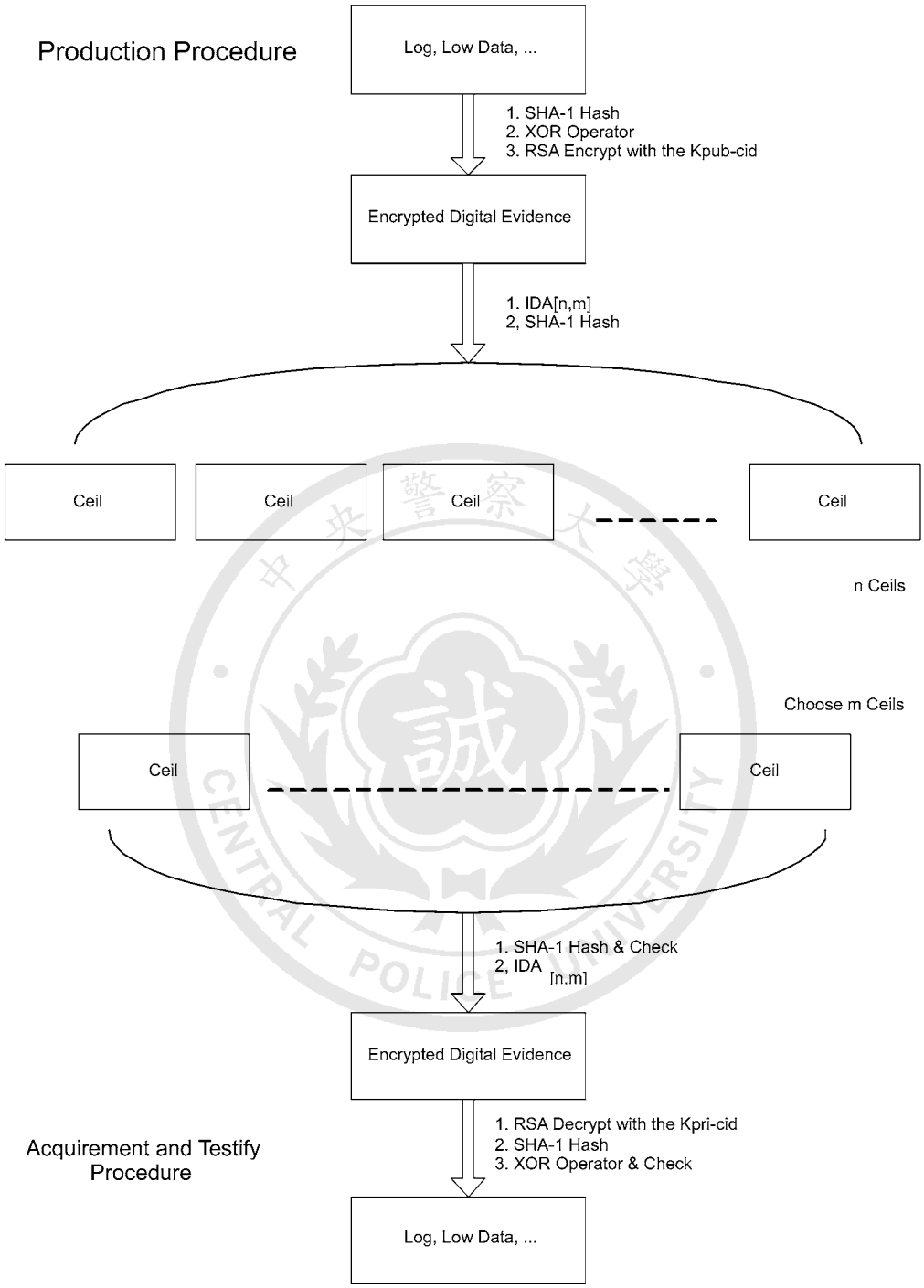

Fig. 1 The framework of keeping digital evidence.

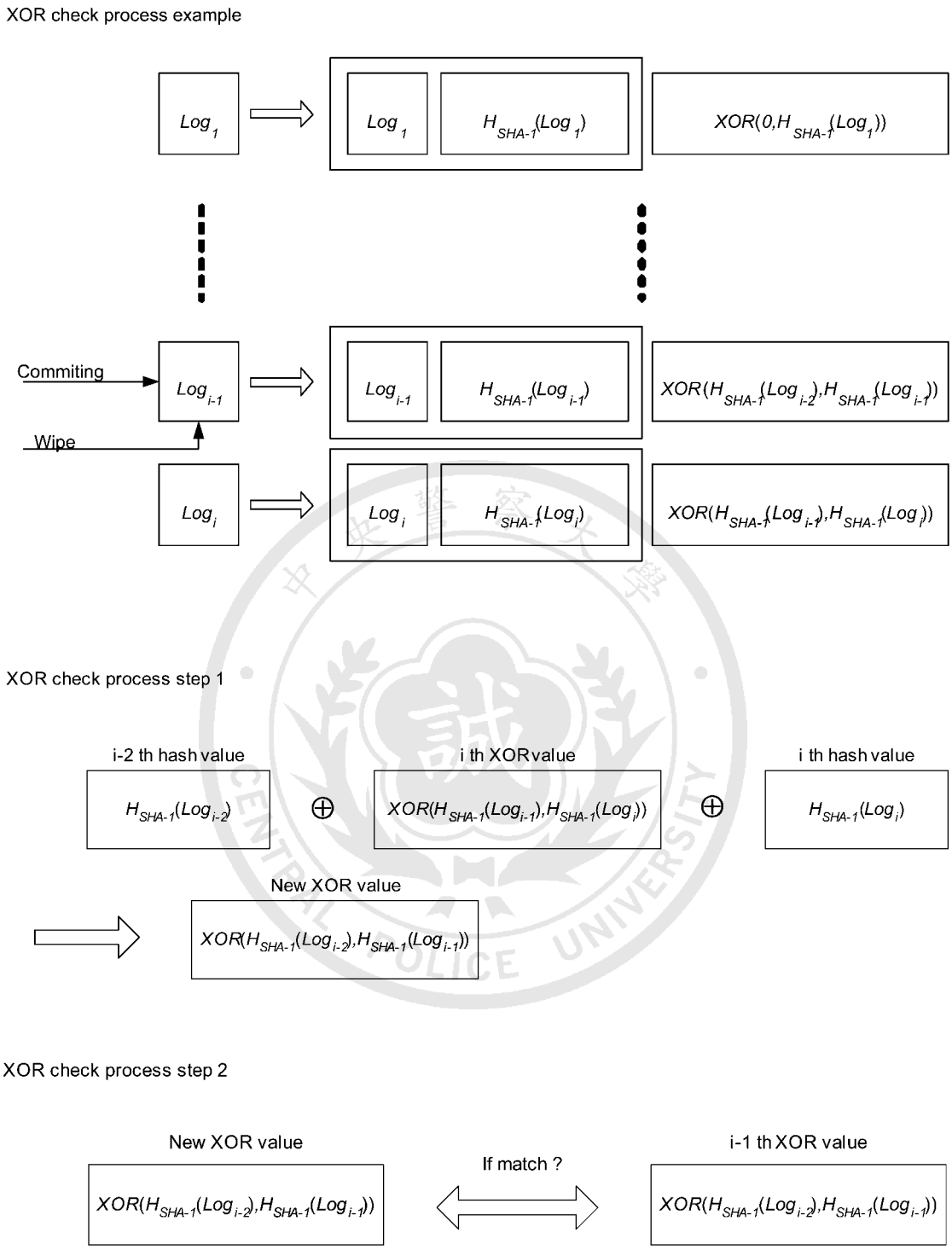

Fig. 2 The XOR check process example (The process can be used to verify log_{i-2}, log_{i-1} at the same time). Assume the hacker or offender commits at $time_{i-1}$, the log_{i-1} will record some criminal offense. During the $time_{i-1}$ to $time_i$, the hacker or offender finishes his criminal offense, and wipes the log_{i-1}, then logout.

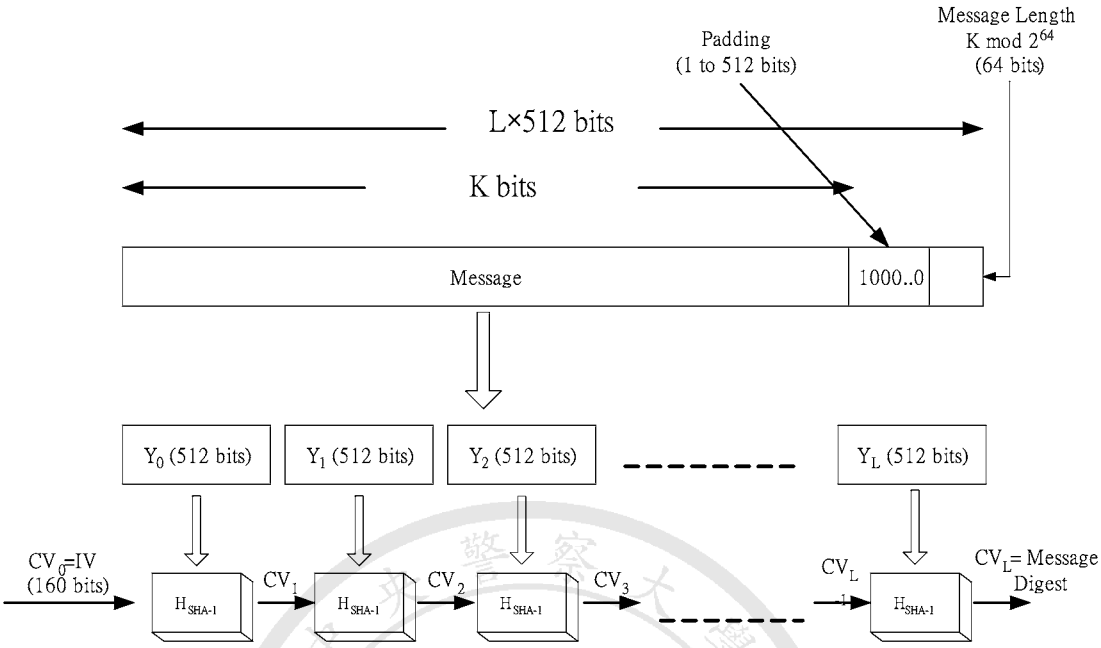

Fig. 3 The SHA-1 overall process of a message.

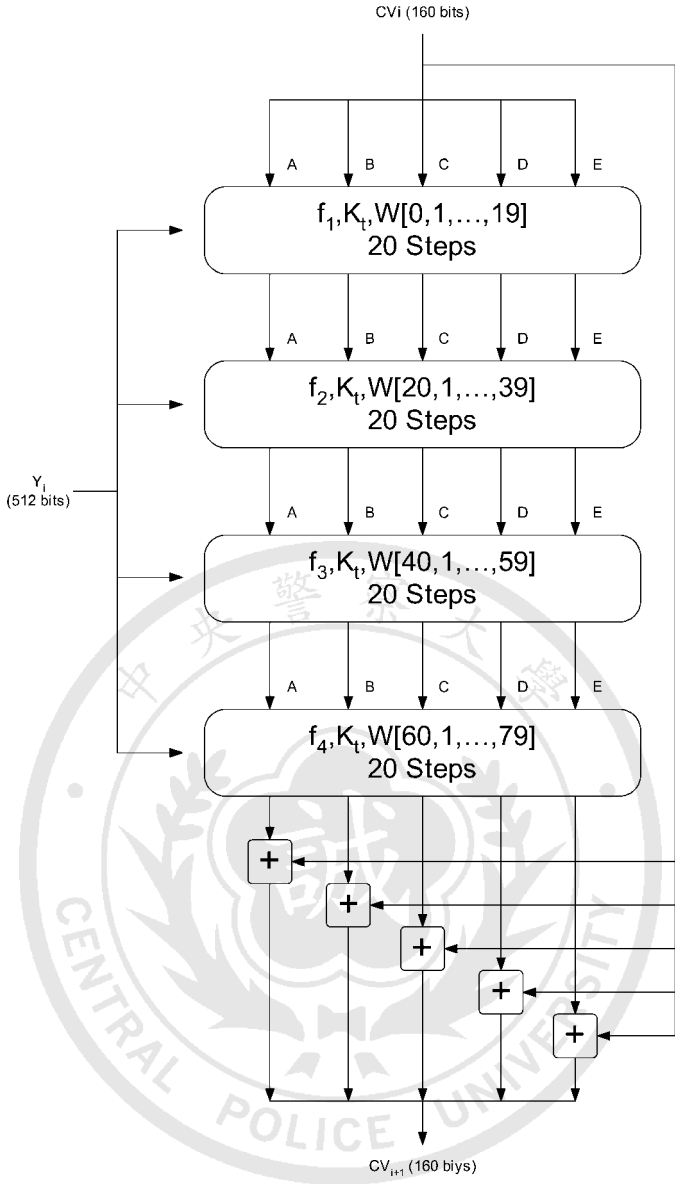

Fig. 4 The SHA-1 process of a single 512-bit block (SHA-1 compression function).

A, B, C, D, E =the five words of the buffer

t =step number $0 \le t \le 79$;

f1, f2, f3, f4 =f(t, B, C, D); primitive logical function for step t, is defined as follow:

Step	Function Name	Function Value
$0 \le t \le 19$	$f_1 = f(t, B, C, D)$	$(B \wedge C) \vee (B \wedge D)$
$20 \le t \le 39$	$f_2 = f(t, B, C, D)$	$B \oplus C \oplus D$
$40 \le t \le 59$	$f_3 = f(t, B, C, D)$	$(B \wedge C) \vee (B \wedge D) \vee (C \wedge D)$
$60 \le t \le 79$	$f_4 = f(t, B, C, D)$	$B \oplus C \oplus D$

Note: The logical operators (AND, OR, XOR) are represented by the symbols ($\wedge$, $\vee$, $\oplus$)

K_t =an addition constant; four distinct values are used, as defined previously

W_t =if ， $0 \le t \le 15$,W_t = 16t, 16t+1, 16t+2, ⋯, 16t+15 bit of Y_i

else, $W_t = S\text{-}1(W_{t-16} \oplus W_{t-14} \oplus W_{t-8} \oplus W_{t-3})$

original message	192.168.100.100 19/Sep/2003:02:28:23 GET /c/winnt/system32/cmd.exe?/c+dir HTTP/1.0 404 296										
original message and SHA-1 hash value	192.168.100.100 19/Sep/2003:02:28:23 GET /c/winnt/system32/cmd.exe?/c+dir HTTP/1.0 404 296 HG7uvnte5bHoFQMe+n0Uqa8XdYY=										
decimal message	49 57 50 46 49 54 56 46 49 48 48 46 49 48 48 32 49 57 47 83 101 112 47 50 48 48 51 58 48 50 58 50 56 58 50 51 32 71 69 84 32 47 99 47 119 105 110 110 116 47 115 121 115 116 101 109 51 50 47 51 109 100 46 101 120 101 63 47 99 43 100 105 114 32 71 84 84 80 47 49 46 48 32 52 48 52 32 50 57 54 72 71 55 117 118 110 116 101 53 98 72 111 70 81 77 101 43 110 48 85 113 97 56 88 100 89 89 61										
encrypted decimal message	59 184 84 41 59 164 78 41 59 108 108 41 59 108 108 76 59 184 38 161 118 107 38 84 108 108 68 148 108 84 148 84 78 148 84 68 76 147 137 101 76 38 176 38 102 62 66 66 40 38 4 77 4 40 118 131 68 84 38 68 131 144 41 118 120 118 160 38 176 87 144 62 75 76 147 101 101 126 38 59 41 108 76 171 108 171 76 84 184 164 183 147 132 178 50 66 40 118 179 21 183 155 9 174 121 118 87 66 108 17 71 58 78 11 144 166 166 73										
data matrix 3×40	59 184 84 41 59 164 78 41 59 108 108 41 59 108 108 76 59 184 38 161 118 107 38 84 108 108 68 148 108 84 148 84 78 148 84 68 76 147 137 101 76 38 176 38 102 62 66 66 40 38 4 77 4 40 118 131 68 84 38 68 131 144 41 118 120 118 160 38 176 87 144 62 75 76 147 101 101 126 38 59 41 108 76 171 108 171 76 84 184 164 183 147 132 178 50 66 40 118 179 21 183 155 9 174 121 118 87 66 108 17 71 58 78 11 144 166 166 73 0 0										
$Ceil_1$	6	4	3	781	1580	1436	911	1086	1745	960	762
	1066	1292	1213	995	766	1342	1270	1178	746	1794	917
	1301	1781	1683	419	1498	1491	1474	1309	1238	1676	903
	1677	926	1002	1225	1524	1310	1358	1605	974	842	
$Ceil_2$	20	15	12	2812	5546	5232	3442	4006	6262	3462	2818
	3988	4698	4416	3739	2824	4896	4530	4277	2680	6356	3478
	4492	6521	6160	1483	5538	5412	5346	4804	4322	6096	3189
	5972	3306	3621	4232	5613	4867	5027	5706	3310	2905	
$Ceil_3$	15	12	10	2207	4296	4132	2781	3189	4914	2722	2247
	3205	3716	3498	3009	2253	3880	3536	3372	2101	4948	2816
	3441	5172	4883	1152	4416	4270	4216	3810	3336	4812	2474
	4658	2584	2850	3242	4464	3892	4012	4447	2511	2223	
$Ceil_4$	42	35	30	6368	12298	11968	8182	9288	14188	7866	6552
	9398	10786	10166	8827	6578	11276	10166	9757	6058	14208	8296
	9772	15031	14184	3301	12878	12366	12206	11066	9526	13936	7083
	13386	7438	8241	9206	12993	11371	11707	12774	7084	6307	

$Ceil_5$	28	24	21	4337	8332	8172	5651	6368	9671	5364	4496
	6476	7380	6963	6083	4520	7722	6906	6658	4124	9646	5735
	6581	10291	9707	2237	8838	8445	8334	7571	6442	9516	4797
	9091	5058	5622	6199	8904	7814	8038	8673	4748	4244	
new data matrix 3×40	59 184 84 41 59 164 78 41 59 108 108 41 59 108 108 76 59 184 38 161 118 107 38 84 108 108 68 148 108 84 148 84 78 148 84 68 76 147 137 101 76 38 176 38 102 62 66 66 40 38 4 77 4 40 118 131 68 84 38 68 131 144 41 118 120 118 160 38 176 87 144 62 75 76 147 101 101 126 38 59 41 108 76 171 108 171 76 84 184 164 183 147 132 178 50 66 40 118 179 21 183 155 9 174 121 118 87 66 108 17 71 58 78 11 144 166 166 73 0 0										
decrypted decimal message	49 57 50 46 49 54 56 46 49 48 48 46 49 48 48 32 49 57 47 83 101 112 47 50 48 48 51 58 48 50 58 50 56 58 50 51 32 71 69 84 32 47 99 47 119 105 110 110 116 47 115 121 115 116 101 109 51 50 47 51 109 100 46 101 120 101 63 47 99 43 100 105 114 32 71 84 84 80 47 49 46 48 32 52 48 52 32 50 57 54 72 71 55 117 118 110 116 101 53 98 72 111 70 81 77 101 43 110 48 85 113 97 56 88 100 89 89 61										
new message	192.168.100.100 19/Sep/2003:02:28:23 GET /c/winnt/system32/cmd.exe?/c+dir HTTP/1.0 404 296 HG7uvnte5bHoFQMe+n0Uqa8XdYY=										

Fig. 5 An experiment for our experiment. We use the SHA-1 hash algorithm to hash the original message for future verification and use the RSA algorithm with the public-key (187,23) to encrypt the original message and hash digest, and with the privacy-key (187,7) to decrypt. By using the IDA, we disperse the message to 5 ceils. All of the three ceils can be restored back the original message.An original message is a illegal access fictitious log of IIS, decimal message is the decimal code of the original message and SHA-1 hash digest that are presented by ASCII code; encrypted decimal message is the RSA encryption data of decimal message by the public-key, (187,23); data matrix is the matrix of decimal message and we pad 0's to it if the length of encrypted decimal message is less than data matrix; $Ceil_1$ to $Ceil_5$ are the data after dispersing with dispersed vectors, the data in the shadow means the vectors of IDA, and others mean the data after dispersal; the new data matrix is the restoration data from $Ceil_1$, $Ceil_3$ and $Ceil_5$ (inverse(vectors matrix) · data matrix); decrypted decimal message is the RSA decryption data of decimal message by the privacy-key, (187,7); new message is the ASCII code of the new data matrix.