Computer Science and Information Systems 11(3):905–924 DOI: 10.2298/CSIS130921056H

# A True Random-Number Encryption Method Employing Block Cipher and PRNG

Yi-Li Huang, Fang-Yie Leu, Jian-Hong Chen, William Cheng-Chung Chu

Department of Computer Science, Tunghai University,
No. 1727, Section 4, Taiwan Boulevard, Taichung City, Taiwan
{yifung, leufy, g01350027, cchu}@thu.edu.tw

**Abstract.** In January 1999, distributed.net collaborated with the Electronic Frontier Foundation to break a DES (i.e., Data Encryption Standard) key, spending 22 hours and 15 minutes, and implying that the DES is no longer a secure encryption method. In this paper, we propose a more secure one, called the True Random Number Encryption Method (TRNEM for short), which employs current time, true random numbers and system security codes as parameters of the encryption process to increase the security level of a system. The same plaintext file encrypted by the TRNEM at different time points generates different ciphertext files. So these files are difficult to be cracked. We also analyze the security of the DES, AES (i.e., Advanced Encryption Standard) and TRNEM, and explain why the TRNEM can effectively defend some specific attacks, and why it is safer than the DES and AES.

**Keywords:** DES, AES, true random number, SSC, block cipher, wrapped ciphertext file

## 1. Introduction

Due to the popularity of computer systems and network services, the Internet-access security and information security have been a part of the focuses of computer research since when accessing the Internet, users may anytime anywhere face different kinds of attacks [1]. Thus, protecting important data stored in a computer or a cloud system and messages delivered in a network system is a challenge. Data Encryption Standard (DES) [2] and Advanced Encryption Standard (AES) [3,4] were then developed. However the DES has been cracked and the AES may someday be solved, e.g., by differential attack [5] or linear attack [6]. On the other hand, security data is often encrypted by random numbers which play a critical role in information security services, e.g., when employing an one-way hash function [7] to generate message digests, encrypting messages [8], and signing an electronic document with a digital signature [9,10]. Unfortunately, true random numbers are difficult to obtain since it is hard for us to design themin a deterministic way. However, human activities and the information having been collected in a website as well as their description own the characteristics similar to those of a true random number since before reading them, we do not know what has been collected and how they are described. These data often continuously and randomly vary at different time. In fact, we can randomly select a short fragment of the data as true random numbers from a randomly chosen website and use the segment to

encrypt plaintext. In this study, we develop a data protection mechanism, named True Random Numbers Encryption Method (TRNEM for short), which encrypting plaintext by employing true random numbers is a secure encryption approach which is difficult to be cracked by using brute force attacks and ciphertext analyses.

The rest of this paper is organized as follows. Section 2 briefly describes the related studies of this paper, including AES, and DES, and their vulnerabilities. Section 3 introduces the encryption/decryption process of the TRNEM. The security and performance of the TRNEM and the comparison between the TRNEM and the AES are presented in Section 4. Section 5 concludes this paper and outlines our future studies.

## 2. Common Block Cipher

Currently, the most common block cipher modes are the DES and AES.

### 2.1. Data Encryption Standard (DES)

DES [2] is a symmetric block cipher algorithm in which the encryption and decryption details are almost the same. The length of a key is 56 bits (the key is typically expressed as a 64-bit number, but the first eight bits are used for parity check). The DES encrypts a 64-bit plaintext block into a 64-bit ciphertext block. Its key generation process can be mainly divided into two steps, the initial permutation and the inverse permutation.

In the initial permutation step, the 64-bit input block is permuted to generate two outputs L0 and R0, each of which is 32 bits long. After 16 times of iteration, L0 and R0, respectively, become L16 and R16, which are then input to the inverse permutation process to recover these bitsto their original sequence. The result is the corresponding ciphertext block. DES [11] is unsafe because a brute force attack may succeed. Currently, one of its threats is the linear cryptanalysis [12] which collected 243 known plaintexts. The cracking time complexity ranges between $2^{39}$ and $2^{43}$ [13]. But the complexity can be reduced to 1/4 [14] with the help of a chosen-plaintext attack.

Three effective DES attacks, include differential cryptanalysis [15], linear cryptanalysis [12] and Davies' attack [16], which can break the 16 rounds of DES with the time complexity lower than that of a brute-force method.

### 2.2. Advanced Encryption Standard (AES)

AES [17] algorithm was developed based on bit permutation and substitution. It re-arranges the sequence of the original data, and substitutes a data unit by another. As an iterative and symmetric-key block cipher technique with 128, 192, or 256 bits as its key length, AES encrypts a data block with 10 rounds of duplication and transformation. Each round comprises the SubBytes, ShiftRows, MixColumns and AddRoundKey steps, except the final round in which the MixColumns is substituted by an AddRoundKey. Generally, in the AddRoundKey step, each byte of the data is bitwise-xored with a round key.

In the SubBytes step, each byte is substituted by another one following the content of a predefined lookup table. The ShiftRows rotates a row of a state where a state is an AES calculation on a 4×4 column-major order matrix of bytes. The initial value of this matrix is a plaintext block. In the MixColumns step, a column-wise linear transformation is performed by multiplying a constant matrix and the state matrix to produce a new state matrix.

In 2009, the side-channel attack [18,19] successfully cracked an easy version of the AES. But the National Security Agency (NSA) reviewed all the AES finalists, and claimed that all of them were secure enough for U.S. Government non-classified data. But the weak version that has been successfully cracked and the number of encryption loop of this version are almost the same as those of original version. Cryptographers are worrying about the security of the AES. If the penetrating capabilities of some well-known attack are improved, this block encryption system may someday be cracked again.

### 2.3 Block Cipher Mode of Operation

An operation mode is mainly used to encrypt and authenticate delivered messages. An operational model defines the process of encrypting a data block, often based on a given initialization vector (IV for short) as an additional parameter to further enhance the security of the encrypted data.

If different IVs are given, the same plaintext will generate different ciphertext, even though the plaintext is encrypted by using the same key. The purpose is to avoid regenerating the same ciphertext.

The Cipher Block Chaining (CBC), the Propagating Cipher Block Chaining (PCBC), Cipher feedback (CFB), Output feedback (OFB) and Counter (CTR) are block cipher standards having been recognized by the National Institute of Standards and Technology (NIST). With the CBC mode, as shown in the following two statements, a plaintext block $P_i$ is XORed with the ciphertext $C_{i-1}$ generated in the previous encryption round. The XORed result and the encryption key K are then input to the Block-Cipher-Encryption function to produce the ciphertext $C_i$ where $C_0$ is the *IV* of the CBC mode.

$$C_1 = E_K(P_1 \oplus IV)$$

$$C_i = E_K(P_i \oplus C_{i-1}), 2 \le i \le \mathrm{n}$$

With the PCBC mode, as illustrated in the following two statements, a plaintext block $P_i$is XORed with $P_{i-1} \oplus C_{i-1}$. The XORed result and the encryption key K are then input to the Block-Cipher-Encryption function to produce the ciphertext $C_i$ where $P_0 \oplus C_0$ is the *IV* of the PCBC mode.

$$C_1 = E_K(P_1 \oplus IV)$$

$$C_i = E_K(P_i \oplus P_{i-1} \oplus C_{i-1}), 2 \le i \le \mathrm{n}$$

The following two statements show the encryption process of the CFB mode. The ciphertext generated in the previous round and the encryption K are input to the Block-Cipher-Encryption function. The result is then XORed with plaintext $P_i$ to yield the ciphertext $C_i$ where $C_0$ is the *IV* of the CFB mode.

$$C_1 = E_K(IV) \oplus P_1$$

$$C_i = E_K(C_{i-1}) \oplus P_i, 2 \le i \le n$$

In the OFB mode, $O_{i-1}$ and the encryption key $K$ are input to the Block-Cipher-Encryption function to produce $O_i$. $O_i$is then XORed with plaintext $P_i$ to produce the ciphertext $C_i$where $O_0$ is the *IV*.

$$C_0 = P_i \oplus E_K(IV)$$

$$C_i = P_i \oplus E_K(O_{i-1}), 2 \le i \le n$$

Similar to that of the OFB mode, the CTR mode ciphers a plaintext block with a stream-cipher method. It generates the next key-stream block by using a counter which is often a function of time with a very long repeating cycle. During encryption, the encryption key K and the counter are input to the Block-Cipher-Encryption function. The result is then XORed with plaintext $P_i$ to produce the ciphertext $C_i$. After an encryption round, the counter value is increased by one. The new value is used to encrypt the next plaintext block.

Although these modes provide a security system with data integrity and confidentiality, they are still vulnerable to known plaintext-ciphertext cryptanalysis attacks.

## 3. The Proposed Method

In this section, we first define the parameters and codes used by the TRNEM.

### 3.1. The Parameters

The parameters are as follows.

*File name*: which is the name of the file being encrypted. Its length is 16 characters. If originally the length is longer than 16, we keep the first 16 characters and truncate the remaining ones. However, if the length is shorter than 16, we extend it by duplicating the file name n times until the length is equal to or longer than 16, $n$>1, and then extract the first 16 characters.

*Filename_ext*: which isthe filename extension of the file. Its length is also 16 characters. If originally it is longer than 16, we extral the first 16 and truncate the remaining ones. If the length is shorter than 16, we extend it with the same method as that used to extend its file name. However, if the length is zero, we put 16 *s as the filename_ext.

*SSC*: which stands for system security code.*SSC* has 16 members where *SSC*(*i*) is the $i^{th}$

system security code, $1 \leqq i \leqq 16$, and the length of $SSC(i)$ is 128 bits.

$\triangle h$：which is a variable of 11 bits long for indicating the length of a pseudo random number sequence (PRNS), i.e., $1 \leqq \triangle h \leqq 2047$.

$K\triangle h$：which is an encryption key of 128 bits long. It is generated by the concatenation of 12 $\triangle hs$, but discarding the last 4 bits.

*KCT*: which is a current-time encryption key defined as a bit sequence obtained by concatenating the following items, including $\triangle h$, and current values of the system clock which contains nanosecond, second, minute, hour, and nanosecond of the clock, i.e., *KCT*=$\triangle h$||nanosecond||second||minute|| hour||nanosecond ||$\triangle h$ , where "||" denotes concatenation. $\triangle h$ consists of 4 digits, nanosecond is 9 digits long, each of the remaining items is 2 digits in length and each digit is 4 bits long, i.e., |*KCT*| =128 bits (= 4+9+2+2+2+9+4=32 digits).

*WI* (Web-Index): We randomly select an URL as the *WI* from those dynamically crawled webpages (named crawled files), $1 \leqq WI \leqq 1023$.

*Sd* (Start-distance): which isthe start point of the encrypting segment extracted from the $WI^{th}$ crawled file. The start point is the $Sd^{th}$ character of the file, $1 \leqq Sd \leqq 1023$.

*TRNS*: which stands for True Random Number Sequence (TRNS). It is the segment extracted from the $Sd^{th}$ character of the $WI^{th}$ web's content.

$\triangle L$: which is the length of TRNS, $1024 \leqq \triangle L \leqq 2047$.

RIG*y*(X): which is the value of the y right-most bits of the key *X*, i.e., if *X*=*x*[1] *x*[2]… *x*[|*X*|], RIGy(*X*)= *x*[|*X*|-(*y*-1)]~*x*[|*X*|], where *x*[*i*] is the $i^{th}$ bit of *X*, *i*=1,2,…|*X*|, and *y*=8, 128 or 256, e.g., when *y*=256, RIG256(*X*)=*x*[|*X*|-255] ~*x*[|*X*|], and when *y*=8, RIG8(*X*)=*x*[|*X*|-7]~*x*[|*X*|]. If *X* is a character string, we treat it as a long bit string by sequentially substituting these characters by their ASCII codes, e.g., if *X*=abc, 616263 will be the corresponding bit string of 24 bits long.

LEF*y*(*X*): which is the value of the *y* left-most bits of *X*, LEF*y*(*X*)=*x*[1]~*x*[*y*]. For example, when *y*=128, LEF128(*X*)=*x*[1]~*x*[128], and when *y*=8, LEF8(*X*)= *x*[1]~*x*[8].

### 3.2. The Equations used to Generate Encryption Keys

The equations employed in this study are as follows.

$$\triangle h = [(\sum_{i=1}^{4} \text{RIG20}(SSC(i)) + \text{RIG20}(\textit{file name})) * (\sum_{i=5}^{8} \text{RIG20}(SSC(i)) + \text{RIG20}(\textit{filename\_ext})) + (\sum_{i=9}^{12} \text{RIG20}(SSC(i)) + \text{LEF20}(\textit{file name})) * (\sum_{i=13}^{16} \text{RIG20}(SSC(i)) + \text{LEF20}(\textit{filename\_ext}))] \bmod 2047 + 1 \quad (1)$$

which randomly varies each time when it is invoked. It is the first parameter adopted by the TRNEM.

$$DA = \text{HMAC}(SSC(1) \oplus KCT \| SSC(2) \oplus KCT \| SSC(3) +_2 K \triangle h \| SSC(4) +_2 K \triangle h, SSC(5) \oplus KCT) \quad (2)$$

which randomly varies each time when it is invoked. It is the first dynamic key employed by the TRNEM.

$$DB = \text{HMAC}(SSC(6) \oplus DA \| SSC(7) \oplus DA \| SSC(8) +_2 KCT \| SSC(9) +_2 KCT, DA +_2 K \triangle h) \quad (3)$$

which randomly varieseach time when it is generated. It is the second dynamic key employed by TRNEM.Eqs. (2) and (3), that respectively generate dynamic keys *DA* and *DB*, together are called Equation-group 1.

$$CDA = [((SSC(10) \oplus DA) +_2 SSC(11)) +_2 (K \triangle h \oplus SSC(12))] \oplus (SSC(13) +_2 K \triangle h) \quad (4)$$

$$CDB = [((SSC(14) \oplus DB) +_2 SSC(15)) +_2 (DA \oplus K \triangle h)] \oplus (SSC(16) +_2 DA) \quad (5)$$

Eqs. (4) and (5), that respectively produce the encrypted dynamic keys *CDA* and *CDB*, together are called Equation-group 2.

$$\triangle L = [\text{LEF12}(SSC(2)) * \text{RIG12}(DA) + \text{LEF12}(SSC(3)) * \text{RIG12}(DB) + (\text{LEF12}(K \triangle h) + \text{LEF12}(DA) + \text{LEF12}(DB)) * \text{LEF12}(SSC(4))] \bmod 1024 + 1024 \quad (6)$$

$$WI = [\text{LEF12}(SSC(5)) * \text{LEF12}(DA) + \text{LEF12}(SSC(6)) * \text{LEF12}(DB) + (\text{LEF12}(K \triangle h) + \text{RIG12}(DA) + \text{RIG12}(DB)) * \text{LEF12}(SSC(7))] \bmod 1023 + 1 \quad (7)$$

$$Sd = [\text{LEF12}(SSC(8)) * \text{LEF12}(DA) + \text{LEF12}(SSC(9)) * \text{LEF12}(DB) + (\text{RIG12}(K \triangle h)^2 + \text{RIG12}(DA)^2 + \text{RIG12}(DB)^2) * \text{LEF12}(SSC(10))] \bmod 1023 + 1 \quad (8)$$

$$Pk_1 = \text{HMAC}(SSC(11) +_2 DA \| SSC(12) +_2 DA \| SSC(13) \oplus DB \| SSC(14) \oplus DB, (SSC(15) +_2 DB) \oplus DA) \quad (9)$$

which as a pseudorandom key is the first pointing key employed by the TRNEM to generate the *PRNS*1, *PRNS*2 and *CTRNS*. Eqs. (6) ~ (9), that respectively generate $\triangle L$, *WI*, *Sd* and $Pk_1$, together are called Equation-group 3.

$E(k, str)$: An encryption function defined as:

$$E(k, str) = k \oplus s_1 \| k \oplus s_2 \| k \oplus s_3 \| \ldots \| k \oplus s_n, \quad (10)$$

where $str = s_1 s_2 s_3 \ldots s_n$ is a string.

$$TRNS(j) = \text{HMAC}(\text{E}(SSC(j), TRNS) \| E(SSC(17\text{-}j), TRNS), SSC(j+7) +_2 DB), 1 \leqq j \leqq 4 \quad (11)$$

$$\triangle t = (\text{RIG12}(DA)^3 + \text{RIG12}(DB)^3 + \text{LEF12}(DA)^3 + \text{LEF12}(DB)^3 + \text{RIG12}(TRNS(1))^3 + \text{LEF12}(TRNS(1))^3) \bmod 1023 + 1 \quad (12)$$

which as a pesudorandom parameter is the length of $PRNS2$. $\triangle t$ together with $\triangle h$ are adopted to protect the CTRNS and ciphertext in the wrapped ciphertext file.

$$Pk_2 = \text{HMAC}(TRNS(2) \oplus DA \,\|\, TRNS(3) \oplus DB \,\|\, TRNS(4) +_2 DA, TRNS(1) \oplus DB) \quad (13)$$

which as a pseudorandom key is the second pointing key employed by the TRNEM to generate the ciphertext.

Eqs. (10) ~ (13), that respectively produce $E(k, str)$, $TRNS(1)$ ~ $TRNS(4)$, $\triangle t$ and $Pk_2$, together are called Equation-group 4.

### 3.3. The TRNEM Encryption Process

Fig. 1 illustratively summarizes the encryption flow of the TRNEM. The details are as follows.

**Step 1**: Generating $\triangle h$ and $K\triangle h$. The TRNEM's encryption process invokes the non-invertible $\triangle h$ generation equation defined above to read the file name of the file being encrypted. The file name, filename extension and *SSCs* are the parameters used to produce $\triangle h$ and $K\triangle h$.

**Step 2**: Generating dynamic keys *DA* and *DB*. The TRNEM derives *KCT* from $\triangle h$ and current time (*CT*), and invokes Equation-group 1 which uses $K\triangle h$, *KCT* and *SSCs* as its parameters to produce dynamic keys *DA* and *DB*.

**Step 3**: Encrypting dynamic keys. The TRNEM invokes Equation-group 2 which consisting of two invertible equations defined above employs the generated *DA*, *DB*, *SSCs* and $K\triangle h$ as the parameters to produce *CDA* and *CDB* so that the TRNEM can securely store *CDA* and *CDB* into the wrapped ciphertext file and decrypt *DA* and *DB* from *CDA* and *CDB* carried in the received wrapped ciphertext file.

**Step 4**: Generating $\triangle L$, *WI*, *Sd* and $Pk_1$. The TRNEM invokes Equation-group 3, consisting of four non-invertible equations defined above, to respectively produce $\triangle L$, *WI*, *Sd* and $Pk_1$ by employing the generated *DA*, *DB*, *SSCs* and $K\triangle h$ as input parameters.

**Step 5**: Generating $TRNS(1)$ ~ $TRNS(4)$, $\triangle t$, and $Pk_2$. The TRNEM randomly reads data of $\triangle L$ bytes from the chosen webpage indexed by *WI* and the first character is the $Sd^{th}$ byte of the webpage. These data are our *TRNS*. The TRNEM invokes Equation-group 4 which consisting of some non-invertible equations defined above in turn invokes the generation equations of the *DA*, *DB*, *SSCs* and *TRNS* to produce $TRNS(1)$ ~ $TRNS(4)$, $\triangle t$ and $Pk_2$.

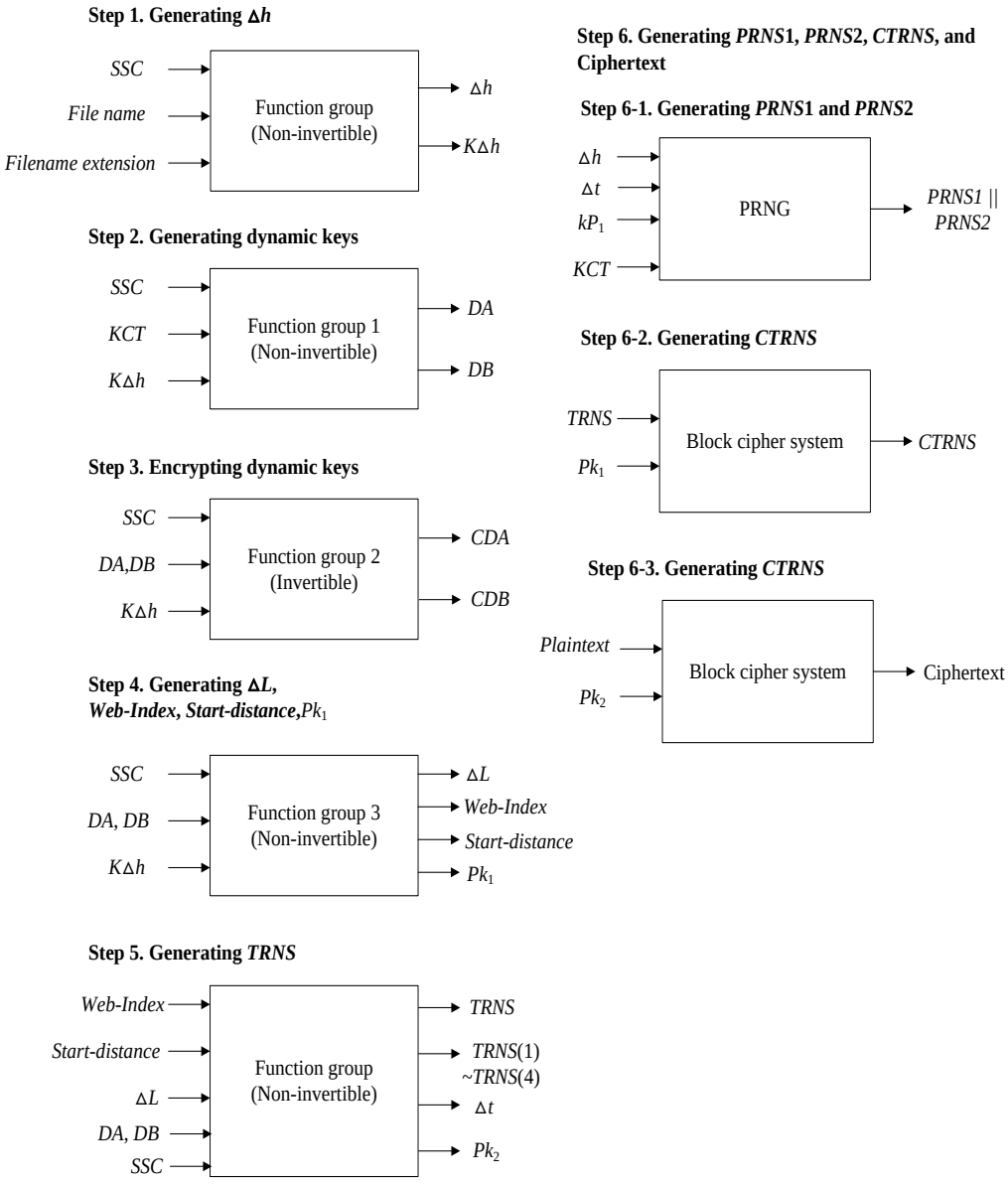


**Fig. 1.** The encryption flow of the TRNEM

**Step 6**: Generating *PRNS*1, *PRNS*2, *CTRNS* and ciphertext.

**Step 6-1**: Generating *PRNS*1 and *PRNS*2. The TRNEM grabs the time parameters from system clock to produce a new *KCT*. After that, *KCT*, $\triangle h$, $\triangle t$ and $Pk_1$are input to the pseudo random number generator (PRNG for short) to produce *PRNS*1 and *PRNS*2.

**Step 6-2**: Generating *CTRNS*. The *CTRNS* is produced by the adopted block cipher system (e.g., AES) with the TRNS as the plaintext and key $Pk_1$ as an input parameter.

**Step 6-3**: Encrypting plaintext (generating ciphertext). A plaintext to be encrypted and key $Pk_2$ are input to the adopted block cipher system to produce the corresponding ciphertext.

**Step 7**: Generating a wrapped ciphertext file. The TRNEM concatenates *PRNS*1, *CDA*, *CDB*, *CTRNS*, the ciphertext generated in Step 6 and *PRNS*2 to produce a wrapped ciphertext file, the format of which is shown in Fig. 2.

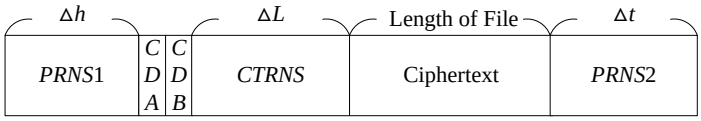


**Fig. 2.** The format of the wrapped ciphertext file generated by the TRNEM

### 3.4. The TRNEM Decryption Process

Fig. 3 illustrates the decryption process of the TRNEM. The details are described below.

**Step 1**: Calculating $\triangle h$ and removing *PRNS*1 from the received wrapped ciphertext file. To decrypt the ciphertext, a user needs to invoke the $\triangle h$ generation equation, which in turn reads the file name and filename extension of the designated file to produce $\triangle h$, with which to delete *PRNS*1 from the wrapped ciphertext file. It further generates $K\triangle h$.

**Step 2**: Retrieving and calculating *DA* and *DB*. Reads *CDA* and *CDB* from the wrapped ciphertext file and decrypts them by using the following two decryption equations, i.e., Eqs. (14) and (15), to obtain the dynamic keys *DA* and *DB*.

$$DA = [CDA \oplus (SSC(13) +_2 K\triangle h)] -_2 (K\triangle h \oplus SSC(12)) -_2 SSC(11) \oplus SSC(10) \quad (14)$$

where $-_2$ is the inverse operation of $+_2$ [20].

$$DB = [CDB \oplus (SSC(16) +_2 DA)] -_2 (DA \oplus K\triangle h)) -_2 SSC(15) \oplus SSC(14) \quad (15)$$

**Step 3**: Calculating $\triangle L$ and retrieving *CTRNS*

(1) Invoking Eq. (6) which employs *SSCs*, $K\triangle h$, *DA* and *DB* as its parameters to calculate $\triangle L$.

(2) Retrieving *CTRNS* from the wrapped ciphertext file based on the calculated $\triangle L$ since *CTRNS* is $\triangle L$ in length.

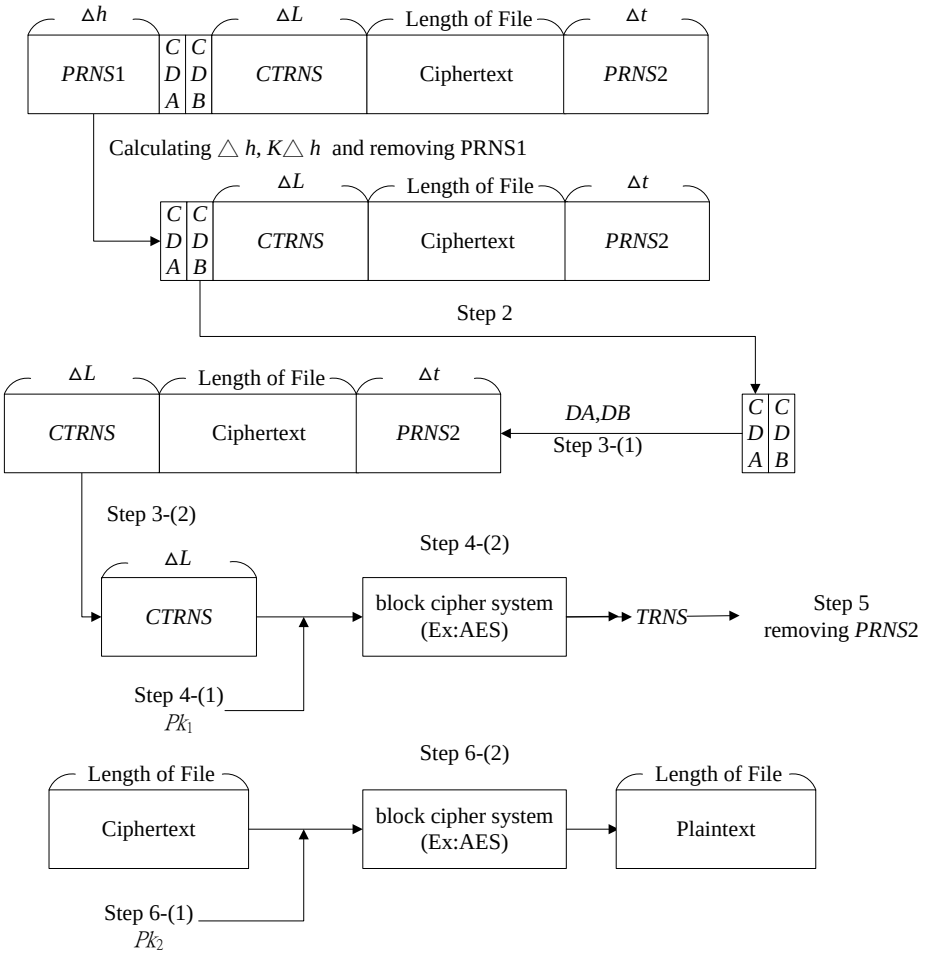


**Fig. 3.** The decryption flow of the TRNEM

**Step 4**: Retrieving *TRNS*. Retrieve *TRNS* by inputting *CTRNS* and $Pk_1$ to the adopted block cipher system.

(1) Producing $Pk_1$ by invoking Eq. (9) which employs*DA*, *DB* and *SSCs* as its parameters.
(2) Invoking the adopted block cipher system to decrypt the *CTRNS* retrieved from the wrapped ciphertext file with $Pk_1$so as to produce *TRNS*.

**Step 5**: Retrieving $\triangle t$ and removing *PRNS*2.

(1) Producing *TRNS*(1) ~ *TRNS*(4) by invoking Eqs. (10) and (11) which employ *SSCs* and *TRNS* as their parameters.
(2) Producing $\triangle t$ by invoking Eq. (12) which utilizes *DA*, *DB* and *TRNS*(1) as its parameters.
(3) Removing *PRNS*2 from the wrapped ciphertext file.

**Step 6**: Generating $Pk_2$ and decrypting the ciphertext.

(1) Invoking Eq. (13) which uses *TRNS*(1)~*TRNS*(4), *DA* and *DB* as its parameters to produce $Pk_2$.
(2) Decrypting the plaintext from the ciphertext by inputting $Pk_2$ and the ciphertext to the adopted block cipher system so as to revert the plaintext.

### 3.5. The Features and Advantanges of the TRNEM

The TRNEM has five features, including

(1) employing filename, filename extension, system security codes and a non-invertible equation to generate $\triangle h$, making $\triangle h$ be one with high security;
(2) utilizing current time to produce dynamic keys *DA* and *DB* which are different when they are generated at different time points since current time continuously varies;
(3) using the *DA* and *DB* to fetch the true random numbers, with which to encrypt the plaintext so as to enhance the security of the ciphertyext;
(4) employing scalable parameters $\triangle h$, $\triangle L$ and $\triangle t$, with which to construct a wrapped ciphertext file. The purpose is enhancing the security of the ciphertext file;
(5) the *CDA*, *CDB* and CTRNS are embedded in the wrapped ciphertext file to effevtively protect the ciphertext.

Beside the mentioned security features, based on the Interent as its data pool, the TRNEM creates a true random number sequence to encrypt plaintext so that the ciphertext has a very high degree of security. Furthermore, the ciphertext is embedded in the position located between *PRNS*1 and *PRNS*2. Hackers cannot directly obtain the (plaintext, ciphertext) pairs from the wrapped ciphertext file, thus highly enhancing the security of the TRNEM.

## 4. Security and Performance Analysis

In this section, we analyze the security levels of different TRNEM parameters and generated data, including $\triangle h$, dynamic keys *DA* and *DB*, the *TRNS*, a wrapped ciphertext file, and $Pk_2$. We also evaluate the security and performance of the TRNEM.

### 4.1. Security of $\Delta$h

There are three major reasons to say that $\triangle h$ possess high security. According to Eq. (1), without the 16 system security codes *SSC*(1) – *SSC*(16), hackers cannot correctly calculate $\triangle h$, even though they have caught the file name and filename extension. Also, the value generated for each term contained in Eq. (1) is larger than $2^{20}$ which is very larger than the upper limit of $\triangle h$, i.e., 2047. After that, the value generated before the modulus operation is reduced to $\triangle h$ through the non-invertible modulus equation,

implying that $\triangle h$ has high randomness and security. Furthermore, the $\triangle h$ is an internal variable of the TRNEM. Hackers cannot derive it from the ciphertext and solve it.

Authough, hackers can try a lot of file names and filename extensions to respectively substitute for the original file name and filename extension contained in the encryption expression, attempting to analyze the possible $\triangle h$. However, the length of the wrapped ciphertext file is $\triangle h$+32+$\triangle L$+|the plaintext| + $\triangle t$ bytes, in which 32 is the length of *DA*+*DB*, and $\triangle L$ and $\triangle t$ randomly change at each encryption, even the file name, filename extension and plaintext remain unchanged. $\triangle h$ is well protected due to the dynamic values of $\triangle L$ and $\triangle t$. Now, we dare to say that $\triangle h$ and the protected system are very safe.

### 4.2. Security of Dynamic Keys DA and DB

There are two methods to obtain *DA*. First, hackers may directly generate *DA* by employing Eq. (2). However, *SSC*(1) ~ *SSC*(5), $K\triangle h$ and *KCT* are unknown to hackers and *KCT* continuously changes at each encryption. That means hackers cannot directly generate *DA* by employing Eq. (2). Second, hackers may crack *CDA* to obtain *DA*. However, according to Eq. (4), they need *SSC*(10)~*SSC*(13) and $K\triangle h$. But these parameters are unknown to hackers. Furthermore, *CDA* is embedded in the wrapped ciphertext file. Hackers need $\triangle h$ to correctly fetch it. But $\triangle h$ is unknown to hackers. Thus, *DA* is secure. Similarly, to derive *DB* from *CDB*, hackers need *SSC*(14) ~ *SSC*(16), $K\triangle h$ and *DA* which are unknown to hackers, i.e., *DB* is secure.

### 4.3. Security of the TRNS

The TRNEM collects a webpage based on a randomly chosen *WI*, and accesses the content of the webpage from the position indicated by the Sd to the position pointed to by *Sd* +$\triangle L$. In other words, |*TRNS*|=$\triangle L$. But the characteristics and contents of different pages vary with time, and the page contents may be changed frequently. Under this circumstance, extracting web contents from a randomly chosen webpage can make a number sequence, i.e., the *TRNS*, truly random.

Hackers may obtain *TRNS* by decrypting *CTRNS* embedded in the wrapped ciphertext file. However, to fetch the *CTRNS*, parameters $\triangle h$, $\triangle L$, length of the plaintext and $\triangle t$ are required. But, hackers cannot obtain them from the wrapped ciphertext file. That is, hackers cannot correctly fetch *CTRNS* from this wrapped ciphertext file. Furthermore, if CTRNS is known by hackers, they still cannot decrypt *CTRNS* to obtain *TRNS* since $PK_1$ is unknown to them. So, *TRNS* is secure.

### 4.4. Security of a Wrapped Ciphertext File

This system adopts a wrapping ciphertext approach, in which the ciphertext as shown in Fig. 2 is wrapped by *PRNS*1 of length $\triangle h$, *CTRNS* of length $\triangle L$, and *PRNS*2 of length

$\triangle t$. Parameters $\triangle L$ and $\triangle t$ are different at each encryption even though the plaintext is the same. Hackers cannot obtain $\triangle h$, $\triangle L$ and $\triangle t$ to unwrap the ciphertext, i.e., hackers cannot collect (plaintext, ciphertext) pairs when plaintext is known. They need to crack $\triangle h$ before solving other parameters, meaning that the ciphertext file is securely protected by the TRNEM.

### 4.5. Security of the $Pk_2$

$Pk_2$ as a pseudorandom key is the second pointing key employed by the TRNEM to generate the ciphertext. The security of the ciphertext strongly depends on the security of $Pk_2$ and the block cipher system. In the following, we would like to identify the security level of $Pk_2$. Theorem 1 proves that its security level is the same as that when it is solved by using a blind guess method.

**Theorem 1.** If the key length of the TRNEM is n bits, then the probability of solving the correct value of $Pk_2$ from the wrapped ciphertext file is $1/2^n$.

*Proof.* $Pk_2$ as an internal pseudorandom key used by the TRNEM does not appear in the wrapped ciphertext file. Hackers cannot directly break it. Two methodscan be used to break $Pk_2$, excluding the blind guess approach. The first is that,according to Eq. (13), i.e., $Pk_2$ = HMAC($TRNS(2)\oplus DA \parallel TRNS(3)\oplus DB \parallel TRNS(4) +_2 DA$, $TRNS(1)\oplus DB$), only the one who knows*DA*, *DB*, $TRNS(1) \sim TRNS(3)$ can correctly generate $Pk_2$. However, the dynamic keys *DA* and *DB* are secure and the true random number sequences, i.e., $TRNS(1) \sim TRNS(3)$, derived from*TRNS* and *DB* are secure, too, based on the abovementioned description. The dynamic keys *DA* and *DB*, which are functions of *KCT*, vary randomly each time when it is generated, implying that the generated messages, *TRNS* and hence, $TRNS(1) \sim TRNS(3)$ change randomly each time when they are produced so that $Pk_2$is secure.

The second method is breaking the block cipher system to obtain plaintext from the ciphertext. However, the ciphertext embedded in the position located between *PRNS*1 and *PRNS*2 is secure, according to that described in section 4.4. That is, hackers cannot break $Pk_2$ from the ciphertext. In the worst case, if the ciphertext is known by the hackers, they need to break the block cipher system. But this is not an easy work since hackers require a massive amount of (plaintext, ciphertext) pairs given the same parent key. Hence they still cannot break the block cipher system to obtain $Pk_2$.

There are no useful method to obtain $Pk_2$ other than the blind guess approach. Therefore, the probability of solving the correct value of$Pk_2$ from the wrapped ciphertext is $1/2^n$. Q.E.D.

### 4.6. Security of the TRNEM

Due to involving current time and TRNS, the encryption results generated on the same plaintext at different time points vary, implying that TRNEM can effectively prevent those linear cryptanalysis attacks [21,22]. In the TRNEM, the mechanism that wraps a

ciphertext file can effectively defend the known plaintext attacks because hackers cannot correctly collect different (plaintext, ciphertext) pairs.

In fact, the TRNEM integrates time variables, i.e., the current time and TRNS. So the wrapped-ciphertext-file mechanism can effectively prevent the protected system from brute force attacks.

### 4.7. Generation Times of Parameters

The TRNEM generates ciphertext by using a block cipher system (e.g., AES or DES). To generate a wrapped ciphertext file, we produce several parameters introduced above. Table 1 lists the times required to produce these parameters. We also used these parameters to produce the *PRNS*1, *PRNS*2, *CDA*, *CDB* and *CTRNS*. Table 2 lists the times required to produce them and the wrapped ciphertext file.

**Table 1.** The times required to produce different required parameters

| Parameter | Parameter generation time (ms) |
|---|---|
| Eq. (1):$\triangle h$ | 0.01948 |
| Eq. (2):*DA* | 0.30646 |
| Eq. (3):*DB* | 0.27196 |
| Eq. (4):*CDA* | 0.23230 |
| Eq. (5):*CDB* | 0.23624 |
| Eq. (6):$\triangle L$ | 0.00963 |
| Eq. (7):*WI* | 0.00958 |
| Eq. (8):*Sd* | 0.01020 |
| Eq. (9):$Pk_1$ | 0.42009 |
| Eq. (11):*TRNS*(1) | 15.06665 |
| Eq. (11):*TRNS*(2) | 14.96699 |
| Eq. (11):*TRNS*(3) | 15.41611 |
| Eq. (11):*TRNS*(4) | 15.52798 |
| Eq. (12):$\triangle t$ | 0.00820 |
| Eq. (13):$Pk_2$ | 0.28322 |
| Total | 62.78509 |

**Table 2.** The time required to produce the wrapped ciphertext file

| Item | Generation time (ms) |
|---|---|
| *PRNS*1\|\|*PRNS*2(length of$\triangle h+\triangle t$) | 20 |
| *CDA* | 0.2323 |
| *CDB* | 0.23624 |
| *CTRNS*(length of$\triangle L$) | 13 |
| Ciphertext | The same as the time required by AES or DES |

No matter what size of the file to be encrypted is, the times the TRNEM spent to generate *PRNS*1, *PRNS*2, *CDA*, *CDB* and *CTRNS* are themselves the same. Compared with other block cipher techniques, it only takes a very short extra time to encrypt a file. But the security level on the contrary dramatically increases.

Since the lengths of *PRNS*1||*PRNS*2 (i.e., $\triangle h+\triangle t$) and *CTRNS* (i.e., $\triangle L$) are not fixed, we individually chose the max lengths of them to calculate their generation times. Ciphertext is encrypted by block ciphering. Its generation time is the same as those of the adopted block cipher system, e.g., AES and DES. The extra time required by the TRNEM is 95.78529 ms.

The difference between the decryption process and the encryption process of the TRNEM is that when decrypting the ciphertext file, *DA* and *DB* are acquired by invoking Eqs. (4) and (5), which further invoke the invertible equations to generate *CDA* and *CDB* where *CDA* and *CDB* are retrieved from the wrapped ciphertext file. *TRNS* is decrypted by inputting *CTRNS* and $\mathrm{Pk}_1$ to the adopted block cipher system where *CTRNS* is also retrieved from the wrapped file. Since the formulas used to generate other parameters for decryption are the same as those when encrypting the plaintext file, the times required to produce $\triangle h$, $\triangle L$, $Pk_1$, *TRNS*(1), *TRNS*(2), *TRNS*(3), *TRNS*(4), $\triangle t$ and $Pk_2$ are then individually the same as those when encrypting the file. Table 3 lists the times required to produce parameters for decrypting *CTRNS*, and Table 4 shows the ciphertext decryption time.

**Table 3.** The times required to produce different parameters for decrypting the wrapped ciphertext file

| Parameter | Parameter generation time (ms) |
|---|---|
| Eq. (1):$\triangle h$ | 0.01948 |
| Eq. (4):*CDA* | Reads *CDA* from the wrapped file |
| Eq. (5):*CDB* | Reads *CDB* from the wrapped file |
| Eq. (14):*DA* | 0.19569 |
| Eq. (15):*DB* | 0.19926 |
| Eq. (6):$\triangle L$ | 0.00963 |
| Eq. (9):$Pk_1$ | 0.42009 |
| Eq. (11):*TRNS*(1) | 15.06665 |
| Eq. (11):*TRNS*(2) | 14.96699 |
| Eq. (11):*TRNS*(3) | 15.41611 |
| Eq. (11):*TRNS*(4) | 15.52798 |
| Eq. (12):$\triangle t$ | 0.0082 |
| Eq. (13):$Pk_2$ | 0.28322 |
| Total | 62.1133 |

**Table 4.** The times required to decrypt the *CTRNS* and the wrapped ciphertext file

| Item | Generation time (ms) |
|---|---|
| *TRNS* | 13 |
| plaintext | The same as the time required by AES or DES |

Table 5 lists the computational efforts in terms of different numbers of operations employed by the encryption/ decryption processes of the TRNEM.

**Table 5.** All computational efforts in terms of different numbers of operations employed by the encryption/ decryption processes of the TRNEM

| TRNEM | Encryption | Decryption |
|---|---|---|
| Eq. (1):$\triangle h$ | 18+s + 2*s + 1 mod | 18+s + 2*s + 1 mod |
| Eq. (2):*DA* | 3$\oplus$s (128 bits) + 2+$_2$s (128 bits) + 1HMAC | does not generate |
| Eq. (3):*DB* | 2$\oplus$s (128 bits) + 3+$_2$s(128 bits) + 1HMAC | does not generate |
| Eq. (4):*CDA* | 3$\oplus$s (128 bits) + 3+$_2$s(128 bits) | does not generate |
| Eq. (5):*CDB* | 3$\oplus$s (128 bits) + 3+$_2$s(128 bits) | does not generate |
| Eq. (6):$\triangle L$ | 3*s + 4+s + 1 mod | 3*s + 4+s + 1 mod |
| Eq. (7):*WI* | 3*s + 4+s + 1 mod | does not generate |
| Eq. (8):*Sd* | 6*s + 4+s + 1 mod | does not generate |
| Eq. (9): $Pk_1$ | 3$\oplus$s (128 bits) + 3+$_{2s}$(128 bits) + 1HMAC | 3 $\oplus$ s (128 bits) + 3+$_2$s(128 bits) + 1HMAC |
| Eq. (11): *TRNS*(1) | 2$E$(*k*,*str*) + 1+$_{2s}$ (128bits) + 1HMAC | 2$E$(*k*,*str*) + 1+$_2$s (128bits) + 1HMAC |
| Eq. (11): *TRNS*(2) | 2$E$(*k*,*str*) + 1+$_{2s}$ (128bits) + 1HMAC | 2$E$(*k*,*str*) + 1+$_2$s (128bits) + 1HMAC |
| Eq. (11): *TRNS*(3) | 2$E$(*k*,*str*) + 1+$_{2s}$ (128bits) + 1HMAC | 2$E$(*k*,*str*) + 1+$_2$s (128bits) + 1HMAC |
| Eq. (11): *TRNS*(4) | 2$E$(*k*,*str*) + 1+$_{2s}$ (128bits) + 1HMAC | 2$E$(*k*,*str*) + 1+$_2$s (128bits) + 1HMAC |
| Eq. (12):$\triangle t$ | 18*s + 5+$_2$s + 1mod | 18*s + 5+$_2$s + 1mod |
| Eq. (13): $Pk_2$ | 3$\oplus$s (128 bits) + 1+$_{2s}$(128 bits) + 1HMAC | 3 $\oplus$ s (128 bits) + 1+$_2$s (128 bits) + 1HMAC |
| Eq. (14): *DA* | does not generate | 3 $\oplus$ s (128 bits) + 1+$_2$s (128 bits) + 2-$_2$s (128bit) |
| Eq. (15): *DB* | does not generate | 3 $\oplus$ s (128 bits) + 1+$_2$s (128 bits) + 2-$_2$s (128bit) |
| *CTRNS*/*TRNS* | The same as the time of AES or DES | The same as the time of AES or DES |
| Ciphertext/ plaintext | The same as the time of AES or DES | The same as the time of AES or DES |

### 4.8. Performance Analysis

Table 6 summarizes the computational efforts required by the DES, AES, and TRNEM to encrypt and decrypt a data file.

**Table 6.** The summary of the computational efforts required by the DES, AES and TRNEM to encrypt and decrypt a data file.

| Scheme | Encryption | Decryption |
|---|---|---|
| DES (64-bit block) [23,24] | 16 $\oplus$s (32 bits) + 16 $\oplus$s (48 bits) + 1 IP (64 bits) + 1 IP-1 (64 bits ) + 128 S-Box (6 bits) + 16 Expansions (48 bits) + 16 Permutations (32 bits) | The number of operations is the same as that of the encryption process. |
| AES (128-bit block, 128-bit key) [25] | (AddRoundKey)<br>176 $\oplus$s (8 bits)<br>(SubBytes)<br>160 Substitutions (8 bit) [26]<br>(ShiftRows)<br>30 ShiftRows (128 bit)<br><br>(MixColumns)<br>36 Rijndael columns mixing [26] (128 bits) | The number of operations is the same as the sum of the numbers of those operations employed by the encryption process for the three stages, including AddRoundKey, SubBytes, and ShiftRows<br>(MixColumns)<br>36 Rijndael columns mixing [27] (128 bits).<br>(Generally, the operations of a decryption process are often more complex than those of the corresponding encryption process.) |
| TRNEM | 30+s + 32*s + 17$\oplus$s (128 bits) + 24+$_2$s (128 bits) + 8 $E(k,str)$ + 8 HMAC + 5 mod + 2*(176 $\oplus$ s (8 bits) +160 Substitutions (8 bit)+ 30 ShiftRows +36 Rijndael columns mixing ) in which the last term 2*(176 $\oplus$s …) is the time required to produce *CTRNS* from *TRNS* and generate ciphertext from plaintext | 22+s + 23*s + 12$\oplus$s (128 bits) + 15+$_2$s (128 bits) + 4-$_2$s(128bit) + 8 $E(k,str)$ + 6 HMAC + 3 mod +2*(176 $\oplus$ s (8 bits) +160 Substitutions (8 bit)+ 30 ShiftRows +36 Rijndael columns mixing ) in which the last term 2*(176 $\oplus$ s …) is the time required to produce *TRNS* from *CTRNS* and generate plaintext from ciphertext |

The following analyses show that TRNEM is more secure than the AES. First, the plaintext is encrypted by the pseudorandom key $PK_2$ when the TRNEM employs the adopted block cipher system. If the block cipher system is the AES, then the TRNEM is still more secure than it since, by Theorem 1, $PK_2$ varies at each encryption, whereas the parent key adopted by the AES is fixed for encrypying a file.Second, the ciphertext of the TRNEM is embedded in a wrapped ciphertext file. It is not easy for hackers to correctly fetch the ciphertext and analyze it. But AES does not have this protection mechanism.

Third, the AES suffers brute force attacks, e.g., the known plaintext/ciphertext attack [28], chosen plaintext attack, such as differential cryptanalysis attack [30], and linear cryptanalysis attack [21,22] since the AES is an combinatorial-logic style encryption method [29]. However, in the TRNEM, when a plaintext block is encrypted at different time points, different current time key*KCTs*and hence different other keys, including *DA*, *DB*, $Pk_1$, *TRNS*(1)~*TRNS*(4) and $PK_2$, are produced, thus resulting in diffenent wrapped ciphertext files. The value of*KCT* randomly changes and has no regular rule. Hence, the following keys generated, including *DA*, *DB*, $Pk_1$, *TRNS*(1) ~ *TRNS*(4) and $PK_2$, also randomly vary. Therefore, they can effectively defend the abovementioned attacks. In summary,*KCT*and *TRNS*are the two keys making the TRNEM more secure than the AES.

As shown in Fig. 2, due to concatenating *PRNS*1, *CDA*, *CDB*, *CTRNS* and *PRNS*2, and the lengths of them are, respectively, $\triangle h$, |*CDA*|, |*CDB*|, $\triangle L$ and $\triangle t$. Therefore, the data transmission efficiency of the TRNEM is

$$\frac{|ciphertext|}{\triangle h + |CDA| + |CDB| + \triangle L + \triangle t + |ciphertext|}$$

## 5. Conclusions and Future Work

This system utilizes a wrapping ciphertext approach, which prevents hackers from identifying the correct position of ciphertext. So the hackers cannot easily crack the protected ciphertext. Additionally, the TRNEM encrypts plaintext by using *TRNS*, which is highly random by randomly choosing a webpage and randomly accessing its content $\triangle h$ in length. Moreover, even though given the same plaintext, the TRNEM generates different ciphertext at different time points. This can effectively prevent hackers from issuing known plaintext/ciphertext attacks. So we dare to say that the TRNEM is very secure.

However, a portable encryption/decryption system, like DES and AES, does not create system parameters in it. To develop an algorithm, with which the system security codes in the TRNEM can be generated by the input password or parent key, is necessary and important. Furthermore, to enhance the performance of the TRNEM, the block cipher system adopted by the TRNEM does not need to be DEA or AES. To develop a secure and efficient encryption/decryption method, we plan to utilize the keys generated by the TRNEM, e.g.,$K\triangle h$, *DA*, *DB*, $PK_1$, $PK_2$ and *SSCs*, as the parameters to establish a new block cipher system, which is then substituted for the AES or DES to perform the block ciphering for the TRNEM. These constitute our further studies.

**Acknowledgments.** The work was partially supported by TungHai University under the project GREENs and the National Science Council, Taiwan under Grants NSC 102-2221-E-029-003-MY3, NSC 101-2221-E-029-003-MY3 and NSC 100-2221-E-029-018.

## References


1. Wiki, Computer insecurity, http://en.wikipedia.org/wiki/Computer_insecurity
2. Category, G.M.:The PPP DES Encryption Protocol. RFC 2419,September 1998, Version 2 (DESE-bis)
3. Daemen, J., Rijmen, V.: The Design of Rijndael: AES The Advanced Encryption Standard. New York, USA, Springer-Verlag.(2002)
4. Prodanović, R., Simić, D.: Holistic Approach to Wep Protocol in Securing Wireless Network Infrastructure. Computer Science and Information Systems, vol. 3, issue 2, 97-113.(2006)
5. Bahrak, B., Aref, M.R.: Impossible Differential Attack on Seven-round AES-128. Published in IET Information Security, vol. 2, Issue 2, 28 – 32. (2008)
6. Wiki, Linear cryptanalysis, http://en.wikipedia.org/wiki/Linear_cryptanalysis
7. Li, P., Sui, Y., Yang, H., Li, P.: The Parallel Computation in One-Way Hash Function Designing. International Conference on Computer, Mechatronics, Control and Electronic Engineering, Conference, vol. 1, 189 - 192. (2010)
8. Wang, M., Zhu, G., Zhang, X.: General Survey on Massive Data Encryption.International Conference on Computing Technology and Information Management, vol. 1, 150- 155.(2012)
9. Kaur, R., Kaur, A.: Digital signature. International Conference on Computing Sciences, 295-301. (2012)
10. Živković, Z.V., Stanojević, M.J.: Simulation Analysis of Protected B2B e-commerce Processes. Computer Science and Information Systems, vol. 3, issue 1, 77-91. (2006)
11. Wiki, DES, http//en.wikipedia.org/wiki/Data_Encryption_Standard
12. Matsui, M.: The First Experimental Cryptanalysis of the Data Encryption Standard. In Advances in Cryptology CRYPTO'94, Lecture Notes in Computer Science 839, Springer Verlag, 1-11. (1994)
13. Junod, P.: On the complexity of Matsui's attack. Selected Areas in Cryptography, Lecture Notes in Computer Science 2259, 199-211. (2001)
14. Knudsen, L.R., Mathiassen, J.E.: A Choice Plaintext Linear Attack. DES Fast Software Encryption, 62-272. (2000)
15. Biham, E., Shamir, A.: Differential Cryptanalysis of the Data Encryption Standard - Advances in Cryptology.The Annual International Cryptology Conference, CRYPTO '92, 487-496. (1992)
16. Biham, E., Biryukov, A.: An Improvement of Davies' attack on DES. Journal of Cryptology, vol. 10, no. 3, 195-206. (1997)
17. National Institute of Standards and Technology, Advanced Encryption Standard, NIST FIPS PUB 197.(2001)
18. Bernstei, D.J.: Cache-timing Attacks on AES. Citeseer. 2005.04. http://cr.yp.to/antiforgery/cachetiming-20050414.pdf
19. Tromer, E., Osvik, D.A., Shamir, A.: Efficient Cache Attacks on AES, and Countermeasures. Journal of Cryptology, vol. 23, Issue 1, 37-71.(2010)
20. Huang, Y.L., Leu, F.Y., Wei, K.C.: A Secure Communication over Wireless Environments by using a Data Connection Core. Mathematical and Computer Modeling, vol. 58, issues 5–6, 1459–1474. (2013)
21. Matsui, M.: Linear cryptanalysis method for DES cipher, in Advances in cryptography - Eurocrypt 1993. Springer-Verlog, Berlin, 386-397.(1993)
22. Biham, E.: On Matsui's Linear Cryptanalaysis. Springer-Verlag 1998, 341-344.(1998)
23. Katz, J., Lindell, Y.: Introduction to Modern Cryptography, Chapman & Hall/CRC Press.(2008)
24. Bellare, M., Rogaway, P.: Introduction to Modern Cryptography, Chapter 3, May 11, 2005. http://digidownload.libero.it/persiahp/crittografia/2005_Introduction_to_Modern_Cryptography.pdf
25. Federal Information Processing Standards Publication 197: Announcing the Advanced Encryption Standard (AES).(2001)

26. Cui L., Cao, Y.: A New S-Box Structure Named Affine-Power-Affine. International Journal of Innovative Computing, Information and Control, vol. 3, no. 3, 751-759. (2007)
27. Daemen, J., Rijmen, V.: AES Proposal: Rijndael. The First Advanced Encryption Standard Candidate Conference, NIST, 1999.
28. Wiki, Known-plaintext attack, http://en.wikipedia.org/wiki/Known-plaintext_attack
29. Qaosar,M., Ahmad, S.: A Combinational Logic Approach by using HDL to Implement DES Algorithm. Canadian Journal on Electrical and Electronics Engineering, vol. 3, no. 7, 384-389. (2012)
30. Biham, E., Shamir, A.: Differential Cryptanalysis of the Full 16-round DES. In E. F. Brickell, editor, Advanced in Cryptology-Crypto'92, vol. 740 of *Lectures* Notes in Computer Science, 487-496. (1992)

**Yi-Li Huang** received his master degrees from National Central University of Physics, Taiwan, in 1983. His research interests include security of network and wireless communication, solar active-tracking system, pseudo random number generator design and file protection theory. He is currently a senior instructor of Tunghai University, Taiwan, and director of information security laboratory of the University.

**Fang-Yie Leu** received his B.S., M.S. and Ph.D. degrees from National Taiwan University of Science and Technology, Taiwan, in 1983, 1986 and 1991, respectively, and another M.S. degree from Knowledge Systems Institute, USA, in 1990. His research interests include wireless communication, network security, Grid applications and Chinese natural language processing. He is currently a full professor of Tunghai University, Taiwan, the director of database and network security laboratory of the University, the chair of MCNCS and CWECS workshops, and the editorial board member of several international journals. He is also a member of IEEE Computer Society.

**Jiang-Hong Chen** graduated from Computer Science Department, Tunghai University, Taiwan, in 2012. He is now a master student of this department. His research interests include wireless communication and network security.

**William C. Chu**, the Director of Software Engineering and Technologies Center of Tunghai University, a professor of the Department of Computer Science, he had served as the Dean of Engineering College at Tunghai University, Taiwan. From 2008 to 2011, Dean of Research and Development office at Tunghai University from 2004 to 2007, Taiwan. In 1992, he was also a visiting scholar at Stanford University. His current research interests include software engineering, embedded systems, and E-learning. Dr. Chu received his MS and PhD degrees from Northwestern University in Evanston Illinois, in 1987 and 1989, respectively, both in computer science.

*Received: September 21, 2013; Accepted: January 21, 2014.*