applied sciences

MDPI

Article

Analysis of Vulnerabilities That Can Occur When Generating One-Time Password

Hyunki Kim [1], Juhong Han [1], Chanil Park [2] and Okyeon Yi [1,*,†]

1 Financial Information Security, Kookmin University, 77 Jeongneung-RO, Seongbuk-GU, Seoul 02707, Korea; lplp456@kookmin.ac.kr (H.K.); hjhong@kookmin.ac.kr (J.H.)

2 Agency for Defence Development, Daejeon 34186, Korea; chanil@add.re.kr

* Correspondence: oyyi@kookmin.ac.kr; Tel.: +82-02-910-5120

† These authors contributed equally to this work.

Received: 18 March 2020; Accepted: 22 April 2020; Published: 24 April 2020

Featured Application: Cryptography; Random Number analysis.

Abstract: A one-time password (OTP) is a password that is valid for only one login session or transaction, in IT systems or digital devices. This is one of the human-centered security services and is commonly used for multi-factor authentication. This is very similar to generating pseudo-random bit streams in cryptography. However, it is only part of what is used as OTP in the bit stream. Therefore, the OTP mechanism requires an algorithm to extract portions. It is also necessary to convert hexadecimal to decimal so that the values of the bit strings are familiar to human. In this paper, we classify three algorithms for extracting the final data from the pseudo random bit sequence. We also analyze the fact that a vulnerability occurs during the extraction process, resulting in a high frequency of certain numbers; even if cryptographically secure generation algorithms are used.

Keywords: one-time password; random number; extraction algorithm

1. Introduction

In general, fixed passwords are vulnerable to re-use attacks due to malicious attacker's password collection. One-Time Password (OTP), on the other hand, is a password that is used differently each time for a login session or transaction [1]. Thus, this makes it impossible to reuse previously collected things. It also has the property of randomness, which makes it mathematically impossible to infer the next password from the current one. It is known to be highly secure because of these features, and it can be used as an authentication service in Information & Communication Technology (ICT) [2] fields such as mobile-payment [3,4] and mobile-cloud computing [5,6].

OTP mechanism generates passwords using secrets shared in advance between the certification authority and the user, and time or event occurrence information. So the attacker can not infer this by himself. Currently, it is provided as a token, card type device (Hardware), and mobile phone application type (Software) [7,8]. Hardware type is frequently used in the financial sector, and the number of issuances and transactions of this is continuously increasing. Software in mobile phones is widely used in online games and IT industries, but financial institutions are developing mobile OTP based on USIM.

2. Background

2.1. Random Number

2.1.1. Randomness

The random number sequence is the same as the result of a series of unbiased "fair" coins. Each side of the coin is 0 or 1, the value of the bit [9]. If it is thrown in succession, 0 and 1 are identically distributed and each trial runs independently. Thus it is equivalent to the result of a perfect random number generator. All parts of the random sequence are independent of each other, so the parts that have already been generated can not affect the next time and can not be predicted [10].

2.1.2. Unpredictability

The (pseudo) random numbers used in cryptography should be unpredictable. In the case of Cryptographically secure Pseudo-Random Number Generators (CPRNG), if the attacker does not know the seed, the output must be unpredictable, and he must not know the input value only by looking at the generated random number sequence (providing forward/backward unpredictability) [9,10]. In general, algorithms for random number generators are public, so the seed must be kept secret to provide unpredictability of the output random number.

2.1.3. Pseudo Random Number Generator (PRNG)

PRNG generates a large number of pseudo random sequences through one or more inputs. Its input value is called seed, and it is generated through a true random number generator (TRNG) because it must provide unpredictability and randomness. Therefore, PRNG is usually used together with TRNG because it uses the output of TRNG as seed [11,12].

Since the PRNG mitigates the bias, statistical characteristics, and correlations of the seeds, the pseudo random number sequence is more random than the true random number sequence. This generally uses cryptographic algorithms and is deterministic.

2.2. One-Time Password (OTP)

2.2.1. OTP Authentication Type

Synchronization type: This is an authentication technique in which the input value of the algorithm is kept the same by the user token and the authentication server [13]. This can be classified into time synchronization [14], event synchronization, time and event combination synchronization, and so on [15].

Non-synchronization method: In this method, the authentication process proceeds without any synchronization information established between the user token and the authentication server. A representative example is the challenge-response method.

2.2.2. OTP Token

This means a dedicated hardware or a software module installed in a mobile phone or the like for OTP. Dedicated hardware tokens have a built-in display for viewing disposable values, and a built-in numeric keypad for PIN entry (optional) [16].

3. Related Works

3.1. Vulnerabilities of OTP on PC

The OTP implemented as S/W on the PC can grasp the actual operation process through reverse engineering. By analyzing this, it is possible to obtain the master key and time information in the OTP, and it is possible to forge/modify the PIN, master key, and time information. If an attacker exploits this, it is possible to generate a valid OTP and even obtain the OTP value in the future. In order to eliminate such functional and logical vulnerabilities, it is required to apply various security reviews when implementing the OTP generation program mounted on the PC. In particular, anti-reversing technology must be applied so that the attacker(s) can not reverse the OTP program to easily perform security function bypass attacks [17].

3.2. Vulnerabilities of OTP in Internet Banking

Although OTP is one of the strongest authentication method to protect against account theft, there is always a possibility of account theft due to the evolution of hacking methods. The MITM (Man-in-the-Middle) attack accesses data transaction paths, and snatches information in the middle of data transactions. The MITB (Man-in-the-Browser) attack refers hacking methods which use malignant programs in the web browser. The MITPC (Man-in-the-PC) attack abuses the vulnerability of hardware environment or the operating system. It controls and attacks every data access path in the OS including the key input, mouse movement, screen, and even memories [18].

3.3. NIST Standards for Cryptographic Random Numbers

SP800-90 (A, B, C)

SP800-90A describes a PRNG that uses approved cryptographic algorithm. This makes it impossible to distinguish between an ideal random sequence and a pseudo random sequence without infinite computational power. It is designed with a fixed finite security strength to measure the workload required to attack PRNG. Ref. [19] SP800-90B describes mechanisms and entropy sources for obtaining unpredictable bits through non-deterministic processes. The entropy value of this source is equal to the Security Strength. Ref. [20] SP800-90C describes the overall RNG structure combining 90A and 90B, as shown in Figure 1 [12].

4. Materials and Methods

4.1. OTP Mechanism

OTP is generated following the procedure in Figure 1.

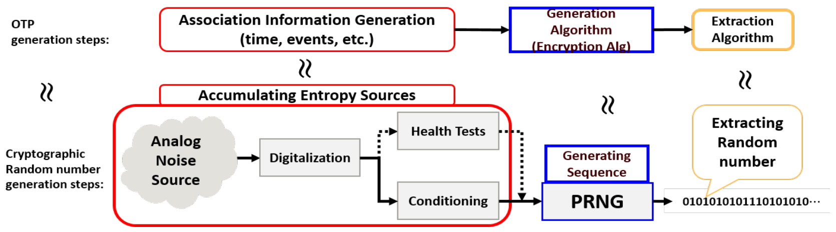


Figure 1. Three steps of the OTP generation.

The OTP generation process can be roughly classified into three steps [21]: 'Association information generation', 'generation algorithm', and 'extraction algorithm'. This is very similar to the process of generating pseudo random numbers used in cryptography [22]. Figure 1 shows similar parts of the two processes in the same color.

4.1.1. Association Information Generation

Association information means to a random value that can be collected from time, event information, and the like. This may be a raw value, depending on the collection method, or it may be a random number converted through additional processes such as hashing and encryption. Association information is similar to the noise source (or entropy source) needed to generate cryptographic pseudo random numbers and is necessary for the unpredictability of OTP. This process corresponds to the red part of Figure 1.

4.1.2. Generation Algorithm

This is a cryptographic algorithm used to generate OTP. This encrypts the association information to produce a 20-byte bit string [23]. The Korean OTP standard [24] introduces the cryptographic algorithm used in this step. Since the generation algorithm is deterministic, it generates the same ciphertext from the same association information. The cryptographic algorithms used should provide a security strength of at least 112 bits [21,23,25]. Different OTP values may be generated according to extraction algorithms in one same bit string. This process corresponds to the blue part of Figure 1.

4.1.3. Extraction Algorithm

This is an algorithm for extracting 3-byte data from the cipher string for use as OTP. This can be divided into static and dynamic algorithms (RFC4226 [23], an OTP-related standard, specifies only dynamic algorithms, and the Telecommunication Technology Association (TTA)'s standard [21] specifies static, dynamic, and improved dynamic algorithms.). This algorithm uses 'extraction information' to specify the location to extract. It also uses the value as the 'extraction data' at the location specified by the extraction information. The extraction information may use a value of a specific area of the cipher text or a predefined value. Then, the extracted 3-byte hexadecimal value is encoded in decimal so that humans can recognize it. At this time, the extracted 3-byte range is [0, 16777215] (i.e., $[0, 2^{24} - 1]$), but the top two digits have little change. Therefore, the two digits are discarded and only the lower six digits are used (Only $0 \sim 1$ for 10^7, only $0 \sim 6$ for 10^6); so the final range is [0, 999999]. This process corresponds to the yellow part of Figure 1.

The static algorithm extracts data by specifying in advance the extraction information (the offset of the cipher text) to be used [21].

An example is shown in Figure 2: Data is extracted by specifying the extraction information as the first starting point of the cipher text. It extracts 3-byte contiguous data from the extraction information. If the result of the extracted data is 0x1A2B3C, the number is 1,715,004 when expressed as a decimal number, and the final output OTP is derived as 715,004 corresponding to the lower 6 digits of the decimal number.

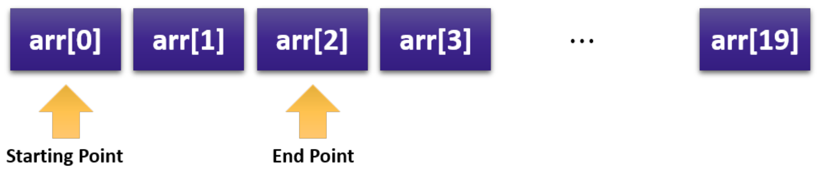


Figure 2. Static extraction.

The dynamic algorithm extracts data by obtaining extraction information from the cipher text [21,23].

About this example is shown in Figure 3: The value of the lower 4-bit of the lowest byte (i.e., 19th) in the cipher text is used as extraction information. The extracted data is contiguous 3-byte with an index of the extraction information as a starting point. If the result of the extracted data is 0xE7F809, that's decimal is 15,202,313, the final output OTP is derived as 202,313 corresponding to the lower 6 digits.

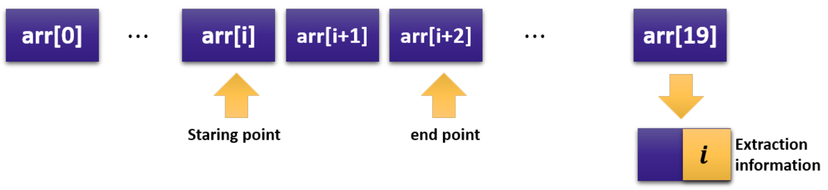


Figure 3. Dynamic extraction.

The improved dynamic algorithm is used when generating OTP using a chip used as a smart card or USIM [21]. Figure 4 shows an improved dynamic algorithm; it extracts three bytes from two or more pieces of extraction information in the cipher text.

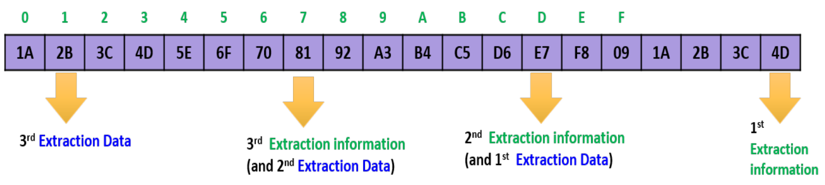

0 1 2 3 4 5 6 7 8 9 A B C D E F
1A 2B 3C 4D 5E 6F 70 81 92 A3 B4 C5 D6 E7 F8 09 1A 2B 3C 4D
3rd Extraction Data
3rd Extraction information (and 2nd Extraction Data)
2nd Extraction information (and 1st Extraction Data)
1st Extraction information

Figure 4. Improved dynamic extraction.

Extraction information of this algorithm is used to designate specific area of cipher text. If the value of the lower 4-bit of the lowest byte is used as the first extraction information (namely, LSB 4-bit 0x0D, the lowest byte 0x4D of the cipher text), then the data 0xE7 at that location (0x0D) is used as the first extraction data. And the second extraction information is 0x07, the value of the lower 4-bit of the first extraction data. The same method is repeated to select 3-byte extracted data. Final OTP is 171,883 which is the lower 6 digits of 15,171,883 that converts the extracted data 0xE7812B to decimal.

5. Analysis of OTP Vulnerabilities

5.1. Vulnerability in OTP Generation Algorithm

The pseudo random number generation algorithm specified in the TTA's standard "One-time password (OTP) encryption key management security requirements" [24] is shown in Table 1.

Table 1. Standard generation algorithms for OTP in Korea.

Classification	**Standard Algorithms That Specified**	
Hash based	DSA-PRNG-SHA1 ECDSA-PRNG-SHA11 HMAC-DRBG-SHA-256	Hash-DRBG-SHA1 KCDSA-PRNG-HAS1160
Block cipher based	ANSI X9.31 triple DES ANSI X9.31 AES ANSI X9.17 PRNG-triple DES CTR-DRBG-tiple-DES CTR-DRBG-tiple-AES-128,192,256	DSA-PRNG-ARIA DSA-PRNG-SEED ECDSA-PRNG-ARIA ECDSA-PRNG-SEED
etc.	Micali Schnorr-DRBG	

Table 2 shows the security strength defined by the NIST [25]. Currently, Korea's OTP standard recommends to provide more than 112-bit security strength as a cryptographic algorithm that can be used to generate OTP. As shown in Table 2, SHA1 has a security strength of 80-bit or less, so NIST prohibits its use for key derivation and random number generation. However, the TTA standard still includes SHA1, which is unsuitable for using OTP; since HAS-160 is also a hash function built on SHA1, its security level is similar to SHA1. Therefore it should not be used anymore. In addition, SHA1 and HAS-160 are currently excluded from the National Intelligence Service (NIS)'s approved algorithm list, as are AES and Triple-DES. In order to proceed with the procedure of generating and communicating random numbers between entities for the purpose of authentication, the NIS's Approved algorithm should be used, but this is not currently observed in the OTP standard.

Table 2. Hash functions that can be used to provide the targeted security strengths.

Security Strength	Digital Signatures and Hash-Only Applications	HMAC, Key-Derivation Functions, Random Number Generation
$\leq$80	SHA-1	
112	SHA-224, SHA-512/224, SHA3-224	
128	SHA-256, SHA-512/256, SHA3-356	SHA-1
192	SHA-384, SHA3-384	SHA-224, SHA-512/224
$\geq$256	SHA-512, SHA3-512	SHA-256, SHA-512/256, SHA-384, SHA-512, SHA3-512

5.2. Vulnerabilities in OTP Extraction

The improved dynamic extraction algorithms can be vulnerable to OTP generation. Figure 5 illustrates a situation where a vulnerability can occur in that case.

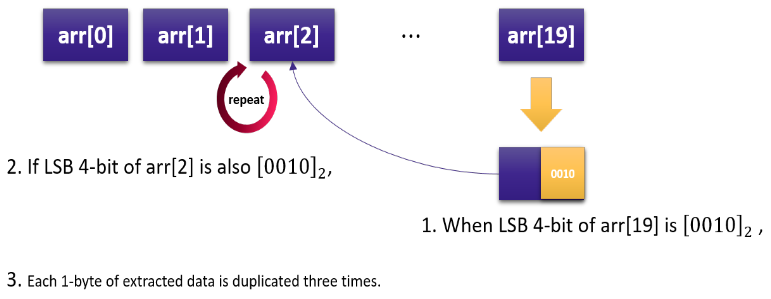

arr[0]
arr[1]
arr[2]
···
arr[19]
repeat
0010
2. If LSB 4-bit of arr[2] is also $[0010]_2$,
1. When LSB 4-bit of arr[19] is $[0010]_2$,
3. Each 1-byte of extracted data is duplicated three times.

Figure 5. Case for vulnerabilities in Improved Dynamic Extraction Algorithm.

The improved dynamic algorithm derives new extraction information from the location indicated by the value based on the predetermined extraction information. However, in this process, if the first extraction information and the next one are the same, all three bytes of the extraction data are duplicated. For example, when the least significant byte $arr[19]$ of the cipher text is 0x42, the extraction information is 0x02, which is LSB 4-bit, and the first extraction data is $arr[2]$. Then, LSB 4-bit of $arr[2]$ is the second extraction information. At this time, if the LSB 4-bit of $arr[2]$ is also 0x02, the second extracted data is $arr[2]$ again. The third and last process is the same. As a result, a total of three bytes are duplicated. Example output is $arr[2]||arr[2]||arr[2]$. If all the extracted data values are duplicated in the above situation, OTP range is a set of elements that is $arr[i]||arr[i]||arr[i]$ ($0x00 \leq arr[i] \leq 0xFF$, $[0, 255]$) form. If this reduced set is called R, then $|R|$ Becomes 256, so $|R|$ is reduced by approximately 99% compared to $[0, 999999]$ (i.e., $[0, 10^6 - 1]$), making it easier to predict the random numbers that can be output.

The following assumptions are made to analyze the probability that this situation can occur.

- Store the cipher text 160-bit in the array $arr[]$, which is 8-bit in size.
- For the integer $t \in [0, 19]$, the lower 4-bits of $arr[t]$ are expressed as $[arr[t]]_4$.
- If $[arr[19]]_4 = i$, then $arr[i]$ is used as the first extraction data.
- If $[arr[i]]_4 = j$, then $arr[j]$ is used as the second extraction data.
- If $[arr[j]]_4 = k$, then $arr[k]$ is used as the third extraction data.

In this case, the probability that a duplicate situation occurs in the extraction algorithm can be obtained as follows. Since the bit string is the output of the encryption algorithm, i.i.d is satisfied between all bits. That is, each bit is independent and the probability of getting 0 or 1 is $\frac{1}{2}$. Pr[All extraction information is duplicated] is

$$\begin{aligned}
&Pr[\text{output of extraction} = (arr[i]||arr[i]||arr[i])] \\
&= \sum_{i=0}^{15} Pr[[arr[19]]_4 = i, [arr[i]]_4 = i,, [arr[j]]_4 = i] \\
&= \sum_{i=0}^{15} Pr[[arr[19]]_4 = i, [arr[i]]_4 = i] \\
&= \sum_{i=0}^{15} Pr[([arr[19]]_4 = i) \times ([arr[i]]_4 = i)], (\because i.i.d) \\
&= \sum_{i=0}^{15} \frac{1}{16} \times \frac{1}{16} \\
&= 16 \times \frac{1}{16} \times \frac{1}{16} = \frac{1}{16}
\end{aligned} \tag{1}$$

In addition, the probability through the total number of cases can be obtained.

$$\begin{aligned}
&Pr[\text{All extraction information is duplicated}] \\
&= \frac{\text{the number of cases where the lower 4-bits of the two positions are the same}}{\text{the number of total cases of 160-bit string}} \\
&= \frac{2^{152} \times 2^4 \times 1}{2^{160}} = \frac{1}{16}
\end{aligned} \tag{2}$$

This means that whenever a random cipher text 20-byte (160-bit) is generated, there is a $\frac{1}{16}$ probability that the above duplication occurs, resulting in a vulnerability that reduces the size of the range (That is, the probability that the extracted information is duplicated is equal to the probability that the range is reduced.). Similarly, even if the data is duplicated only 2 times instead of 3 times, this process reduces the range and increases the predicted probability.

Moreover, when converting from extracted 3-bytes (24-bits) to 6-digit decimal number, the ideal probability that each appears as output in the range is as follows. Let $N \geq m \geq 1$ be integers, and let $(q, r) = IntDiv(N, m)$ (or $N = mq + r$). And let the random variable X be uniformly distributed over Z_N. For z in Z_m, let: $P_{\{N,m\}}(z) = Pr[X \text{ mod } m = z : X \text{ randomly picked in } Z_N]$ [23].

Then for any z in Z_m,

$$P_{\{N,m\}}(z) = \begin{cases} \frac{q+1}{N}, & \text{if } 0 \leq z < r \\ \frac{q}{N}, & \text{if } r \leq z < m \end{cases} \tag{3}$$

Similarly, $Pr[X \text{ mod } 10^6 = z$: X randomly picked in $Z_{2^{24}}]$ is

$$P_{\{2^{24},10^6\}}(z) = \begin{cases} \frac{16+1}{2^{24}}, & \text{if } 0 \leq z < 777216 \\ \frac{16}{2^{24}}, & \text{if } 777216 \leq z < 999999 \end{cases} \tag{4}$$

In other words, the ideal probability is approximately $1/2^{20}$. On the other hand, if the above-mentioned vulnerability occurs, the probability of each element in the set R is converted is as follows. $Pr[X \text{ mod } 2^8 = z$: X randomly picked in $Z_{2^{24}}]$ is

$$P_{\{2^{24},2^8\}}(z) = \begin{cases} \frac{2^{16}+1}{2^{24}}, & \text{if } 0 \leq z < 0 \\ \frac{2^{16}}{2^{24}}, & \text{if } 0 \leq z < 256 \end{cases} \approx \frac{1}{2^8} \tag{5}$$

Since there is no situation where the integer r satisfies $0 \leq r < 0$, it is only in the range $0 \leq z < 256$. In other words, the ideal probability is $\frac{1}{2^8}$. Therefore, if the above mentioned vulnerability occurs, then the probability of each number appears is about 4000 times higher than that of an ideal situation, which occurs in 1 out of 16 cases. As mentioned in the first part of this paper, OTP is used for authentication purposes in various ways, and is particularly widely used in online banking. If an attacker attempts to attack an online banking system that follows this vulnerability, the attacker can significantly reduce the number and time of OTP predictions. In general, an authentication system that performs user authentication through OTP defines Time-To-Live (TTL); this limits the OTP value to be valid only for a specific time. Therefore, in order to succeed in the attack, he must find a valid value within the timeframe. According to the above analysis, the probability of an attacker succeeding in prediction increases significantly from $\frac{1}{2^{20}}$ to $\frac{1}{2^8}$.

This extraction method is vulnerable in terms of unpredictability, although specified in the standard. The standard for OTP-related TTA [21] was issued in 2012, and even RFC 2289 [22], 4226 [23], and 1760 [1] were published between 1995 and 2005 and have not been updated to date. Standards for generating and applying cryptographic random numbers, such as NIST's SP800-90 series [12,19,20], have only been steadily revised in recent years. In 2015 and 2018, 90A and 90B were released as final versions, and 90C is still in draft. Thus, in the field, the period focused on various properties of cryptographic random numbers (predictability and bias, independence, entropy, etc.) is not long. And only recently has the trend to deal with the security of random numbers in depth. Therefore, looking at the OTP standards of RFCs and TTAs from a modern point of view can reveal vulnerabilities that have not been identified in the past.

To solve this problem, we suggest the following solutions.

- Use of static or dynamic algorithm: Rather than using an improved algorithm that can cause vulnerabilities, using a static or dynamic algorithm is suitable for information security by supplementing the randomness and unpredictability of OTP.
- Increases the number of LSB digits used as extraction information, and increases the length of the output bit string accordingly: Since 16 offsets can be selected using LSB 4-bit extraction information, the length of the output bit stream is used as 20-bytes $(= 16 + 4)$. If this is used as LSB 5-bit or 6-bit, the offset that can be selected increases to 32 or 64, so the length of the output bit string is 36-bytes$(= 32 + 4)$ and 68-bytes$(= 64 + 4)$. In this case, the probability of offset overlapping decreases by $\frac{1}{2}$ each time the bit is increased by one. (For example, Pr[All extraction information is duplicated] for 5-bit = $\frac{1}{32}$, Pr for 6-bit = $\frac{1}{64}$)
- Increase the amount of extraction to 4-byte: When the amount of extraction is increased from 3-byte to 4-byte, the range is [0, 16777215] (i.e., $[0, 2^{24} - 1]$) to [0, 4294967295] (i.e., $[0, 2^{32} - 1]$, About 4 billion cases). Therefore, OTP can be used up to 7 and 8 digits instead of 6 digits. In this case, since $[0, 10^7 - 1]$ and $[0, 10^8 - 1]$ can be used, the predictable probability decreases by $1/10$ each time by one digit increment.

6. Results and Discussion

6.1. Experiments

6.1.1. Experiment 1: Does This Extraction Vulnerability Really Happen?

To verify that this vulnerability actually occurs depending on the extraction algorithm, the following two experiments were conducted.

Figure 6 shows an experiment to check if duplicate 3-bytes actually occur frequently when different extraction algorithms are applied to the same bit string by encryption algorithm. The experiment proceeds as follows. In order to continuously obtain 3-bytes of different OTP data, an arbitrary bit sequence (2^{25}-bit, or 2^{22}-byte) is input to the block cipher AES [26], and a continuous cipher sequence is generated accordingly. Data that does not apply any extraction method to this string is called a pure cipher text sequence. In this bit string, each static/dynamic/improved algorithm is applied every 20-byte to extract 3-byte. It repeats this to accumulate 1,000,000-byte per algorithm.

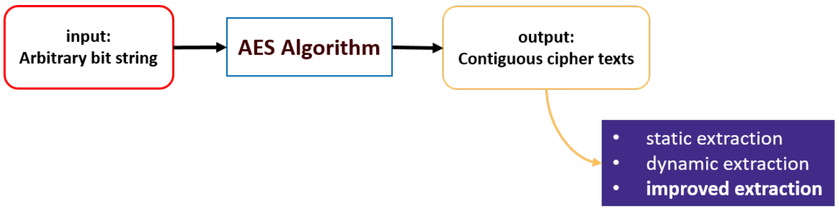
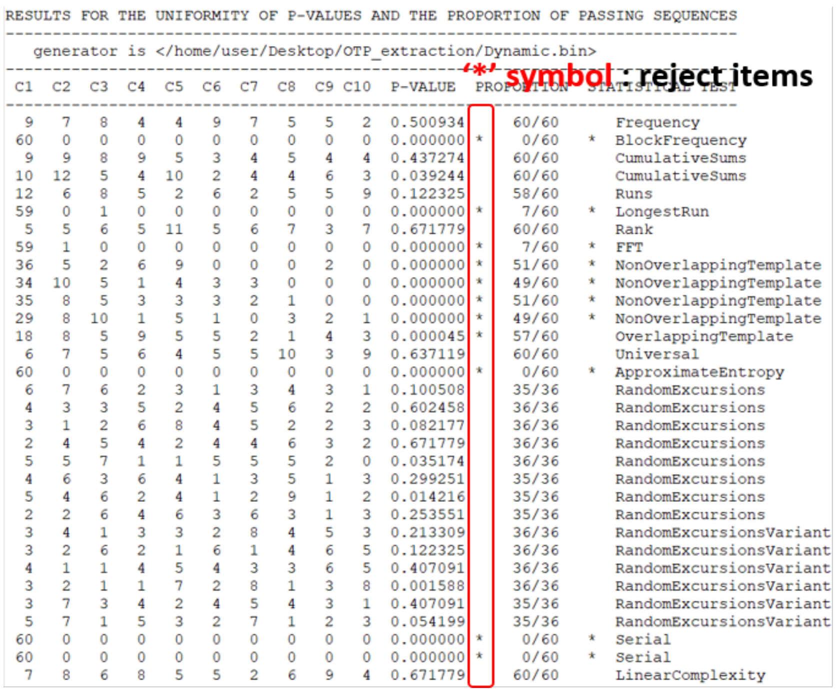


Figure 6. Experiment on whether extraction algorithm affects bit string.

And this was done by applying NIST's statistical randomness test [27] for each data set. The experimental group is the Improved algorithm sequence, and the control groups are the pure cipher text sequence before OTP data extraction, the static algorithm sequence, and the dynamic algorithm sequence.

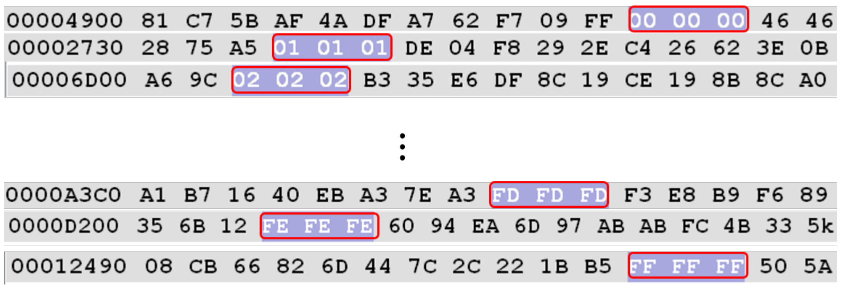
The result is shown in Figure 7. All of the control group passed every statistical random number test, but the experimental group did not. Although this statistical test cannnot entirely a substitute for cryptographic analysis, it can be useful at least as a first step in determining whether a pseudo random number generator is appropriate for a particular cryptographic application. The experimental group did not pass a number of tests such as BlockFrequency, Longest Run, Fourier Transform Test, etc., and did not cross the minimum margin line to determine whether it could be used as a cryptographic pseudo random number. Therefore, it can be judged that this data set is very weak to use as a cryptographic pseudo random number.



```
RESULTS FOR THE UNIFORMITY OF P-VALUES AND THE PROPORTION OF PASSING SEQUENCES
------------------------------------------------------------------------------
   generator is </home/user/Desktop/OTP_extraction/Dynamic.bin>
------------------------------------------------------------------------------
 C1  C2  C3  C4  C5  C6  C7  C8  C9 C10  P-VALUE  PROPORTION  STATISTICAL TEST
------------------------------------------------------------------------------
  9   7   8   4   4   9   7   5   5   2  0.500934     60/60      Frequency
 60   0   0   0   0   0   0   0   0   0  0.000000 *    0/60   *  BlockFrequency
  9   9   8   9   5   3   4   5   4   4  0.437274     60/60      CumulativeSums
 10  12   5   4  10   2   4   4   6   3  0.039244     60/60      CumulativeSums
 12   6   8   5   2   6   2   5   5   9  0.122325     58/60      Runs
 59   0   1   0   0   0   0   0   0   0  0.000000 *    7/60   *  LongestRun
  5   5   6   5  11   5   6   7   3   7  0.671779     60/60      Rank
 59   1   0   0   0   0   0   0   0   0  0.000000 *    7/60   *  FFT
 36   5   2   6   9   0   0   0   2   0  0.000000 *   51/60   *  NonOverlappingTemplate
 34  10   5   1   4   3   3   0   0   0  0.000000 *   49/60   *  NonOverlappingTemplate
 35   8   5   3   3   3   2   1   0   0  0.000000 *   51/60   *  NonOverlappingTemplate
 29   8  10   1   5   1   0   3   2   1  0.000000 *   49/60   *  NonOverlappingTemplate
 18   8   5   9   5   5   2   1   4   3  0.000045 *   57/60      OverlappingTemplate
  6   7   5   6   4   5   5  10   3   9  0.637119     60/60      Universal
 60   0   0   0   0   0   0   0   0   0  0.000000 *    0/60   *  ApproximateEntropy
  6   7   6   2   3   1   3   4   3   1  0.100508     35/36      RandomExcursions
  4   3   3   5   2   4   5   6   2   2  0.602458     36/36      RandomExcursions
  3   1   2   6   8   4   5   2   2   3  0.082177     36/36      RandomExcursions
  2   4   5   4   2   4   4   6   3   2  0.671779     36/36      RandomExcursions
  5   5   7   1   1   5   5   5   2   0  0.035174     36/36      RandomExcursions
  4   6   3   6   4   1   3   5   1   3  0.299251     35/36      RandomExcursions
  5   4   6   2   4   1   2   9   1   2  0.014216     35/36      RandomExcursions
  2   2   6   4   6   3   6   3   1   3  0.253551     35/36      RandomExcursions
  3   4   1   3   3   2   8   4   5   3  0.213309     36/36      RandomExcursionsVariant
  3   2   6   2   1   6   1   4   6   5  0.122325     36/36      RandomExcursionsVariant
  4   1   1   4   5   4   3   3   6   5  0.407091     36/36      RandomExcursionsVariant
  3   2   1   1   7   2   8   1   3   8  0.001588     36/36      RandomExcursionsVariant
  3   7   3   4   2   4   5   4   3   1  0.407091     35/36      RandomExcursionsVariant
  5   7   1   5   3   2   7   1   2   3  0.054199     35/36      RandomExcursionsVariant
 60   0   0   0   0   0   0   0   0   0  0.000000 *    0/60   *  Serial
 60   0   0   0   0   0   0   0   0   0  0.000000 *    0/60   *  Serial
  7   8   6   8   5   5   2   6   9   4  0.671779     60/60      LinearComplexity
```

Figure 7. SP800-22 tested result of Improved extraction data.

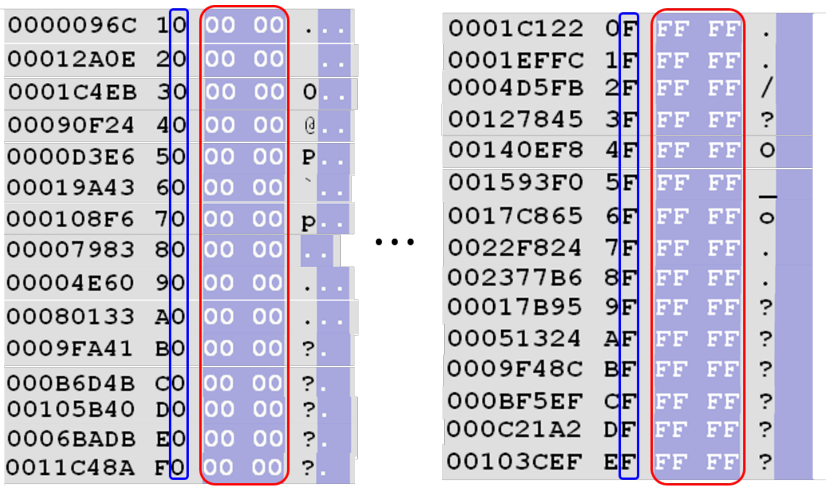
Figure 8 also shows the results of analyzing this data set with the hex viewer. As shown in Figure 8, it can be seen that 256 cases of 3-byte overlap frequently occur from 0x00 to 0xFF.



```
00004900 81 C7 5B AF 4A DF A7 62 F7 09 FF 00 00 00 46 46
00002730 28 75 A5 01 01 01 DE 04 F8 29 2E C4 26 62 3E 0B
00006D00 A6 9C 02 02 02 B3 35 E6 DF 8C 19 CE 19 8B 8C A0
                          ⋮
0000A3C0 A1 B7 16 40 EB A3 7E A3 FD FD FD F3 E8 B9 F6 89
0000D200 35 6B 12 FE FE FE 60 94 EA 6D 97 AB AB FC 4B 33 5k
00012490 08 CB 66 82 6D 44 7C 2C 22 1B B5 FF FF FF 50 5A
```

Figure 8. Real case of vulnerable data: overlapping 3-bytes.

Figure 9 shows the result of 2-byte duplication due to the improved algorithm. Even if the first extraction information and the second one are different, if the second and third ones are the same, the remaining 2-bytes are duplicated. When the LSB 4-bit of the first extraction data satisfies a specific condition (if the extraction informations are the same) as the blue part, the rest appears as the red part.



```
0000096C 10 00 00 ...        0001C122 0F FF FF .
00012A0E 20 00 00  ..        0001EFFC 1F FF FF .
0001C4EB 30 00 00 0..        0004D5FB 2F FF FF /
00090F24 40 00 00 @..        00127845 3F FF FF ?
0000D3E6 50 00 00 P..        00140EF8 4F FF FF O
00019A43 60 00 00 `..        001593F0 5F FF FF _
000108F6 70 00 00 p..        0017C865 6F FF FF o
00007983 80 00 00 ..    •••  0022F824 7F FF FF .
00004E60 90 00 00 ...        002377B6 8F FF FF .
00080133 A0 00 00 ...        00017B95 9F FF FF ?
0009FA41 B0 00 00 ?.         00051324 AF FF FF ?
000B6D4B C0 00 00 ?.         0009F48C BF FF FF ?
00105B40 D0 00 00 ?.         000BF5EF CF FF FF ?
0006BADB E0 00 00 ?.         000C21A2 DF FF FF ?
0011C48A F0 00 00 ?.         00103CEF EF FF FF ?
```

Figure 9. Real case of vulnerable data: overlapping 2-bytes.

6.1.2. Experiment 2: How Frequent Is This Extraction Vulnerability?

Experiment 2 follows Figure 10. This checks how often the phenomenon shown in Experiment 1 occurs. It extracts 3-bytes from contiguous cipher text with static, dynamic, and improved algorithms, and converts them to decimal. The samples in the experiment are the result of calculating the bit stream that generated by each algorithm from mod 10^2 to mod 10^6 (the range size is increased by 10 times). At this time, and 2^{20} samples were examined for each algorithm. For example, converting 3-byte data 0xE7812B of OTP to decimal is 15,171,883, then mod 10^2 result is 83, and possible range is $[0, 10^2 - 1]$. Or mod 10^4 result is 1883; the possible range is $[0, 10^4 - 1]$. Similar to before, the experimental group is an improved algorithm data set, and the controls are static/dynamic algorithm data sets. Since this experiment is a process of checking the frequency of a specific decimal number after extraction, a pure cipher text sequence is not included in the control group. The results of truncation of the values obtained by each algorithm for digits are summarized in the following Table 3.

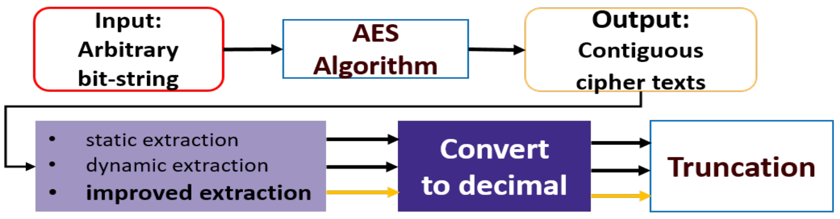


Figure 10. Truncation of OTP strings to varying lengths.

Table 3. Results of Experiment 2: This table shows numerically how often certain numbers occur when the improved algorithm is used.

- Element: A decimal number obtained with each extraction algorithm(static, dynamic, improved)
- Experimental group: Improved algorithm data set
- Control groups: Static/dynamic algorithm data sets
- Sample : 2^{20} elements (1,048,576)
- Random variable X: Number of occurrences of each "element" in the range (frequency)
- $E(X)$: Expected value of X when each "element" in the range is uniformly occur. ($\frac{\sum X_i}{|Range|}$)
- $MAX(X)$: The number of 'most frequent' element(s) in the range
- Threshold H: Heuristic value that distinguishes each extraction method
- $\#(X > H)$: The number of elements whose random variable X exceeds the threshold
- Eccentricity e : Relative measure of how far from normal. ($\frac{MAX(X)}{H}$)

Range	$E[X]$		$MAX(X)$	H	$\#(X > H)$		$\#(X \leq H)$		e
$[0, 10^1 - 1]$	$\approx 10^5$	-		-	-		-		-
$[0, 10^2 - 1]$	$\approx 10^4$	static	10,714	10,700	static	2	static	98	1.001
		dynamic	10,679		dynamic	0	dynamic	100	0.998
		improved	10,821		improved	10	improved	90	1.011
$[0, 10^3 - 1]$	$\approx 10^3$	static	1151	1100	static	61	static	939	1.046
		dynamic	1154		dynamic	59	dynamic	941	1.050
		improved	**1358**		*improved*	**256**	improved	744	**1.235**
$[0, 10^4 - 1]$	$\approx 10^2$	static	148	200	static	0	static	10000	0.740
		dynamic	151		dynamic	0	dynamic	10000	0.755
		improved	**413**		*improved*	**256**	improved	9744	**2.065**
$[0, 10^5 - 1]$	$\approx 10^1$	static	33	100	static	0	static	100,000	0.33
		dynamic	29		dynamic	0	dynamic	100,000	0.29
		improved	**333**		*improved*	**256**	improved	99744	**3.33**
$[0, 10^6 - 1]$	$\approx 10^0$	static	9	35	static	0	static	1,000,000	0.257
		dynamic	8		dynamic	0	dynamic	1,000,000	0.229
		improved	**320**		*improved*	**256**	improved	999744	**9.143**

6.2. Results

The random variable X represents the frequency of occurrence of each "element" of range. Here, element means a decimal number obtained through each extraction algorithm. For this experiment, we collect a total of 2^{20} elements and check the frequency of each decimal number. $E(X)$ is the expected value. If the probability of occurrence of all elements is uniform, the frequency of each element in the range should be distributed equally by $E(X)$. In addition, if the probability is uniform, the maximum number of element(s) in the range that occur most in the sample, $MAX(X)$, should not be much different from $E(X)$. In all ranges, the control group's $MAX(X)$ is close to $E(X)$ and the difference between each control group (static, dynamic) is 5 or less. The experimental group also shows a similar tendency to the control group when the range was less than or equal to $[0, 10^2 - 1]$. However, the gap between the experimental group and the control group becomes wider when the range is greater than or equal to $[0, 10^3 - 1]$. The $MAX(X)$ of the experimental group differs from its $E(X)$ by more than $200 \sim 300$. Similarly, the difference in $MAX(X)$ between the control group and the experimental group can be seen that the difference is about 300.

Based on the above analysis, we present Threshold H, an experimentally obtained value that can clearly classify experimental and control groups. The samples in the control group appear to be close to the expected values, while the samples in the experimental group differ significantly from $E(X)$. Therefore, the improved algorithm could be identified from the three algorithms using the threshold values for each range. If the improved algorithm is used, OTP is selected from the reduced set R ($|R| = 256$) with a probability of 1/16, which maps many times to 256 elements. Evidence for this is that the number of elements larger than the threshold ($\#(X > H)$) is 256 (in bold), and even one element in the rest of the controls does not exceed the threshold.

6.3. Discussion

Based on this, we define the eccentricity, e, as a relative measure of how far it is from normal, as $\frac{MAX(X)}{H}$, which gives us the last column in Table 3. The larger the range, the closer e becomes to 0 in the control group, but it becomes larger than 2 in the experimental group. This means that the closer you get to the OTP that people use in real life, the less random the data in the Improved algorithm is. In other words, OTP can provide sufficient randomness because the greater the range, the smaller the uniform probability of the elements that can be mapped. However, when using this (improved) algorithm, the range is frequently reduced, leading to a sharp increase in the probability that certain 256 elements will appear.

Combining Experiments 1 and 2, if an attacker wants to predict the value of OTP using the improved algorithm, the value of OTP can be predicted with probability $\frac{1}{2^8}$, which is much higher than $\frac{1}{2^{20}}$. From the point of view of "Generating secure OTP values", in order to alleviate the problems presented in Vulnerabilities 1 and 2 and Experiments 1 and 2 against them, we recap the solution we proposed.

- Enhance the generation algorithm: When generating an OTP, a cryptographic algorithm that provides security strength suitable for the current era should be used. As shown in Table 2 above, there are methods to use hash functions such as SHA2 and SHA3 that provide 112-bit or higher security strength, or block ciphers such as AES-128,192,256. It is also okay to use standard cryptographic algorithms defined by the country that implements the OTP service;in Korea, LEA-128,192,256, ARIA-128, 192, 256, etc. can be used.
- Use of static or dynamic algorithm: We should not use the improved algorithm. Using this algorithm increases the probability of predicting the value of OTP. In cryptosystems, OTP is widely used because of its advantages; the value is valid only once, has randomness, and can not predict the next value. However, if the values can be easily predicted as in the results of Experiments 1 and 2, the meaning of using this will fade. Therefore, static and dynamic algorithms should be used instead of improved algorithms that eliminate these advantages.
- Increase the amount of extraction to 4-byte: Increase the OTP extraction range. This way, we can also use digits where the number change is not large, so that we can reduce the probability of prediction each time we increase one digit. When the amount of extraction is increased from 3-byte to 4-byte, the range is [0, 16777215] (i.e., $[0, 2^{24} - 1]$) to [0, 4294967295] (i.e., $[0, 2^{32} - 1]$, About 4 billion cases). Therefore, OTP can be used up to 7 and 8 digits instead of 6 digits. In this case, since $[0, 10^7 - 1]$ and $[0, 10^8 - 1]$ can be used, the predictable probability decreases by 1/10 each time by one digit increment.
- Increases the number of LSB digits used as extraction information, and increases the length of the output bit string accordingly: Since 16 offsets can be selected using LSB 4-bit (2^4) extraction information, the length of the output bit stream is used as 20-bytes (=16 + 4). Here, adding 4 to the length to be extracted has this meaning. Suppose that when the dynamic algorithm is used, the maximum value that can be expressed in 4 bits is 15.At this time, if 3-byte data is extracted, it becomes $arr[15]||arr[16]||arr[17]$, and if 4-byte is extracted, it becomes $arr[15]||arr[16]||arr[17]||arr[18]$. If 4 is not added to the length, there is insufficient extraction data and a memory overflow occurs, or the data at the last offset used for extraction information is accessed again. That way, the data is not independent. When LSB is used as 5-bit or 6-bit, it is also appropriate to generate at least 4-byte additionally. In this case, the probability of offset overlapping decreases by $\frac{1}{2}$ each time the bit is increased by one.

7. Conclusions

OTP is used for various purposes such as multi-factor authentication [28]. Although 5G and next-generation cryptography are being studied, it is still a reality to trust and use legacy cryptosystems. However, OTP standards in Korea have not been updated since enactment in early 2010, and even when cryptographically secure algorithms (satisfying security strength) are used in the OTP generation process, the vulnerability can occur as the result of this paper during the extraction process. We presented and analyzed the weaknesses of the OTP system, which is still widely used by many people in daily life. Both the process of extracting a small bit and the process of converting it to a decimal are computing processes included for human convenience. We insist that the processes applied for Human-Centered computing can cause critical defects. We also contributed to reflect on the security perspectives that current OTP systems should consider by providing solutions to address the vulnerabilities we claim. In fact, neither the system was critically attacked by the vulnerability mentioned in this paper, nor did the problem occur. Even so, you should be aware that problems can arise, and you should consider these problems in authentication systems, which are currently vulnerable situations. It is also necessary to revise the standard, and ongoing research is needed to ensure that existing authentication methods such as OTP can be trusted and used; until one of the multi-factor authentication technologies has been replaced.

Author Contributions: Conceptualization, H.K. and J.H.; methodology, H.K.; software, H.K. and J.H.; validation, C.P.and O.Y.; formal analysis, H.K.; investigation, H.K.; resources, H.K.; writing—original draft preparation, H.K.; writing—review and editing, H.K., C.P. and O.Y.; supervision, C.P. and O.Y. project administration, C.P. and O.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Acknowledgments: This work was supported as part of Military Crypto Research Center (UD170109ED) funded by Defense Acquisition Program Administration (DAPA) and Agency for Defense Development (ADD).

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

OTP	One Time Password
PRNG	Pseudo-Random Number Generators
CPRNG	Cryptographically Secure PRNG
MITM	Man-in-the-Middle
MITB	Man-in-the-Browser
MITPC	Man-in-the-PC
LSB	Least Significant Bit
USIM	Universal Subscriber Identity Module
TTA	Telecommunication Technology Association
RFC	Request for Comments
NIST	National Institute of Standards and Technology
NIS	National Intelligence Service
SP	Special Publication
AES	Advanced Encryption Standard

References

1. Haller, N.M. The s/key One-Time Password System. In *Symposium on Network and Distributed System Security*; 1994. Available online: https://tools.ietf.org/html/rfc1760 (accessed on 1 March 2020).
2. Mohammadi, V.; Rahmani, A.M.; Darwesh, A.M.; Sahafi, A. Trust-based recommendation systems in Internet of Things: A systematic literature review. *Hum.-Centric Comput. Inf. Sci.* **2019**, *9*, 21. [CrossRef]
3. Jeong, Y.S.; Park, J.H. Security, Privacy, and Efficiency of Sustainable Computing for Future Smart Cities. *JIPS (J. Inf. Process. Syst.)* **2020**, *16*, 1–5.

4. Park, J.Y.; Huh, E.N. A Cost-Optimization Scheme Using Security Vulnerability Measurement for Efficient Security Enhancement. *JIPS (J. Inf. Process. Syst.)* **2020**, *16*, 61–82.
5. Kang, J. Mobile payment in Fintech environment: Trends, security challenges, and services. *Hum.-Centric Comput. Inf. Sci.* **2018**, *8*, 1–16. [CrossRef]
6. Kim, H.W.; Jeong, Y.S. Secure authentication-management human-centric scheme for trusting personal resource information on mobile cloud computing with blockchain. *Hum.-Centric Comput. Inf. Sci.* **2018**, *8*, 11. [CrossRef]
7. Sun, H.; Sun, K.; Wang, Y.; Jing, J. TrustOTP: Transforming smartphones into secure one-time password tokens. In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, 12–16 October 2015; pp. 976–988.
8. Cheng, F. A secure mobile OTP Token. In *International Conference on Mobile Wireless Middleware, Operating Systems, and Applications*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 3–16.
9. Menezes, A.J.; Katz, J.; Van Oorschot, P.C.; Vanstone, S.A. *Handbook of Applied Cryptography*; CRC Press: Boca Raton, FL, USA, 1996.
10. Stinson, D.R.; Paterson, M. *Cryptography: Theory and Practice*; CRC Press: Boca Raton, FL, USA, 2018.
11. Barker, E.B.; Kelsey, J.M. *Recommendation for Random Number Generation Using Deterministic Random Bit Generators (Revised)*; US Department of Commerce, Technology Administration, NIST: Gaithersburg, MD, USA, 2007.
12. Barker, E.; Kelsey, J. *Recommendation for Random Bit Generator (RBG) Constructions*; Technical Report; National Institute of Standards and Technology: Gaithersburg, MD, USA 2016.
13. Kaur, N.; Devgan, M.; Bhushan, S. Robust login authentication using time-based OTP through secure tunnel. In Proceedings of the 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi, India, 16–18 March 2016; pp. 3222–3226.
14. M'Raihi, D.; Machani, S.; Pei, M.; Rydell, J. Totp: Time-Based One-Time Password Algorithm. In *Internet Request for Comments*. 2011. Available online: https://tools.ietf.org/html/rfc6238 (accessed on 1 March 2020).
15. Jaehoon, N.; Gang, U. *TTAK.KO-12.0120: Assurance Level of One-Time Password Authentication Service*; TTA: Seongnam, Korea, 2009.
16. An, J.W. A Study on Interactive Authentication Method Using Mobile One Time Password Interlocked Transaction for Secure Electronic Financial Transactions. Master's Thesis, Kookmin University, Seoul, Korea, 2010.
17. Hong, W.C.; Lee, K.W.; Kim, S.J.; Won, D.H. Vulnerabilities Analysis of the OTP Implemented on a PC. *KIPS Trans. Part C* **2010**, *17*, 361–370. [CrossRef]
18. Yoo, C.; Kang, B.T.; Kim, H.K. Case study of the vulnerability of OTP implemented in internet banking systems of South Korea. *Multimedia Tools Appl.* **2015**, *74*, 3289–3303. [CrossRef]
19. Barker, E.B.; Kelsey, J.M. *Sp 800-90a. Recommendation for Random Number Generation Using Deterministic Random Bit Generators*; Technical Report; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2012.
20. Turan, M.S.; Barker, E.; Kelsey, J.; McKay, K.A.; Baish, M.L.; Boyle, M. Sp800-90b. Recommendation for the entropy sources used for random bit generation. *NIST Spec. Publ.* **2018**, *5–39*.
21. Gu, H. *TTAK.KO-12.0193: Algorithm Profile for One-Time Password*; TTA: Seongnam, Korea, 2012.
22. Haller, N.; Metz, C.; Nesser, P.; Straw, M. A One-Time Password System. In *Network Working Group Request for Comments*; 1998; Volume 2289. Available online: https://tools.ietf.org/html/rfc2289 (accessed on 1 March 2020).
23. M'Raihi, D.; Bellare, M.; Hoornaert, F.; Naccache, D.; Ranen, O. *HOTP: An HMAC-Based One-Time Password Algorithm*; The Internet Society, Network Working Group, RFC4226, 2005. Available online: https://tools.ietf.org/html/rfc4226 (accessed on 1 March 2020).
24. Huiwon, S.; Ujin Gang, S.S. *TTAK.KO-12.0100: Security Requirements for OTP Key Management*; TTA: Seongnam, Korea, 2009.
25. Barker, E. *NIST Special Publication 800-57 Part 1 Revision 4, Recommendation for Key Management Part 1: General*; NIST: Gaithersburg, MD, USA, 2016.
26. Heron, S. Advanced encryption standard (AES). *Netw. Secur.* **2009**, *2009*, 8–12. [CrossRef]

27. Rukhin, A.; Soto, J.; Nechvatal, J.; Smid, M.; Barker, E.; Leigh, S.; Levenson, M.; Vangel, M.; Banks, D.; Heckert, A.; *NIST Special Publication 800-22 Revision 1a: A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications*; NIST: Gaithersburg, MD, USA, 2010.
28. Kwon, B.W.; Sharma, P.K.; Park, J.H. CCTV-Based Multi-Factor Authentication System. *J. Inf. Process. Syst.* **2019**, *15*, 904–919.