Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2023.037161

ARTICLE

A Wrapping Encryption Based on Double Randomness Mechanism

Yi-Li Huang[1], Fang-Yie Leu[1,2,*], Ruey-Kai Sheu[1], Jung-Chun Liu[1] and Chi-Jan Huang[2,3]

[1]Department of Computer Science, Tunghai University, Taichung, 40764, Taiwan

[2]Emergency Response Management Center, Ming Chung University, Taipei, 11103, Taiwan

[3]General Education Center, Ming Chuan University, Taipei, 11103, Taiwan

*Corresponding Author: Fang-Yie Leu. Email: leufy@thu.edu.tw

Received: 26 October 2022 Accepted: 19 July 2023 Published: 31 October 2023

ABSTRACT

Currently, data security mainly relies on password (PW) or system channel key (SK_{CH}) to encrypt data before they are sent, no matter whether in broadband networks, the 5th generation (5G) mobile communications, satellite communications, and so on. In these environments, a fixed password or channel key (e.g., PW/SK_{CH}) is often adopted to encrypt different data, resulting in security risks since this PW/SK_{CH} may be solved after hackers collect a huge amount of encrypted data. Actually, the most popularly used security mechanism Advanced Encryption Standard (AES) has its own problems, e.g., several rounds have been solved. On the other hand, if data protected by the same PW/SK_{CH} at different time points can derive different data encryption parameters, the system's security level will be then greatly enhanced. Therefore, in this study, a security scheme, named Wrapping Encryption Based on Double Randomness Mechanism (WEBDR), is proposed by integrating a password key (or a system channel key) and an Initialization Vector (IV) to generate an Initial Encryption Key (IEK). Also, an Accumulated Shifting Substitution (ASS) function and a three-dimensional encryption method are adopted to produce a set of keys. Two randomness encryption mechanisms are developed. The first generates system sub-keys and calculates the length of the first pseudo-random numbers by employing IEK for providing subsequent encryption/decryption. The second produces a random encryption key and a sequence of internal feedback codes and computes the length of the second pseudo-random numbers for encrypting delivered messages. A wrapped mechanism is further utilized to pack a ciphertext file so that a wrapped ciphertext file, rather than the ciphertext, will be produced and then transmitted to its destination. The findings are as follows. Our theoretic analyses and simulations demonstrate that the security of the WEBDR in cloud communication has achieved its practical security. Also, AES requires 176 times exclusive OR (XOR) operations for both encryption and decryption, while the WEBDR consumes only 3 operations. That is why the WEBDR is 6.7∼7.09 times faster than the AES, thus more suitable for replacing the AES to protect data transmitted between a cloud system and its users.

KEYWORDS

Initial encryption key; accumulated shifting substitution; three-dimensional operation; wrapped ciphertext file

1 Introduction

In recent years, owing to the fast advance of the 5th generation (5G) networks and Internet techniques and the popularity of mobile phones, a wide range of mobile applications were proposed to provide us with a colorful living environment and enrich our daily lives. Also, with the fast development of cloud computing, people frequently send data to a cloud for storage or processing. But data transmitted via wireless channels may be stolen by hackers, conducting severe security problems. This means we need a more secure data transmission environment before the delivered data can be safely protected.

In addition, in the age of big data, the sizes of files transferred between a cloud system and users are often huge, i.e., encryption/decryption speeds should be two of the key issues if we want to deliver them via a 5G network.

At present, Advanced Encryption Standard (AES) as a block cipher mechanism has been popularly employed to secure delivered data. The AES adopts the combinational logic encryption method [1], consequently attracting different kinds of brute-force attacks [2–4]. According to references [5,6], the AES will soon be insecure since it has been partially solved. Thus, a safer block ciphering method is required shortly. In 2016, Huang et al. [7] introduced a random method to encrypt/decrypt messages/files. However, it is not truly random, since Δh is derived from password (PW) only where Δh is the length of an additional character string utilized to hide the beginning point of the ciphertext. The weakness is that Δh may be solved by Brute-force attacks [7]. After that, the same PW will be employed to crack wrapped ciphertext files.

Thus, in this study, a more secure scheme, named "Wrapping Encryption Based on Double Randomness Mechanism (WEBDR for short) is proposed. The WEBDR enhances the security level of block ciphering by wrapping ciphertext with two dynamic data sequences of variable lengths to form a wrapped file, aiming to hide the ciphertext to protect it from being accessed by hackers. The WEBDR uses four types of keys to encrypt data. The first one is called the initial encryption key (IEK), which is generated by integrating a password key (or a channel key) and an initialization vector (IV). The second is a set of sub-keys ($SK_1 \sim SK_5$), which is produced by using an accumulated shifting substitution (ASS) function and a Three-dimensional encryption method (3D Encryption). Following that, the WEBDR retrieves current time from its internal clock to produce a key, named the current time key (SK_{CT}), as the third type of key. The fourth is a random encryption key (REK) generated randomly.

Owing to using these four types of keys, even encrypting the same plaintext with the same password, the WEBDR generates different corresponding ciphertext of different lengths and different wrapped ciphertext files. Therefore, it is not easy for hackers to access and then solve the ciphertext. Our previous research results can be seen in reference [7]. The main contributions of this study are listed below:

(1) We adopt the timing-random mechanism to randomly wrap ciphertext. This can effectively prevent hackers from solving the relationship between plaintext and ciphertext even when they have ever collected a huge number of (plaintext/ciphertext) pairs.

(2) Using the encryption method of sequential-logic style, due to adopting a feedback mechanism, the generated subsequent ciphertext blocks will vary according to the contents of previous plaintext blocks. This greatly increases the difficulty of illegal decryption.

(3) The WEBDR in its message encryption (decryption) stage only invokes three exclusive OR (XOR) operations, while the AES calls this operation a total of 176 times for each of its message encryption and decryption processes.

The rest of this article is structured as follows. In Section 2, we briefly describe the related studies of this paper. Section 3 introduces the WEBDR. In Section 4, we analyze the security level of the WEBDR. Simulations and performance of the WEBDR are demonstrated and evaluated in Section 5. Section 6 summarizes this study and overviews our future research.

2 Related Studies and Background

In this section, the AES is first described. Security challenges in cloud systems and their data delivery are also discussed.

2.1 The AES and Its Problems

AES as a standard block cipher technique may have different block/key sizes, i.e., 128, 192, or 256 bits [8]. The corresponding numbers of rounds on the data encryption are 10, 12, and 14, respectively, on a 4×4-byte matrix (also called state, denoted by M). The given plaintext block is the initial value.

Giving its 10-round AES encryption as an example. A round has four operations, including SubBytes, ShiftRows, MixColumns, and AddRoundKey. But in the 0^{th} round, i.e., the initial round, only AddRoundKey is executed. The last round performs SubBytes, ShiftRows, and AddRoundKey, skipping the MixColumns. Each of the remaining 9 rounds (rounds 1–9) invokes the mentioned four operations. The SubBytes operation substitutes each byte of the state M with the help of the S-Box; the ShiftRows rotates the last three rows, i.e., rotating the i^{th} row a total of i times, $i = 2,3,4$; the MixColumns multiplies the columns of M with a polynomial function c(x); the AddRoundKey exclusive ORs (XORs) M with the round sub-key [8].

In 2002, the government of the United States (U.S.) adopted the AES as the security standard since it is the most secure encryption method at that time. However, Diehl [9] analyzed a cache attack on the AES, and [10] presented that a biclique attack has been successfully applied to attack AES [8,11] introduced different types of AES attacks, meaning that the AES will be solved soon, or at least, it is not really secure.

2.2 Data Security and Encryption

Today, cloud and Internet of Things (IoTs) systems are two popular applications in the world. Their data securities are essential before these applications can be successfully applied to the world. Reference [12] defined cloud security as the policies, services, controls, and technologies that prevent cloud data, infrastructure, and applications from threats. 7 challenges are also proposed. Among them, Granular Privilege and Key Management are concerned with privilege and cryptography keys. In reference [13], cloud security refers to a broad set of techniques and control methods used to protect data, applications, and cloud computing infrastructures. Because data archived in a cloud system can be accessed by using multiple client devices, when uploading data to the cloud, for security reasons, we need to consider who may access the data (e.g., the staff of the cloud system), and what applications and what methods will be, respectively, requested and utilized to access the data.

Bordak [14] mentioned that before cloud storage, plain-text data can be encrypted to differentiate the ability to save data from the ability to retrieve it. So, it would be better if the encryption key is securely protected to ensure that only authorized users can decrypt data.

Musa et al. [15] enforced their symmetric key encryption to protect a file locally on the client side before uploading it to the cloud system and the file is decrypted after it is downloaded on the client side using the key produced during encryption. Keys are generated by different algorithms, thus offering better security levels and enhanced system performance for large files.

Reis et al. [16] said that cryptography for cloud applications relies on both client-side and server-side cryptographies. The AES-256 in Cipher Block Chain (CBC) mode is employed to encrypt their healthcare data. Client-side cryptography encrypts data at the user's device before sending data to the cloud storage, aiming to ensure user-data privacy and security. Server-side encrypts data before storage, i.e., inside the cloud system, for the reason that encrypts data, saves data, and manages keys at the same location. Of course, before these activities, the ciphertext sent by users should be decrypted first.

Banuelos [17] mentioned that users often utilize integers as keys by invoking a pseudo-random number generator or random-number generator. Sometimes, strings comprising numbers and letters are adopted. Also, a longer key is required, because longer keys consume a longer time to crack. The author also presented that SkyFlow, a data privacy vault company, uses a granular method to encryption keys that conveys a master key named a Key Encryption Key (KEK) and Data Encryption Keys (DEKs). Users may use Amazon Web Services Key Management Service (AWS KMS), Skyflow Key Management, or a bring-your-own-key (BYOK) technique to administrate KEK. But their data stored in the company's vault is still encrypted by using DEKs.

Reference [18] described that an IoT security solution is required for business. Without security, businesses can be vulnerable to hacks and data breaches, making private information exploited and the public which will threaten the reputation and well-being of these corporations/companies.

Schacht et al. [19] evaluated 5 million Open Pretty-Good-Privacy (OpenPGP) keys with the algorithms utilized and internal parameters selected when establishing connections to third-party software. The authors analyzed the properties of keys and the trends of OpenPGP usage in the passing two decades, providing an internal look at OpenPGP and the adoption of public key cryptography. Looking at the details of the keys over time can make us recommend key features that affect real-world use. The analyses of OpenPGP keys give users a way to determine the time duration for changing the default settings of software packets.

Roundy [20] presented that IoT security risks were rising and stated the challenges listed in Verizon Mobile Security Index 2021 for mirroring mobile to the IoT environment. The author proposed a 6-step procedure to better IoT security. The last step is encrypting user and application data, aiming to protect the data from malicious actors. Without cryptography, an organization may face sensitive data leakage, reputational damage, and penalties.

Yang et al. [21] proposed an algorithm by exploiting encrypted packets and modeling network traffic to uncover stepping-stone statements/intrusions. The software tool used is OpenSSH which comprises n paths between Host 0 and Host 1. Each path has its cryptography key under the assumption that Host 0 acts as an intruder, and Host 1 plays the role of a victim. When a path is built, an encryption key is given. Authors claimed that the algorithm demonstrates better performance when detecting intruders' both-side chaff attacks. However, it is better if the keys can be created with a secure approach.

Nowaczewski et al. [22] predicted that Customer Edge Switching (CES) would be used in 5G networks. They described the CES and explained how it works with Domain Name System (DNS). The possible attack models were also discussed. Currently, DNSs lack encryption/authentication. Hackers can exploit the system through man-in-the-middle attacks. They also extend CES's implementation to fix this gap by adding DNSCrypt and DNSSEC functionalities. Their experimental results show that most attacks can be effectively detected by the proposed countermeasures. However, it would be better if the details of cryptography can be presented.

2.3 Three Working Models of Data Encryption

For those systems requesting high-security levels for their data transmission, three data transmission modes can be considered. Mode 1 is encrypting files transmitted between a user and a base station (BS) or a cloud with a channel key established to ensure their point-to-point security [23,24]. With mode 2, data is saved in client devices before its delivery, i.e., data is encrypted before transmission [25]. Therefore, a password given by the user is processed to generate a password key with which to encrypt/decrypt data files. Mode 3 adopts Proxy-based encryption methods to secure archived data. For some existing software or applications with no encryption functions, data can be encrypted by proxy servers [26] before transmission. Our opinion is that the WEBDR can enhance the security and performance of modes 1 and 2, particularly for those medium and large-size files.

2.4 Other Related Studies

Chakravarthy et al. [27] proposed a system named digital signature algorithm (DSA) which works together with deep packet inspection (DPI), known as the DSA-DPI model, to detect and prevent Distributed Denial of Service (DDoS) attacks. DDoS is an attack that overloads Central Process Units (CPUs) of the firewall and other network components and/or consumes their network bandwidths. The proposed system also provides preventive warnings on infrastructure before the malware attack. However, this system does not discuss how to protect, e.g., encrypt/decrypt, data itself. Digital signatures are one kind of anomaly-based detection scheme. Often a signature-based approach is required. DPI is often a function of firewalls. However, packet filtering often consumes a longer checking time.

Chiu et al. [28] proposed a network autonomous security system, named Detection and Defense of Denial of Service (DoS)/DDoS on 5G (DDD5G) which analyzes 5G network traffics and determines whether a protected system is under DoS/DDoS attack or not by using Shannon entropy (SE) and/or a mixed model. The latter mixes Shannon entropy and Cumulative Sum Algorithm (CUSUM) to further enhance a system's security level. Shannon entropy adopts entropy derived from normal traffic at specific time intervals as the threshold and compares it with the entropies of other time intervals, denoted by T, to detect whether there are intrusions and attacks in T or not, while the CUSUM collects traffic and checks to see whether it exceeds the predefined thresholds or not to determine if this system is under attack. Authors claimed that a mixed-mode approach can effectively detect DDoS. However, with the two-stage detection approach, the detection time may be long, i.e., unable to detect attacks in a real-time manner.

Tsai et al. [29] proposed a Two-stage High-efficiency Long Range Wide Area Network (LoRaWAN) encryption key Update Scheme (THUS for short) for changing LoRaWAN's session keys and root keys in an efficient and secure approach. The THUS comprises two stages, i.e., the Root Key Update (RKU) and the Session Key Update (SKU), and with different update periods, the security levels of RKU and SKU are higher than those of normal LoRaWAN specifications. A modified AES cryptography process is also adopted in the THUS to improve the THUS's security level. According to the authors' security analyses, the THUS can effectively protect important parameters in its key update stages, and satisfies the requirements of integrity mutual authentication, and confidentiality.

Moreover, The THUS can further resist replay and eavesdropping attacks. However, THUS procedures can only be applied to LoRaWAN since the mechanism is limited to LoRaWAN, i.e., join-server, end-device, and network server. Also, when a sender generates a new D-box, it needs to deliver it (of course, encrypting it) to his/her target site. Otherwise, the target site does not know how to solve the receiving message, thus increasing the processing costs.

Khan et al. [30] stated that traditional authentication protocols are vulnerable in the quantum computing era. Therefore, they presented an authentication protocol according to the lattice technique for public cloud environments to prevent quantum attacks and avoid all known typical attacks. This protocol provably secures the protected systems with the Real-Or-Random (ROR) model. Their simulation results showed that this protocol is lightweight compared with some existing lattice-based authentication protocols. Their comparative analyses also demonstrated that this protocol is quite appropriate to be implemented in quantum-based environments. However, this scheme is developed for authentication, instead of encrypting/decrypting transmitted data.

Khalaf et al. [31] presented that hackers may send malicious inputs to confuse a web application. The purpose is to access or disable the application's back end. The authors claimed that Cross-site scripting (XSS) and Structured Query Language (SQL) Injection Attacks (SQLIAs) are frequently launched. They then developed an input validation mechanism to check and evaluate for program codes and also developed a script whitelisting interception layer that is a part of the browser's JavaScript engine. The SQLIA can be detected and the XSS attack is resolved with the approach of input verification and script whitelisting by using pushdown automatons. However, this system only focuses on SQLIA, XSS, and buffer overflow.

Yang et al. [32,33] described that Age-of-information (AoI) as an indicator reflects the freshness of data during the communication stage and Unmanned Aerial Vehicles (UAVs) play very important roles in Mobile Edge Computing Networks (MECN). They tried to solve the Channel Access Attack (CAA) problem of AoI-oriented channel access from game-theory viewpoints. A system model with active probability is first built to acquire a MECN-based AoI indicator under CAA attacks. Next, they proposed the AoI-based channel access optimization problem by using Ordinary Potential Game (OPG). At last, a learning algorithm named Distributed Channel Access Strategy Determination (DCASD) is presented to choose the channel access strategies. The experiments given different parameters to enhance the performance of the algorithm are conducted as compared with some state-of-the-art systems. But the proposed scheme is not applied to encrypt/decrypt data. Further, readers may like to know how attackers access the available channels to intrude on sensor nodes. How to implement the proposed approach with Carrier Sense Multiple Access (CSMA) families? How to work with IEEE 802.11 ax/be? It would be better if authors can deeply describe these.

3 The WEBDR

The WEBDR dynamically hides ciphertext in a wrapped cipher file, aiming to hide the right position of ciphertext. Thus, it is not easy for hackers to collect a huge amount of effective (plaintext, ciphertext) pairs with which to break the system.

3.1 Parameters and Operators

All parameters and operators adopted by the WEBDR are listed and defined below:

A. Parameters

Parameters used are as follows:

1. IV: initialization vector, which is inputted to a cryptographic primitive by users to provide the initial state of the WEBDR.
2. PW: the password, comprising 8 to 32 characters, is prepared as one of the inputs by users.
3. SK_{PW}: the system password key derived from PW.
4. dsc: dynamically shifting count when shifting data.
5. SK_{CH}: the system channel key, created for a user and the cloud sever before their communication starts.
6. SK_0: the system zeroth encryption key defined as $SK_0 = SK_{PW}$ or $SK_0 = SK_{CH}$.
7. IEK: the initial encryption key.
8. $SK_1 \sim SK_5$: the system sub-keys produced in the system's initial procedure.
9. $PRS1$: pseudo-random sequence 1, as a random string placed at the beginning of a wrapped ciphertext file.
10. $PRS2$: pseudo-random sequence 2, as a random string placed at the end of a wrapped ciphertext file.
11. $\Delta_1 l$: $|PRS1|$ in bytes. Its usage will be described later.
12. $\Delta_2 l$: $|PRS2|$ in bytes. Its usage will be described later.
13. SK_{CT}: the system time key, produced based on current CPU time, is 128 bits long comprising the following elements: nanosecond/date/hour/minute/second/nanosecond/hour/minute/second.
14. SK_{RCT}: the reverse key of SK_{CT}, 128 bits long, consists of the following elements: second/minute/hour/nanosecond/second/minute/ hour/date/nanosecond.
15. REK: Random Encryption Key, which is employed to generate ciphertexts and the length of $PRS2$.
16. $CREK$: the Ciphertext key of REK.
17. $fb_0 \sim fb_n$: a sequence of internal feedback code.
18. Plaintext blocks: $P_1P_2 \ldots P_j \ldots P_n$, where P_j is plaintext block j and $|P_j| = 128$ bits, $1 \leqq j \leqq n$.
19. Ciphertext blocks: $C_1C_2 \ldots C_j \ldots C_n$, where C_j is ciphertext block j and $|C_j| = 128$ bits, $1 \leqq j \leqq n$.

B. Operators

The operators employed and their functions are defined as follows:

1. XOR, denoted by $\bigoplus$.

Encrypting plaintext p to ciphertext c with key k, i.e., $c = p \bigoplus k$.

Decrypting c to p with k, i.e., $p = c \bigoplus k$.

2. Binary adder [7]: $+_2$

Encrypting plaintext p to ciphertext c with key k, i.e., $c = p +_2 k$, in which we drop the carry generated by the addition of the most significant bit

Decrypting c to p with k, i.e.,

$$p = c -_2 k = \begin{cases} c - k, & if\ c \geq k \\ c + \overline{k} + 1, & if\ c < k \end{cases}, \text{ in which. } -_2 \text{ is the reverse operation of } +_2.$$

3. Rotate-Equivalence operator: $\odot_R$

Encrypting plaintext p_i to ciphertext c_i with key k, i.e.,

$c_i = p_i \odot_R k = p_{iR} \odot k$, where p_{iR} is the key acquired by rotating plaintext p_i clockwise h bits where $h = |k|/4$, i.e., if $|k| = 128$, p_i will rotate 32 bits.

Decrypting c_i to p_i with k, i.e., $p_i = c_i \odot_{IR} k =$ counterclockwise rotating $(c_i \odot k)$ a total of $|k|/4$ bits.

4. Three-dimensional operation: the operation encrypting a message by using encryption keys and three fundamental operators [1], i.e., $\bigoplus$, $+_2$ and $\odot_R$.
5. Modulus operator: mod.

$c = p \bmod n$, where n is a positive integer.

6. *Left* (PW, n): a function that retrieves n leftmost characters from PW, where $n \leqq |PW|$ in bytes.
7. *Right* (PW, n): a function that accesses n rightmost characters from PW, where $n \leqq |PW|$ in bytes.
8. *Trunc* (RN, t): a function that truncates the rightmost t bytes from the random number key RN.

C. Accumulated shifting substitution

In the AES, the SubBytes is a mapping/substitution operation following the content of a given lookup table, i.e., a substitution box (S-Box). Basically, this mapping is a combinatorial-logic style encryption. The substring X in bytes appears at different locations in the plaintext will produce the same cipher substring S(X), consequently decreasing its security level since the mapping from X to S(X) is fixed, rather than a one-to-many mapping.

Next, the Accumulated Shifting Substitution algorithm (ASS), i.e., Algorithm 1, defined below is a sequential-logic style encryption mechanism which encrypts a plaintext into an irreversible ciphertext. The same substring Xs at different locations of the plaintext will be mapped to different cipher substrings. In other words, this is a one-to-many relationship, aiming to significantly enhance the security level of ciphertext.

Algorithm 1: ASS Algorithm

```
Input: a lookup table D-Box and a given character string $ST$ of $n$ bytes long, $n \in Z^+$.
Output: ciphertext $C$.
{ 1) Let $ST = st_1 st_2 st_3 \ldots st_n$ where $st_j$ is a 8-bit character, $1 \leqq j \leqq n$; /*a character string*/
  2) $dsc = \begin{cases} ASCII(st_1), & if\ n = 1 \\ ASCII(st_1) + ASCII(st_n), & if\ n \geq 2 \end{cases}$; /*initial dynamic shift counter*/
  3) For $i = 1$ to $n$
       { $dsc = (dsc + i * ASCII(st_i) + i)\ mod\ 256$; /* modifying $dsc$ value*/
         $tch$ = the corresponding character in D-Box after $st_i$ is substituted; /* $tch$: the temporary
                 substituting character of $st_i$ */
         $ch_i$ = the target character obtained by shifting $tch$ in D-Box $dsc$ times following the index
                 sequence of D-Box;}
  4) $C = ch_1 ch_2 ch_3 \ldots ch_n$;}
```

3.2 Password Key (SK_{PW})

In the WEBDR, SK_{PW} is the initial key. Its content significantly affects the security level of the WEBDR. To generate SK_{PW}, we expand PW following three rules:

(1) The original content of PW is preserved;

(2) The code expanded is generated based on the original content of PW;

(3) When the same character repeatedly appears in PW, the expanded codes varies. The algorithm deriving SK_{PW} from PW is shown in Algorithm 2.

Algorithm 2: Deriving SKPW from PW

```
Input: user password $PW$.
Output: system password key $SK_{PW}$.
{ 1) Let $l = |PW|$ in bytes; /*the length of the given password*/
  2) If ($l < 8$), then request users choosing a new $PW$; /*should $l \geq 8$*/
  3) If ($l = 16$), then {$SK_{PW} = PW$; stop;};
  4) If ($l > 16$), then {$SK_{PW} = Left(PW, 16) +_2 Right(PW, 16)$; stop;}; /*$l$ is fixed to 16*/
  5) $n = 16 - l$; $SK_{PW}$ = Null; /* $8 \leq l \leq 15$, $1 \leq n \leq 8$*/
  6) $w_1w_2w_3 \ldots w_n$ = Left($PW$, $n$), where $w_j$ is an 8-bit character, $1 \leq j \leq n$;
  7) $c_1c_2c_3 \ldots c_n$ = ASS ($w_1w_2w_3 \ldots w_n$, D-Box), where $c_j$ is the corresponding image of $w_j$, $1 \leq j \leq n$;
  8) For $i = 1$ to $n$
     $SK_{PW} = SK_{PW} \parallel w_i \parallel c_i$; /* $\parallel$: concatenation for concatenating $w_i$ and $c_i$*/
  9) $SK_{PW} = SK_{PW} \parallel Right(PW, l–n)$; /*finalizing $SK_{PW}$*/}
```

3.3 Encryption/Decryption

In the WEBDR, before data encryption, there is an initial process used to generate system sub-keys $SK_1 \sim SK_5$ and $\Delta_1 l$ by using PW or a channel key SK_{CH}, both of which have been enhanced by invoking Algorithm 2. The key length is 128 bits.

A. Initial process

The initial process of the WEBDR is shown below:

1. A string I, inputted by user, may be PW or SK_{CH};
2. If $|I| \neq 16$ bytes,
   then {derive SK_0 from I by invoking Algorithm 2;}
   else $SK_0 = I$;
3. Producing a random number RN and let $IV = RN$;
4. $IEK = SK_0 \oplus IV$;
5. Deriving SK_1 from IEK by employing ASS Algorithm, i.e., $SK_1 = ASS(IEK, D - Box)$;
6. $SK_2 = (SK_0 +_2 SK_1) \oplus (SK_1 \odot_R IV)$; (1)
7. Deriving SK_3 from SK_2 by invoking Algorithm 1, i.e., SK_3 = ASS(SK_2, D-Box);
8. $SK_4 = (SK_0 +_2 SK_3) \oplus SK_2$; (2)
   $SK_5 = (SK_1 \odot_R SK_4) +_2 (SK_2 \oplus IV)$; (3)
9. Producing $\Delta_1 l$, $3 \leq \Delta_1 l \leq 1024$ where $\Delta_1 l = ((SK_0 +_2 SK_5) \oplus (SK_1 \odot_R SK_4) \oplus (SK_2 +_2 SK_3))\ mod\ 1022 + 3$; (4)

B. Message encryption

Message encryption has four steps:

Step 1: Producing REK and $CREK$

1. Producing the 0th random encryption key REK_0;
2. Fetching CPU time with which to produce current time key SK_{CT} and SK_{RCT}, i.e., the reverse key of SK_{CT};
3. Yielding the random encryption key REK,

$$REK = (REK_0 +_2 SK_{CT}) \oplus (REK_0 \odot_R SK_{RCT}); \quad (5)$$

4. Encrypting REK to generate $CREK$ where

$$CREK = ((REK +_2 SK_1) \odot RSK_4) \oplus ((SK_2 \oplus SK_3) +_2 SK_5); \quad (6)$$

Step 2: Producing ciphertext and $\Delta_2 l$

1. Let $P_1P_2P_3 \ldots P_n$ be plaintext, and let $C_1C_2C_3 \ldots C_n$ be the corresponding ciphertext, where $|P_j| = |C_j|$ is 128 bits long, $1 \leq j \leq n$;
2. $C_0 = SK_3$;
   $fb_0 = SK_4$;
3. For plaintext block P_i, $1 \leq i \leq n$;

$$fb_i = (P_i \odot_R fb_{i-1}) +_2 ((C_{i-1} \oplus SK_5) +_2 fb_{i-1}); \quad (7)$$

$$C_i i = ((P_i \odot_R fb_{i-1}) +_2 (REK \oplus fb_{i-1})) \oplus ((C_{i-1} \oplus SK_5) +_2 fb_{i-1}); \quad (8)$$

4. $\Delta_2 l = ((SK_2 +_2 REK) \odot_R SK_5 +_2 (SK_4 \oplus REK))\, mod\, 1022 + 3;$ (9)

Step 3: Yielding $PRS1$ and $PRS2$

1. Yielding a random encryption key REK_1;
2. Fetching CPU time with which to produce a system current time key SK_{CT1};
3. $RN(0) = REK_1 \oplus SK_{CT1}$;
   $t(0) = REK_1$;
   $n_1 = \lceil \Delta_1 l/16 \rceil$; /∗ceiling function∗/
   $n_2 = \lceil \Delta_2 l/16 \rceil$;
   $\Delta_1 = 16n_1 - \Delta_1 1$;
   $\Delta_2 = 16n_2 - \Delta_2 l$;
4. For $i = 1$ to $n_1 + n_2$
   $\{RN(i) = (RN(i-1) +_2 t(i-1)) \oplus (t(i-1) +_2 SK_{CT1})$;
   $t(i) = RN(i-1) +_2 t(i-1);\}$
5. $R1 = trunc(RN(n_1), \Delta_1)$;
   $R2 = trunc(RN(n_1 + n_2), \Delta_2)$;
   $PRS1 = RN(1) \parallel RN(2) \parallel RN(3) \parallel \ldots\ldots \parallel RN(n_1 - 1) \parallel R1$;
   $PRS2 = RN(n_1 + 1) \parallel RN(n_1 + 2) \parallel RN(n_1 1 + 3) \parallel \ldots \parallel RN(n_1 + n_2 - 1) \parallel R2$;

Step 4: Concatenating IV, $PRS1$, $CREK$, $Ciphertext$ and $PRS2$ sequentially to produce the wrapped ciphertext file, the layout of which is illustrated in Fig. 1.

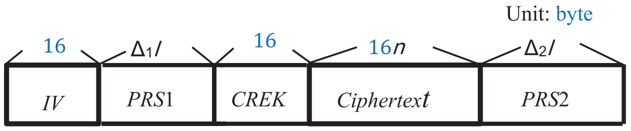

Figure 1: The layout of a wrapped ciphertext file

C. Message decryption

The decryption process is shown below:

1. Identifying the exact position of IV from the very beginning of the wrapped ciphertext file, and deleting it from the file;
2. Invoking the initial process to generate $SK_1 \sim SK_5$ and $\Delta_1 l$;
3. Deleting $PRS1$ from the remaining file based on $\Delta_1 l$;
4. Identifying the exact position of $CREK$ from the very beginning of the remaining portion of the file, and deleting it from the file;
5. Decrypting $CREK$ to acquire REK,

$$REK = (CREK \oplus ((SK_2 \oplus SK_3) +_2 SK_5)) \odot_{IR} SK_4 -_2 SK_1; \tag{10}$$

6. $\Delta_2 l = ((SK_2 +_2 REK) \odot_R SK_5 +_2 (SK_4 \oplus REK))\ mod\ 1022 + 3$;
7. According to $\Delta_2 l$ to remove $PRS2$ from the end of the remaining part of the file to acquire the ciphertext;
   $n = |\text{Ciphertext}|/16$;
   $fb_0 = SK_4$;
   $C_0 = SK_3$;
8. For $i = 1$ to n

$$\{P_i = (C_i \oplus ((C_{i-1} \oplus SK_5) +_2 fb_{i-1})) -_2 (REK \oplus fb_{i-1}) \odot_{IR} fb_{i-1}; \tag{11}$$

$$fb_i = (P_i \odot_R fb_{i-1}) +_2 ((C_{i-1} \oplus SK_5) +_2 fb_{i-1}); \tag{12}$$

9. Output the plaintext $P_1 P_2 \ldots P_n$.

4 Security Analyses

We analyze security of the WEBDR's working environment on operation mode 1 and mode 2 stated above, containing the securities of system sub-keys $SK_1 \sim SK_5$, $\Delta_1 l$, the dynamic random key REK, the wrapped ciphertext file, and the ciphertext, and then demonstrate how they resist against eavesdropping attacks.

4.1 Security on $SK_1 \sim SK_5$

In the WEBDR, the system sub-keys $SK_1 \sim SK_5$ are employed to encrypt/decrypt a given plaintext. Claimed 1 shows that $SK_1 \sim SK_5$ have achieved practical security.

Claimed 1:

When the WEBDR worked on operation mode 1 or mode 2, the generated system sub-keys $SK_1 \sim SK_5$ have achieved practical security.

Proof: In operation mode 1 or mode 2, SK_0 is defined as $SK_0 = SK_{CH}$ (or $SK_0 = SK_{PW}$), where SK_{CH} (or SK_{PW}) is inputted externally, meaning that it is not easy for hackers to guess the value of SK_0. In the two modes, SK_0 is used only once. Thus, the problems resulted from collecting massive data for solving SK_0 can be prevented. Since SK_0 and IV employed in a session are themselves unique from those used in other sessions. Hence, to crack $SK_1 \sim SK_5$, apart from blindly guessing their values, hackers can also utilize three approaches. The first is solving Eqs. (1) to (3). The second is breaking Eq. (4). The last is cracking Eqs. (6) to (9) and solving the wrapped ciphertext file construction (WCFC) by adopting brute-force attacks.

About the first approach, in Eq. (3), $SK_5 = (SK_1 \odot_R SK_4) +_2 (SK_2 \oplus IV)$, in which SK_5 is derived from four parameters, including IV, SK_1, SK_2, and SK_4 and the three-dimensional operation mentioned above. $SK_4 = (SK_0 +_2 SK_3) \oplus SK_2$ in Eq. (2) is derived by utilizing SK_0, SK_2, and SK_3, in which SK_3 is produced by invoking the ASS Algorithm given D-Box and SK_2. Further, SK_1 in the equation $SK_2 = (SK_0 +_2 SK_1) \oplus (SK_1 \odot_R IV)$ (see Eq. (1)) is again acquired by calling the ASS Algorithm given D-Box and $IEK$where $IEK = SK_0 \oplus IV$. Now we can conclude that without correct values of SK_0 and IV, $SK_1 \sim SK_5$ cannot be solved. However, the value of SK_0 is unknown to hackers. In other words, $SK_1 \sim SK_5$ are difficult to break. Also, at different time points, the IV values are different, meaning that hackers have insufficient data to break our system.

For the second approach, $\Delta_1 l = ((SK_0 +_2 SK_5) \oplus (SK_1 \odot_R SK_4) \oplus (SK_2 +_2 SK_3))\ mod\ 1022 + 3$ in Eq. (4) is directly derived from SK_0 and $SK_1 \sim SK_5$. Even ($SK_0 = SK_{PW}$) is reused, $\Delta_1 ls$ varies for different IVs and $SK_1 s \sim SK_5 s$ in different sessions. Thus, no matter whether with operation mode 1 or 2, security is effectively ensured since it is not easy for hackers to derive $\Delta_1 l$ from Eq. (4). Furthermore, even in the case that $\Delta_1 l$ is really accessed by hackers, because $3 \leqq \Delta_1 l \leqq 1024$, and $0 \leqq K_j \leqq (2^{128} - 1)$, $0 \leqq j \leqq 5$, it is hard for hackers to crack K_j from $\Delta_1 l$. That is, without guessing the exact value of SK_0, it is almost impossible to solve $SK_1 \sim SK_5$ by using Eq. (4). Then, we dare to say that $SK_1 \sim SK_5$ are difficult to break.

For the third approach, hackers may break Eqs. (6) to (9) and the wrapped ciphertext file construction by employing brute-force attacks. In mode 1 or mode 2, without knowing plaintext, hackers cannot launch chosen-plaintext attacks and known-plaintext attacks. They can only collect and analyze wrapped ciphertext files. Even though hackers have collected a huge amount of wrapped ciphertext files encrypted by SK_0, each ciphertext file has its own IV which is a random number so that the generated keys, i.e., $SK_1 \sim SK_5$, are themselves different from those $SK_1 s \sim SK_5 s$ produced in other sessions, indicating that hackers cannot acquire the value of $\Delta_1 l$, hence unable to retrieve $CREK$ from the file and acquire the random encryption key, i.e., REK. Even though hackers have guessed the value of $CREK$, based on Eq. (6), i.e., $CREK = ((REK +_2 SK_1) \odot_R SK_4) \oplus ((SK_2 \oplus SK_3) +_2 SK_5)$, when REK is unknown, it is hard to solve REK and $SK_1 \sim SK_5$ based on the $CREK$. Moreover, at different CPU time points, the REK varies, i.e., lacking enough data for hackers to break these parameters.

Furthermore, the WEBDR generates a random key REK for each plaintext P. Each time when the plaintext encryption process is executed, different $SK_1 \sim SK_5$, SK_{CT} and REK values will be derived and given to produce different wrapped ciphertext files; thus, breaking those parameters from these collected wrapped ciphertext files is difficult. Also, $\Delta_1 l$, $\Delta_2 l$ and the size of the plaintext file are unknown to hackers. Consequently, it is not easy for hackers to identify the place of the ciphertext in the wrapped ciphertext file to obtain the ciphertext, thus unable to break the WEBDR.

Next, even hackers correctly guesses the exact location of the ciphertext and obtain (P_i, C_i), $1 \leqq i \leqq n$, pairs, without the value of fb_0 (i.e., SK_4), C_0 (i.e., SK_3), plaintext block P_i, the system sub-key SK_5, the random key REK, and the internal feedback code fb_{i-1}, hackers are unable to obtain SK_3, SK_4, P_i, fb_{i-1}, SK_5, and REK from the ciphertext block C_i based on Eqs. (7) and (8), due to the sequentially generated internal feedback code fb_{i-1}, $1 \leqq i \leqq n$, i.e., $SK_1 \sim SK_5$ are difficult to break.

From the analyses above, we dare to say that in operation mode 1 or mode 2, the generated system sub-keys $SK_1 \sim SK_5$ have achieved practical security (Q.E.D).

4.2 Security on $\Delta_1 l$

The key point for solving the wrapped ciphertext files is acquiring $\Delta_1 l$. But it is crucial for hackers to solve the WCFC and then access the corresponding $CREK$ and ciphertext. That is, $\Delta_1 l$ needs a higher level of security. In operation mode 1 or 2, the probabilities of cracking $\Delta_1 l$ by using probable approaches are similar to that of a blind guess. So, we dare to declare that $\Delta_1 l$ has its practical security. We analyze this in Claimed 2.

Claimed 2:

In operation mode 1 or 2, $\Delta_1 l$ derived from the zeroth encryption key SK_0 and system sub-keys $SK_1 \sim SK_5$ has achieved its practical security.

Proof: Besides a blind guess, there are only two approaches for hackers to solve the value of $\Delta_1 l$, i.e., by solving Eq. (4) or breaking the wrapped ciphertext file format by using brute-force attacks.

With the first approach, in operation mode 1, data transmitted between a user and a base station (or the cloud) is encrypted, i.e., $SK_0 = SK_{CH}$. In operation mode 2, the data file owned by the user is encrypted, i.e., $SK_0 = SK_{PW}$. As mentioned above, both SK_{CH} and SK_{PW} are given by users, i.e., the two parameters are external to our system. Therefore, it is not easy for hackers to correctly guess the one given. Namely, SK_0 is difficult to break.

According to Claimed 1, $SK_1 \sim SK_5$, sequentially derived from SK_0 and IV, also have their own practical securities. Without knowing the values of $SK_0 \sim SK_5$, hackers cannot solve Eq. (4) to break $\Delta_1 l$. The reason is that to calculate the value of $\Delta_1 l$, SK_0 and $SK_1 \sim SK_5$ ought to be solved beforehand. But, the key length is 128 bits. The possible values ranging from 0 and $2^{128} - 1$ is far wider than the probable values of $\Delta_1 l$ which is between 3 and 1024. Without knowing the values of $SK_0 \sim SK_5$, the probability of solving $\Delta_1 l$, produced by using $SK_1 \sim SK_5$ with a Three-dimensional operation and the modulus operation ($mod\ 1022 + 3$), is equal to that of a blind guess [1].

By using the second approach, hackers may try to analyze the wrapped ciphertext file format with brute-force attacks. But, in operation mode 1 or mode 2, they cannot solve the file by employing chosen-plaintext attacks or known-plaintext attacks because they are unable to solve $\Delta_1 l$ and $\Delta_2 l$ and thus do not comprehend the location of ciphertext in the file. Even though the collected wrapped ciphertext files are produced by giving the same SK_0, the values of $\Delta_1 l$ s of all ciphertext files vary due to giving different IVs. The conclusion is that $\Delta_1 l$ is difficult to break.

In addition, the ciphertext is placed between $PRS1$ and $PRS2$ and $CREK$ (see Fig. 1). $PRS1$ and $PRS2$ are all derived from random numbers or pseudo-random numbers. They cannot be solved from the wrapped ciphertext files collected. Also, the size of a wrapped ciphertext file is $(16 + \Delta_1 l + 16 + 16n + \Delta_2 l)$ bytes, where the two 16 bytes are the sizes of IV and $CREK$, $16n$ bytes represent the length of the plaintext and $\Delta_1 l$ $(\Delta_2 l)$ is the size of $PRS1(PRS2)$. Hackers do not know the values of $\Delta_1 l$ and $(\Delta_2 l)$, the probability with which to obtain $\Delta_1 l$ by cracking the entire wrapped ciphertext file structure is the same to that of a blind guess. Here, we can conclude that $\Delta_1 l$ has achieved its practical security (Q.E.D).

4.3 Security on Random Encryption Key REK

Since the dynamic random key REK is used to encrypt a plaintext file, it requires a relatively high level of security. In Eq. (5), i.e., $REK = (REK_0 +_2 SK_{CT}) \oplus (REK_0 \odot_R SK_{RCT})$, REK is generated by utilizing the Three-dimensional operation, and three parameters, including the zeroth random encryption key REK_0 and the two current time keys SK_{CT} and SK_{RCT}, showing that REK is a highly dynamic random key. Theorem 1 will prove that REK's security level is high.

Theorem 1:

In operation mode 1 or 2, if the dynamic random encryption key REK generated is n bits long, the probability with which to obtain REK from an intercepted wrapped ciphertext file is $\frac{1}{2^n}$.

Proof: To break REK, in addition to a blind guess, hackers could also adopt the following three methods. The first is cracking Eq. (6). The second is solving Eqs. (7) and (8). The third is breaking Eq. (9) and analyzing the wrapped ciphertext file format.

Firstly, in operation mode 1 or 2, from previous analyses, we know that $\Delta_1 l$ and $SK_1 \sim SK_5$ have their practical securities. Thus, when $\Delta_1 l$ is unknown, hackers cannot find the position of $CREK$ in the wrapped ciphertext file. Thus, the probability with which to obtain REK is the same to that of a blind guess. Even though hackers may somehow accurately retrieve $CREK$, and try to obtain REK by solving Eq. (6), i.e., $CREK = ((REK +_2 SK_1) \odot_R SK_4) \oplus ((SK_2 \oplus SK_3) + SK_5)$, without knowing the values of $SK_1 \sim SK_5$, they are unable to obtain REK with only one value of $CREK$. Therefore, when the values of $SK_1 \sim SK_5$ are unknown, the probability with which to obtain REK by solving the information concerning $CREK$ is the same to that of a blind guess, i.e., $\frac{1}{2^n}$ [1].

Secondly, in operation mode 1 or mode 2, hackers are unable to successfully crack the WEBDR by submitting chosen-plaintext attacks and known-plaintext attacks. They can only crack the system by analyzing the wrapped ciphertext files collected. But without the values of $\Delta_1 l$ and $\Delta_2 l$, no clues of the exact place of the ciphertext within the wrapped ciphertext file can be found, meaning that the attacker cannot successfully retrieve the ciphertext from the wrapped ciphertext file and crack it. In this case, Eqs. (7) and (8) are not helpful for hackers, i.e., the probability with which to recover REK by solving Eqs. (7) and (8) is the same to that of a blind guess.

On the other hand, hackers may somehow accurately guess where the ciphertext block C_i is (e.g., by brute-force approaches), $1 \leqq i \leqq n$, and attempt to solve Eqs. (7) and (8). But in Eq. (8), i.e., $C_i = ((P_i \odot_R fb_{i-1}) +_2 (REK \oplus fb_{i-1})) \oplus ((C_{i-1} \oplus SK_5) +_2 fb_{i-1})$, the internal feedback key $fb_{i\text{-}1}$, $1 \leqq i \leqq n$, is generated by utilizing Eq. (7) (i.e., $fb_i = (P_i \odot_R fb_{i-1}) +_2 ((C_{i-1} \oplus SK_5) +_2 fb_{i-1})$) and the four parameters, including fb_0, C_0, REK, and SK_5, are unknown. Therefore, the value of fb_1 cannot be uncovered. Likewise, since fb_1, REK, and SK_5 are unknown, the value of fb_2 cannot be solved. Also, because fb_2, REK, and SK_5 are unknown, the value of fb_3 cannot be obtained, and so on.

Therefore, the variables $fb_1, fb_2, \ldots, fb_i, \ldots, fb_n$, form a secure internal feedback-code sequence which is unattainable by hackers. By substituting Eq. (8) with the above results, even if the attacker knows C_i and C_{i-1}, under the condition that fb_{i-1}, REK, and SK_5 are unknown, they cannot reversely derive values of $(P_i \odot_R fb_{i-1})$, $(REK \oplus fb_{i-1})$ *and* $((C_{i-1} \oplus SK_5) +_2 fb_{i-1})$. Namely, the chance of obtaining values of $(P_i \odot_R fb_{i-1})$, $(REK \oplus fb_{i-1})$ *and* $((C_{i-1} \oplus SK_5) +_2 fb_{i-1})$ on the basis in which C_i is known is the same to that of a blind guess [1]. In other words, the dynamic random encryption key REK hidden in the term $(REK \oplus fb_{i-1})$ is secure, and the probability with which to solve REK is $\frac{1}{2^n}$, which is the same as that of a blind guess.

Thirdly, without knowing the values of SK_2, SK_4, SK_5 and REK, hackers cannot obtain $\Delta_2 l$ by solving Eq. (9). Further, without the value of $\Delta_1 l$, hackers cannot reversely derive the value of $\Delta_2 l$ from the total length of the wrapped ciphertext file, i.e., $\Delta_2 l$ is secure. Even though hackers correctly guess the value of $\Delta_2 l$, and try to solve Eq. (9), i.e., $\Delta_2 l = ((SK_2 +_2 REK) \odot_R SK_5 +_2 (SK_4 \oplus REK))\ mod\ 1022 + 3$, to crack REK, the reality is that generation of a $\Delta_2 l$ involves a dynamic encryption key REK. On each generation, the value of $\Delta_2 l$ varies. The value of $\Delta_2 l$ ranges between 3 and 1024, in which 1024 is far smaller than the upper bound of REK ($0 \leqq REK \leqq 2^{128}-1$), plus the fact that hackers do not know the values of $(SK_2 +_2 REK)$, SK_5, and $(SK_4 \oplus REK)$. Thus, the probability with which to obtain REK based on mere value of $\Delta_2 l$ is $\frac{1}{2^n}$, which is the same to that of a blind guess (Q.E.D.).

4.4 Security on a Wrapped Ciphertext File

The ciphertext shown in Fig. 1 is wrapped by $PRS1$ of length $\Delta_1 l$ and $PRS2$ of length $\Delta_2 l$. Since values of $PRS1$, $CREK$, and $PRS2$ are random in different sessions, no methods that can be used to identify each of them in this wrapped ciphertext file. In operation mode 1 or 2, hackers cannot realize the length of ciphertext portion. So, they need to know the values of $\Delta_1 l$ and $\Delta_2 l$ to identify the positions of $PRS1$, $CREK$, and $PRS2$ to acquire the ciphertext. But, as mentioned above, $\Delta_1 l$ and $\Delta_2 l$ are well protected. Hackers cannot identify the exact location of ciphertext, thus unable to access it. Here, we dare to conclude that the security level of a wrapped ciphertext file is high.

4.5 Security on Ciphertext

Assume that hackers, by some method, correctly retrieve the ciphertext from the wrapped ciphertext file. Theorem 2 proves that the plaintext is secure.

Theorem 2:

Let $P_1P_2P_3 \ldots P_m$ be the plaintext, and let $C_1C_2C_3 \ldots C_m$ be the generated ciphertext, where P_i is the i^{th} plaintext block, C_i is its corresponding ciphertext block and both are n bits in length, $1 \leqq i \leqq m$. In operation mode 1 or 2, the probability with which to acquire plaintext $P_1P_2P_3 \ldots P_m$ based on illegally intercepted ciphertext $C_1C_2C_3 \ldots C_m$ is $\left(\frac{1}{2^n}\right)^m$.

Proof: Eq. (11), i.e., $P_i = (C_i \oplus ((C_{i-1} \oplus SK_5) +_2 fb_{i-1})) - 2(REK \oplus fb_{i-1}) \odot_{IR} fb_{i-1}$, indicates that C_{i-1}, SK_5, fb_{i-1}, and REK on the righthand side are required before P_i can be recovered from C_i. Then by Claimed 1 and Theorem 1, SK_5 and REK have achieved their practical securities. The value of $fb_{i\text{-}1}$ can be obtained with the help of Eq. (12), i.e., $fb_i = (P_i \odot_R fb_{i-1}) +_2 ((C_{i-1} \oplus SK_5) +_2 fb_{i-1})$, in which values of P_{i-1}, fb_{i-2}, C_{i-2} and SK_5 are necessary. However, the plaintext block P_{i-1} is hidden from hackers, and both $fb_0 = SK_4$ and $C_0 = SK_3$ are well protected. So $fb_1 = (P_1 \odot_R fb_0) +_2 ((C_0 \oplus SK_5) +_2 fb_0)$ is also safely protected. Similarly, if the hackers cannot solve P_2, fb_1 and SK_5, the value of $fb_2 = (P_2 \odot_R fb_1) +_2 ((C_1 \oplus SK_5) +_2 fb_1)$ is still unknown, and so on, meaning that the internal feedback-code sequence (fb_{i-1}, $1 \leqq i \leqq n$) is well protected. Substituting Eq. (11) with this result will show that $P_1 = (C_1 \oplus ((C_0 \oplus SK_5) +_2 fb_0)) -_2 (REK \oplus fb_0) \odot_{IR} fb_0$ is secure. Thus, when C_0, SK_5, fb_0 and REK are unknown, the probability with which to break P_1 is the same as that of a blind guess, i.e., $\frac{1}{2^n}$ where P_1 is protected by using the Two-dimension operation [1]. Likewise, $P_2 = (C_2 \oplus ((C_1 \oplus SK_5) +_2 fb_1)) -_2 (REK \oplus fb_1) \odot_{IR} fb_1$ is secure when SK_5, fb_1 and REK are unknown. The probability with which to break P_2 protected by the Three-dimensional operation [1] is also $\frac{1}{2^n}$, and so on. Hence, a plaintext block P_i, $1 \leqq i \leqq m$, is safely protected, and the probability with which to solve an individual plaintext block is $\frac{1}{2^n}$. According to Rule of Product, the probability with which to crack the plaintext $P_1P_2P_3 \ldots P_m$ is $\left(\frac{1}{2^n}\right)^m$ (Q.E.D.).

4.6 Security on the WEBDR against Eavesdropping Attacks

In operation mode 1 or 2, active brute-force attacks, like the chosen-plaintext attack and known-plaintext attack, cannot successfully crack a system protected by the WEBDR. Therefore, passive eavesdropping attacks will be the main method used to break the WEBDR by hackers. Now we would like to prove that the WEBDR can effectively defend against eavesdropping attacks.

In operation mode 1, before data files are transmitted between UE and a base station (or a cloud system), both sides of the connection have already owned their channel key, i.e., SK_{CH}, which is used to protect the data files. In fact, without the value of $\Delta_1 l$, the length of the plaintext and the value of $\Delta_2 l$, hackers cannot exactly identify the position of the ciphertext and then crack it. In addition, if hackers attempt to sniff data in a long term so as to collect a large amount of data for further analysis, it is still useless since for each communication session, the channel key SK_{CH} varies and there is no association between two arbitrary SK_{CH^s}. Of course, there is no direct relationship among all wrapped ciphertext files. In other words, the WEBDR can effectively defend against eavesdropping attacks when operation mode 1 is in use.

In operation mode 2, even if a wrapped ciphertext file is stolen. As mentioned above, hackers cannot figure out the right position of the ciphertext, and then crack the wrapped ciphertext file. Nevertheless, even though many wrapped ciphertext files are encrypted by using the same SK_{PW}, their IVs are different so that $SK_1 \sim SK_5$, $\Delta_1 l$and REK are all individually different in different sessions. It is hard for hackers to crack these wrapped ciphertext files without knowing SK_{PW}, meaning that the WEBDR is able to effectively defend against eavesdropping attacks when operation mode 2 is in use.

5 Performance Analyses and Improvements

The performance of encrypting and decrypting data blocks mainly depends upon the number of operating instructions. Table 1 lists the number of operations required by the WEBDR and AES when they encrypt/decrypt data blocks that are 128 bits long.

Table 1: The number of operations consumed by the WEBDR and AES

Method	Operation count on encryption	Operation count on decryption
AES (128-bit, 10-rounds operation mode)	AddRoundKey: 176 ⊕ s (8 bits)	The operation counts for AddRoundKey, SubBytes, and ShiftRows are the same as those in encryption
	SubBytes: 160 Substitutions (8 bits)	
	ShiftRows: 30 ShiftRows (128 bits)	(MixColumns) 36 Rijndael columns mixing (128 bits) (Generally, decryption is often more complex than encryption)
	MixColumns: 36 Rijndael columns mixing (128 bits)	
WEBDR (128-bit)	$3 \oplus s$, $3 +_2 s$, and $1 \odot_R$	$3 \oplus s$, $2 +_2 s$, $1 -_2$, $1 \odot_R$, and $1 \odot_{IR}$

Due to the natural-randomness property, it is difficult for most cryptographic algorithms to theoretically compare time complexity. To demonstrate the better performance of the WEBDR than the AES, we conduct several experiments with test scenarios that encrypt/decrypt data blocks of different sizes from 1 KBs to billion KBs. In each test case, we calculate the average time consumed by each step of pre-processing procedures and the encryption/decryption step by million times of executions.

The experimental results by employing devices of different specifications are shown in Table 2 which shows that the cost of encrypting (decrypting) a plaintext (ciphertext) block by the AES is 6–8 times higher than that of encrypting (decrypting) a plaintext (ciphertext) block by the WEBDR in average. Since before encryption, the AES needs to execute Key-Expansion, i.e., generating round-keys by manipulating its cipher-key. Similarly, before encryption, the WEBDR has to perform preprocessing, including initial process, Step 1 of the encryption process and the generation of $\Delta_2 l$.

Table 2: Experimental results of test cases (μs)

Dev#	Spec.	Method	Pre-proc	Encr	Decr	Encr+Decr	Times (AES/WEBDR)
1	CPU: I5-6198	AES	0.423	0.493	0.507	1	7.09 (=1/0.141)
	RAM: 4G	WEBDR	3.470	0.065	0.076	0.141	
2	CPU: I7-8750	AES	0.276	0.310	0.326	0.636	6.84 (=0.636/0.093)
	RAM: 8G	WEBDR	2.033	0.043	0.050	0.093	
3	CPU: I7-8700	AES	0.250	0.280	0.286	0.566	6.9 (=0.566/0.082)
	RAM: 8G	WEBDR	1.891	0.038	0.044	0.082	
4	CPU: I5-3570	AES	0.343	0.448	0.456	0.904	6.7 (=0.904/0.135)
	RAM: 12G	WEBDR	2.805	0.063	0.072	0.135	
5	CPU: I5-7200	AES	0.373	0.475	0.477	0.952	7.05 (=0.925/0.135)
	RAM: 12G	WEBDR	3.012	0.063	0.072	0.135	

To produce a wrapped ciphertext file, the WEBDR should execute post-processing procedures, i.e., the generation of $PRS1$ and $PRS2$, in Step 3 of the encryption process. The costs of extra sub-operations required by the AES and WEBDR are also listed in Table 2. The costs for pre/post-processing in the decryption process of the WEBDR are lower than that in its encryption process since the decryption does not need to generate $PRS1$ and $PRS2$, only identifying their lengths. If a plaintext file has n plaintext blocks, each of which is 16 bytes long, the theoretical encryption/decryption costs of the WEBDR and AES can be derived from Table 2.

(1) The AES (for Device# = 5)

$$\text{The encryption cost} = (\text{cost for generating sub-keys}) + (\text{cost for encrypting a plaintext block}) * n$$
$$= 0.373 + 0.475 * n\,(\mu s) \tag{13}$$

$$\text{The decryption cost} = 0.373 + 0.477 * n\,(\mu s) \tag{14}$$

(2) The WEBDR (for Device# = 5)

$$\text{The encryption cost} = (\text{cost of pre/post-processing}) + (\text{cost of encrypting a plaintext block}) * n$$
$$= 3.012 + 0.063 * n\,(\mu s) \tag{15}$$

$$\text{The decryption cost} = 3.012 + 0.072 * n(\mu\text{s}) \tag{16}$$

Basically, most of the 5th-generation (5G) applications are data intensive and at least 100 kb of data size. As shown in Table 2, the performance of the WEBDR is around 6.7–7.09 times faster than that of the AES.

The cost of wrapping n-block ciphertext, denoted by CC, in a wrapped file is

$$\text{CC} = \frac{16 + \Delta_1 l + 16 + 16\text{n} + \Delta_2 l}{16\text{n}} \tag{17}$$

$$= 1 + \left(\frac{2}{n} + \frac{\Delta_1 l + \Delta_2 l}{16n}\right) \leq 1 + \frac{130}{n} \tag{18}$$

where $130 = (\Delta_1 l_{max} + \Delta_2 l_{max})/16 + 2 = 1024 * 2/16 + 2$ since $\Delta_1 l_{max} = \Delta_2 l_{max} = 1024$. When n is large, CC approaches 1.

6 Conclusions and Future Works

In this study, the WEBDR is developed by using a randomly wrapped feedback approach based on user passwords or channel keys, which together with IV construct high security wrapped ciphertext files with high performance. When receiving a plaintext at different time points, the dynamic random encryption approach, which adopts current time keys and random keys, will produce different wrapped ciphertext files of different cipher texts and lengths, consequently highly improving the security level of transmitted ciphertext. Our theoretical analyses demonstrate that the WEBDR has achieved practical security in transmitting wireless data and encrypting personal files.

Theorems 1 and 2 prove the security level that the proposed scheme can achieve, i.e., the probability with which to obtain REK from an intercepted wrapped ciphertext file is $\frac{1}{2^n}$ and the probability with which to acquire plaintext $P_1P_2P_3 \ldots P_m$ based on illegally intercepted ciphertext $C_1C_2C_3 \ldots C_m$ is $\left(\frac{1}{2^n}\right)^m$. The performance of the WEBDR when encrypting/decrypting a file longer than 128kb is around 6-8 times faster than that of the AES (see Table 2). All operations required by AES and the WEBDR are listed in Table 1. The former consumes 176 XOR operations for both of its encryption and decryption, while the WEBDR costs only three XOR for each of its message decryption and decryption processes. Therefore, this proposed system is more suitable than AES for protecting data stored in a cloud or transmitted between the cloud and an end user. Of course, readers may say that less operations also easily conduct hackers to break the WEBDR. Yes, it is true. But the time consumed for encrypting/decrypting data for 5G/beyond 5G (B5G)/the 6th generation (6G) networks need to be short to avoid being the bottleneck of data transfer since users of current networks request short transmission time.

According to reference [34], the download speed of a 5G system is about 10 times that of a 4th generation (4G) network, and high-speed communication has been widely requested by users, high-performance transmission is always desired, while keeping the practical security.

In the future, we will continue developing a faster encryption and decryption approach and then apply it to image cryptography [35]. Also, users may forget their passwords. Then they have trouble decrypting their ciphertexts to plaintexts. Therefore, we need a forgotten-password-recovery mechanism following which users can recover their original passwords, and then decrypt the wrapped ciphertext files. These constitute our future studies.

Acknowledgement: We would like to thank reviewers who gave us many valuable comments and suggestions with which we can then significantly improve the quality of our manuscript.

Funding Statement: This work is financially supported in part by Ministry of Science and Technology (MOST), Taiwan under the Grant MOST 109-2221-E-029-017-MY2.

Author Contributions: Study concepts and system design: Yi-Li Huang and Fang-Yie Leu; Data collection and preparation: Ruey-Kai Sheu and Chi-Jan Huang; Draft manuscript preparation: Yi-Li Huang and Fang-Yie Leu; Analysis and interpretation of results: Ruey-Kai Sheu and Jung-Chun Liu; Theorem derivation and proofs: Yi-Li Huang and Jung-Chun Liu.

Availability of Data and Materials: A part of the data adopted in this study is articles randomly collected from the Internet. The remaining part is a company's personnel data. For privacy consideration, the personnel data cannot be accessed without this company's permission.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] Y. L. Huang, C. R. Dai, F. Y. Leu and I. You, "A secure data encryption method employing a sequential-logic style mechanism for a cloud system," *International Journal of Web and Grid Services*, vol. 11, no. 1, pp. 102–124, 2015.
[2] Kasperski, "Brute force attack: Definition and examples," 2023. [Online]. Available: https://www.kaspersky.com/resource-center/definitions/brute-force-attack
[3] Fortinet, "What is a brute force attack?," 2023. [Online]. Available: https://www.fortinet.com/resources/cyberglossary/brute-force-attack
[4] M. Al-Mhiqani, R. Ahmad, Z. Z. Abidin, K. H. Abdulkareem, M. A. Mohammed *et al.,* "A new intelligent multilayer framework for insider threat detection," *Computers & Electrical Engineering*, vol. 97, no. 1, pp. 107597, 2022.
[5] Wikipedia, "The EFF DES cracker," 2023. [Online]. Available: http://en.wikipedia.org/wiki/EFF_DES_cracker
[6] M. Kirschenbaum, "A practical guide for cracking AES-128 encrypted firmware updates," *Hypoxic Extreme Electronics*, 2020. https://gethypoxic.com/blogs/technical/a-practical-guide-for-cracking-aes-128-encrypted-firmware-updates
[7] Y. L. Huang, F. Y. Leu, I. You, H. C. Chen, C. S. Liaw *et al.,* "Random cladding with feedback mechanism for encrypting mobile messages," in *Proc. INFOCOM WKSHPS*, San Francisco, CA, USA, pp. 970–975, 2016.
[8] Wikipedia, "Advanced encryption standard," 2023. [Online]. Available: https://en.wikipedia.org/wiki/Advanced_Encryption_Standard
[9] W. Diehl, "Attack on AES implementation exploiting publicly-visible partial result," *Cryptology ePrint Archive*, 2017.
[10] Wikipedia, "Biclique attack," 2023. [Online]. Available: https://en.wikipedia.org/wiki/Biclique_attack
[11] Tutorialspoint, "What are the types of Cryptanalysis Attacks on AES in information security?," 2023. [Online]. Available: https://www.tutorialspoint.com/what-are-the-types-of-cryptanalysis-attacks-on-aes-in-information-security

[12] Checkpoint, “What is cloud security?,” 2022. [Online]. Available: https://www.checkpoint.com/cyber-hub/cloud-security/what-is-cloud-security/
[13] Wikipedia, “Cloud security,” 2023. [Online]. Available: http://en.wikipedia.org/wiki/Cloud_computing_security
[14] L. Bordak, “Cloud computing security,” in *Proc. of ICETA*, Startfytf Smokovec, Slovakia, pp. 87–92, 2019.
[15] A. Musa and A. Mahmood, “Client-side cryptography based security for cloud computing system,” in *Proc. of ICAIS*, Tamil Nadu, India, pp. 594–600, 2021.
[16] L. H. A. Reis, M. T. de Oliveira, J. Bowden, D. Krefting, S. D. Olabarriaga *et al.,* “Cryptography on untrustworthy cloud storage for healthcare applications: A performance analysis,” in *Proc. of SBESC*, Online, pp. 1–8, 2021.
[17] O. Banuelos, “Encryption key management and its role in modern data privacy,” *SkyFlow*, 2022. [Online]. Available: https://www.skyflow.com/post/encryption-key-management-and-its-role-in-modern-data-privacy?utm_source=google&utm_medium=ppc&utm_campaign=blog&utm_term=encryption%20key&utm_campaign=All+Tiers:+%27How+To%27+Campaign&utm_source=adwords&utm_medium=ppc&hsa_acc=6575335991&hsa_cam=13968847646&hsa_grp=141732157609&hsa_ad=610758177074&hsa_src=s&hsa_tgt=kwd-297166611545&hsa_kw=encryption%20key&hsa_mt=b&hsa_net=adwords&hsa_ver=3&gclid=CjwKCAjwg5uZBhATEiwAhhRLHodD_Y57v3jHwkTHdCbasltaSHnAl5y5g_GVnzwmQYxp6uMUvrKUkxoCDJYQAvD_BwE
[18] Software AG, “What is an IoT security solution?,” 2023. [Online]. Available: https://www.softwareag.com/en_corporate/resources/what-is/iot-security-solution.html
[19] B. Schacht and P. Kieseberg, “An analysis of 5 million OpenPGP keys,” *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA)*, vol. 11, no. 3, pp. 107–140, 2020.
[20] J. Roundy, “IoT security: IoT device security challenges and solutions,” 2023. [Online]. Available: https://www.verizon.com/business/resources/articles/iot-device-security-challenges-and-solutions/
[21] J. Yang, L. Wang and S. Shakya, “Modelling network traffic and exploiting encrypted packets to detect stepping-stone intrusions,” *Journal of Internet Services and Information Security (JISIS)*, vol. 12, no. 1, pp. 2–25, 2022.
[22] S. Nowaczewski and W. Mazurczyk, “Searching future Internet and 5G using customer edge switching using DNSCrypt and DNSSec,” *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA)*, vol. 11, no. 3, pp. 87–106, 2020.
[23] Cloud Security Alliance (CSA), “Cloud security alliance, top threats to cloud computing,” 2023. [Online]. Available: http://www.cloudsecurityalliance.org
[24] F. Chen, K. Wu, W. Chen and Q. Zhang, “The research and implementation of the VPN gateway based on SSL,” in *Proc. of ICCIS*, Shiyang, China, pp. 1376–1379, 2013.
[25] D. Clinton, “How website encryption works,” ENCRYPTION, 2023. [Online]. Available: https://www.freecodecamp.org/news/understanding-website-encryption/
[26] Wikipedia, “Proxy server,” 2023. [Online]. Available: http://zh.wikipedia.org/wiki/%E4%BB%A3%E7%90%86%E6%9C%8D%E5%8A%A1%E5%99%A8
[27] V. D. Chakravarthy, K. L. N. C. Prakash, K. Ramana and T. R. Gadekallu, “A novel DDOS attack detection and prevention using DSA-DPI method,” in *Proc. of ICICC*, Delhi, India, pp. 733–744, 2022.
[28] S. Daneshgadeh, T. Ahmed, T. Kemmerich and N. Baykal, “Detection of DDoS attacks and flash events using shannon entropy, KOAD and mahalanobis distance,” in *Proc. of ICIB*, Thessaloniki, Greece, pp. 222–229, 2019.
[29] K. L. Tsai, L. W. Chen, F. Y. Leu and C. T. Wu, “Two-stage high-efficiency encryption key update scheme for LoRaWAN based IoT environment,” *Computers, Materials & Continua*, vol. 73, no. 1, pp. 547–562, 2022.
[30] N. Khan, J. Zhang, U. Intikhab, S. M. S. Pathan and H. Lim, “Lattice-based authentication scheme to prevent quantum attack in public cloud environment,” *Computers, Materials & Continua*, vol. 75, no. 1, pp. 35–49, 2023.

[31] O. I. Khalaf, M. Sokiyna, Y. Alotaibi, A. Alsufyani and S. Alghamdi, "Web attack detection using the input validation method: DPDA theory," *Computers, Materials & Continua*, vol. 68, no. 3, pp. 3167–3184, 2021.
[32] Y. Yang, W. Wang, R. Xu, G. Srivastava, M. Alazab *et al.,* "AoI optimization for UAV-aided MEC networks under channel access attacks: A game theoretic viewpoint," in *Proc. of ICC*, Seoul, South Korea, pp. 1–6, 2022.
[33] W. Wang, G. Srivastava, J. C. W. Lin, Y. Yang, M. Alazab *et al.,* "Data freshness optimization under CAA in the UAV-Aided MECN: A potential game perspective," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–10, 2022. https://doi.org/10.1109/TITS.2022.3167485
[34] M. V. Nichita, P. Ciotîrnae, R. L. Luca and V. N. Petrescu, "5G propagation: Current solutions and future proposals," in *Proc. of ISETC*, Timisoara, Romania, pp. 47–50, 2016.
[35] E. Bashier and T. Ben Jabeur, "An efficient secure image encryption algorithm based on total shuffling, integer chaotic maps and median filter," *Journal of Internet Services and Information Security (JISIS)*, vol. 11, no. 2, pp. 64–79, 2021.

Appendix A: Abbreviation table for parameters

No.	Param.	Description
1	IV	initialization vector, which is input to a cryptographic primitive by users to provide the initial state of the WEBDR.
2	PW	the password, comprising 8 to 32 characters, is prepared as one of the inputs by users.
3	SK_{PW}	the system password key derived from PW.
4	dsc	dynamically shifting count when shifting data.
5	SK_{CH}	the system channel key, created for a user and the cloud sever before their communication starts.
6	SK_0	the system zeroth encryption key defined as $SK_0 = SK_{PW}$ or $SK_0 = SK_{CH}$.
7	IEK	the initial encryption key.
8	$SK_1 \sim SK_5$	the system sub-keys produced in the system's initial procedure.
9	$PRS1$	Pseudo-random sequence 1, as a random string placed at the beginning of a wrapped ciphertext file.
10	$PRS2$	pseudo-random sequence 2, as a random string placed at the end of a wrapped ciphertext file.
11	$\Delta_1 l$	$\|PRS1\|$ in bytes.
12	$\Delta_2 l$	$\|PRS2\|$ in bytes.
13	SK_{CT}	the system time key, generated according to current CPU time, is 128 bits long comprising the following elements: nanosecond/date/hour/minute/second/nanosecond/hour/minute/second.
14	SK_{RCT}	1. The reverse key of SK_{CT}, 128 bits long, consists of the following elements: second/minute/hour/nanosecond/second/minute/ hour/date/nanosecond.
15	REK	Random Encryption Key, which is employed to generate ciphertexts and the length of $PRS2$.
16	$CREK$	the Ciphertext key of REK.
17	$fb_0 \sim fb_n$	a sequence of internal feedback code.
18	Plaintext blocks	$P_1P_2P_3 \ldots P_n$, where P_j is plaintext block j and $\|P_j\| = 128$ bits, $1 \leqq j \leqq n$.

(Continued)

Appendix A (continued)

No.	Param.	Description
19	Ciphertext blocks	$C_1C_2C_3\ldots C_n$, where C_j is ciphertext block j and $\|C_j\| = 128$ bits, $1 \leqq j \leqq n$.

Appendix B: Table for summarized operator

No.	Operator	Description
1	XOR, denoted by $\bigoplus$	Encrypting plaintext p to ciphertext c with key k, i.e., $c = p \bigoplus k$. Decrypting c to p with k, i.e., $p = c \bigoplus k$.
2	Binary adder [26] denoted by $+_2$	Encrypting plaintext p to ciphertext c with key k, i.e., $c = p +_2 k$, in which we drop the carry generated by the addition of the most significant bit Decrypting c to p with k, i.e., $p = c -_2 k = \begin{cases} c - k, & \textit{if } c \geq k \\ c + \overline{k} + 1, & \textit{if } c < k \end{cases}$, in which. $-_2$ is the inverse operation of $+_2$.
3	Rotate-Equivalence operator denoted by $\odot_{\mathrm{R}}$	Encrypting plaintext p_i to ciphertext c_i with key k, i.e., $c_i = p_i \odot_R k = p_{IR} \odot k$, where $p_{i\mathrm{R}}$ is the key obtained by rotating plaintext p_i clockwise h bits where $h = \|k\|/4$, e.g., if $\|k\| = 128$, p_i will be rotated 32 bits. Decrypting c_i to p_i with k, i.e., $p_i = c_i \odot_{IR} k =$ counterclockwise rotating $(c_i \odot k)$ a total of $\|k\|/4$ bits.
4	Three-dimensional operation	the operation that encrypts a message by using encryption keys and three fundamental operators [8], i.e., $\bigoplus$, $+_2$ and $\odot_{\mathrm{R}}$.
5	1. Modulus operator: mod	$c = p \bmod n$, where$n$ is a positive integer.
6	*Left* (PW, n)	a function that retrieves n leftmost characters from PW, where $n \leqq \|PW\|$ in bytes.
7	*Right* (PW, n)	a function that accesses n rightmost characters from PW, where $n \leqq \|PW\|$ in bytes.
8	*trunc* (RN, t)	a function that truncates the rightmost t bytes from the random number key RN.