applied sciences

MDPI

Article

Analysis of Distinguishable Security between the One-Time Password Extraction Function Family and Random Function Family

Hyunki Kim [1,†] and Okyeon Yi [2,*,†]

[1] Hyundai Autoever Co., Ltd., Platform Element Technology Team, 12, Teheran-ro 113-gil, Gangnam-gu, Seoul 06171, Republic of Korea; hyunki.kim@hyundai-autoever.com

[2] Financial Information Security, Kookmin University, 77 Jeongneung-ro, Seongbuk-gu, Seoul 02707, Republic of Korea

* Correspondence: oyyi@kookmin.ac.kr

† These authors contributed equally to this work.

Featured Application: Cryptography; Random Number analysis.

Abstract: A one-time password is a security system that uses a password that is only used once for authentication, and it is commonly used in multi-factor authentication systems. The process of generating an OTP is very similar to generating pseudorandom sequences in cryptography. However, since only a part of the bit string is used in OTP, an algorithm is needed to extract that part. In addition, the OTP process also includes converting the value of the bit string value into decimal form for human perception. This paper focuses on analyzing the extraction function, which is the step before the hexadecimal is reprocessed into the decimal form. We analyze a function family, which includes functions used in the process of extracting a bit string in terms of distinguishable security. As a result, we conclude that the OTP extraction function family is vulnerable in terms of distinguishable security compared to the random function family.

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

check for updates

Citation: Kim, H.; Yi, O. Analysis of Distinguishable Security between the One-Time Password Extraction Function Family and Random Function Family. *Appl. Sci.* **2023**, *13*, 8761. https://doi.org/10.3390/app13158761

Academic Editors: Gianluigi Ferrari, Leandros Maglaras, Helge Janicke and Mohamed Amine Ferrag

Received: 1 June 2023
Revised: 7 July 2023
Accepted: 26 July 2023
Published: 28 July 2023

1. Introduction

This paper is a follow-up to a previous publication entitled "Analysis of Vulnerabilities That Can Occur When Generating One-Time Password" [1]; it includes multiple experimental data on vulnerability points that can occur in OTP systems. While the previous paper hypothesized and conducted experiments to derive OTP security, this follow-up paper establishes an oracle attack model and derives theoretical security from a cryptographic perspective, which is the biggest difference. Reviewing the previous paper first may help in understanding this paper.

A cryptographic system is applied, assuming the use of secure random numbers. In other words, random numbers are essential elements for cryptographic functions, such as confidentiality, authentication, availability, etc. [2]. Meanwhile, the encoding of data is a fundamental procedure in all digital environments. The values generated by the algorithm are bit strings consisting of zeroes and ones. IT developers apply these values by combining zeroes and ones into four or eight units and reading or writing them in hexadecimal. However, situations can arise where people who are not in the IT industry or are not familiar with PCs (such as the elderly or children) need to use the values generated by the algorithm. This includes "typing in passwords for bank transactions" or "typing in One-Time Passwords (OTP) values issued for entity authentication". If the generated bit string is used as hexadecimal, it can be applied to the authentication system, selected by uniform distribution without wasting any values. However, considering people who are not familiar with hexadecimal (such as $0xA$, $0xB$, $0xC$, $0xD$, $0xE$, and $0xF$), the selected hexadecimal values are conveniently reprocessed into decimal values [3,4].

An OTP, which guarantees security based on the characteristics of random numbers, is vulnerable if weaknesses are exposed during the actual application step (such as the extraction step), even if this is generated based on a secure algorithm. When extracting the number of bits to be used as the OTP and converting them into decimal form, the variation in each digit ($\forall i \in Z_8,\ 10^i$th) is not uniform. To increase unpredictability, which should be difficult to predict for higher security, the digits with the smallest variation are excluded.

This paper analyzes the extraction function, which is a step before the hexadecimal is reprocessed into decimal form. In this process, the function family, including the OTP extraction function used, is analyzed in terms of distinguishable security; as a result, it is concluded that the OTP extraction function family is less secure in terms of distinguishability when compared to the random function family.

2. Background

2.1. Random Number

Cryptographically secure random numbers are used in various security parameters. They must satisfy unpredictability, unbiasedness, and bit independence, and must provide security strength recommended in modern times. To ensure unbiasedness and bit independence, a deterministic random number generator (DRBG) is used; to satisfy unpredictability and security strength recommended by the government, various entropy sources (noise sources) are collected and mixed into the DRBG as input with recommended or higher security strength.

In the field of cryptography (and statistics), randomness refers to the property of being selected randomly within a defined range, making it difficult to predict the outcome of an event or find patterns. Randomness is like the result of tossing a fair and unbiased coin continuously. If we assume that each side of the fair coin is represented by 0 or 1 when it is tossed once, the probability of getting either 0 or 1 is the same, and each toss is independent [5]. Therefore, the result of continuously flipping a fair coin is equivalent to the result of an ideal random number generator. A sequence is considered to be random if each bit is independent and identically distributed (IID) with equal probability [6].

In an environment where encryption is used, random numbers must be unpredictable. Even a slight possibility of predicting random numbers can greatly affect cryptographic security. Depending on the situation, an attack may be possible on a secure protocol designed by predicting sensitive parameters such as an encryption key, and encrypted messages output by secure standard algorithms can be decrypted with a predicted key. Random numbers generated through encryption algorithms should not only be unpredictable for future values based on current values but should also prevent inferring past records. Since random number generation algorithms are generally publicly available, the input values (entropy sources) of the algorithm must remain confidential to ensure the unpredictability of values [5,6].

Since cryptographic systems frequently use random numbers, secure random numbers within the system must be generated quickly. However, obtaining random numbers for cryptographic purposes without a separate algorithm is impossible, and even if a generator can produce unpredictable values each time it outputs (a true random number generator), its speed may not meet the required availability. For this reason, deterministic algorithms are used in cryptography to generate secure random numbers quickly. The algorithm used in this case is called a deterministic random bit generator (DRBG), which generates random numbers according to a predetermined logic based on the input seed [7]. Therefore, if the same seed is set for the identical DRBG, the output values are exactly the same. Due to this characteristic, if the algorithm user uses a predictable value as the seed, an attacker can accurately determine the random numbers generated by the user. Therefore, the seed used to initialize or update the DRBG must be unpredictable and different every time. Generally, the seed is composed of noises that can be easily obtained in the operating environment [8–11].

2.2. One-Time Password (OTP)

A one-time password is an electronic financial transaction authentication service that authenticates users with a different password for each transaction, making it mathematically impossible to guess the next password from the currently used password. By using shared secret information between the institution and the user, such as timing or event information, the algorithm for generating the disposable password is created internally so that attackers cannot guess it on their own. Currently, OTPs are provided as tokens, card-type terminals, and mobile applications (e.g., phones) [12].

Authentication methods using OTPs are divided into synchronous and asynchronous methods, and are classified into timing synchronization, event synchronization, and combination synchronization [13]. Each method has its advantages and disadvantages, and the timing method may vary, depending on the situation. Timing synchronization requires accurate synchronization between the OTP terminal and the authentication server, and with event synchronization, events generated on the terminal and the server must be synchronized accurately. Combination synchronization complements the problems of timing and event synchronization methods, but issues such as synchronization failure or error range may still occur [14,15].

An asynchronous method is an authentication technique that does not require synchronous information between the OTP terminal and the authentication server. An example is the challenge–response method, which sends a query value to a terminal that generates a password, verifies whether the user's response is valid, and performs authentication. When a user requests a transaction using that password, the server sends a random query value to the user, and the user enters the output value from the terminal and sends it back to the server. If the response sent by the user is valid, the server authenticates the user and performs the transaction. However, this challenge–response method has an inconvenience—the user must enter the query value and response value, and when using the communication network, there may be charges for the transmitted data. In addition, the server bears the burden of generating and managing the query and response values.

3. Related Works

3.1. NIST Standards for Cryptographic Random Numbers

Ref. [7] describes a PRNG that uses the approved cryptographic algorithm. This makes it impossible to distinguish between an ideal random sequence and a pseudorandom sequence without infinite computational power. It is designed with a fixed finite security strength to measure the workload required to attack PRNG. Ref. [9] describes mechanisms and entropy sources for obtaining unpredictable bits through non-deterministic processes. These entropy sources are equal to the security strength. Ref. [16] describes the overall RNG structure that combines 90A and 90B, as shown in Figure 1 [7,9,16].

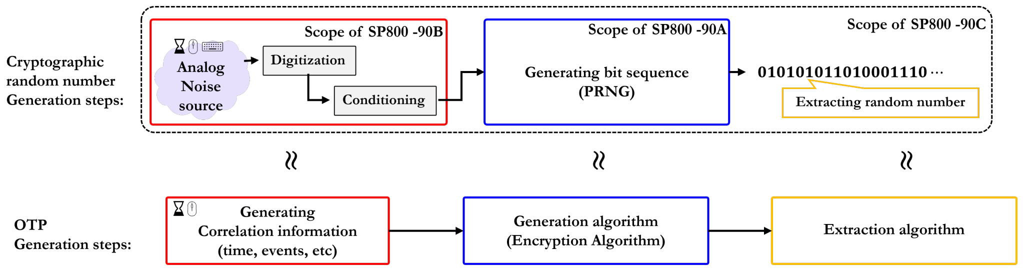

Figure 1. Similar steps of the OTP and random number generation.

3.2. OTP Mechanism

The OTP generation mechanism consists of three stages [17]: 'Generating correlation information', 'Generation algorithm', and 'Extraction algorithm'. This is very similar to the process of generating pseudorandom numbers used in cryptography[18]. Figure 1 shows the similar parts of the two mechanisms in the same color.

OTP is generated following the procedure in Figure 1.

3.2.1. Correlation Information Generation

Correlation information refers to random numbers that can be collected from time information, event occurrence information, and so on. This can be a complete value generated by the OTP generation method based on the current time, event occurrence, or a random number that has been transformed through additional processes, such as hashing or encryption. Correlation information is necessary for the unpredictability of OTP, similar to the noise source (or entropy source) required to generate cryptographically secure pseudorandom numbers. This process corresponds to the red part in Figure 1.

3.2.2. Generation Algorithm

The generation algorithm refers to the cryptographic algorithm used when generating OTP, and generates a 20-byte bit string by encrypting the correlation information using this algorithm [19]. The Korean OTP standard [20] introduces the cryptographic algorithm used in this process. The generation algorithm is deterministic, so it generates the same ciphertext from the same correlation information. However, depending on the extraction algorithm applied to the generated bit string, different OTP values can be generated from a single bit string. The encryption algorithm used must provide a security strength of at least 112 bits [8,17,19]. This process corresponds to the blue part in Figure 1.

3.2.3. Extraction Algorithm

This is an algorithm that extracts three-byte or four-byte data from the encrypted bit string to use as the OTP. This can be divided into static algorithms and dynamic algorithms (RFC4226 [19], an OTP-related standard, only specifies dynamic algorithms; and the Telecommunication Technology Association (TTA) [17] specifies static, dynamic, and improved dynamic algorithms), and it uses an 'extraction index' to specify the extraction location. This algorithm uses the value at the extraction index as the 'extracted data'. The extraction index can use a value defined in a specific area of the ciphertext or an undefined value. After extracting three bytes, the hexadecimal value of the three bytes is converted to a decimal form for ease of human recognition. At this point, the range of the extracted 3-byte value is [0, 16,777,215] (i.e., [0, $2^{24}-1$]), but the upper two digits change little (10^7: $d_{7th\ digit} \in \{0,1\}$, 10^6: $d_{6th\ digit} \in \{0,1,2,3,4,5,6\}$), so they are discarded, and only the lower six digits are used. Therefore, the final range is [0, 999,999]. This process corresponds to the yellow part in Figure 1.

The static algorithm extracts data by specifying in advance the extraction information (the offset of the cipher text) to be used. The dynamic algorithm extracts data by obtaining extraction information from the cipher text. The improved dynamic algorithm is used when generating OTP using a chip used as a smart card or USIM [17]. Figure 2 shows the extraction algorithms by type. It illustrates the operation of static, dynamic, and improved dynamic extraction algorithms in order, showing slight differences in their operations [1].

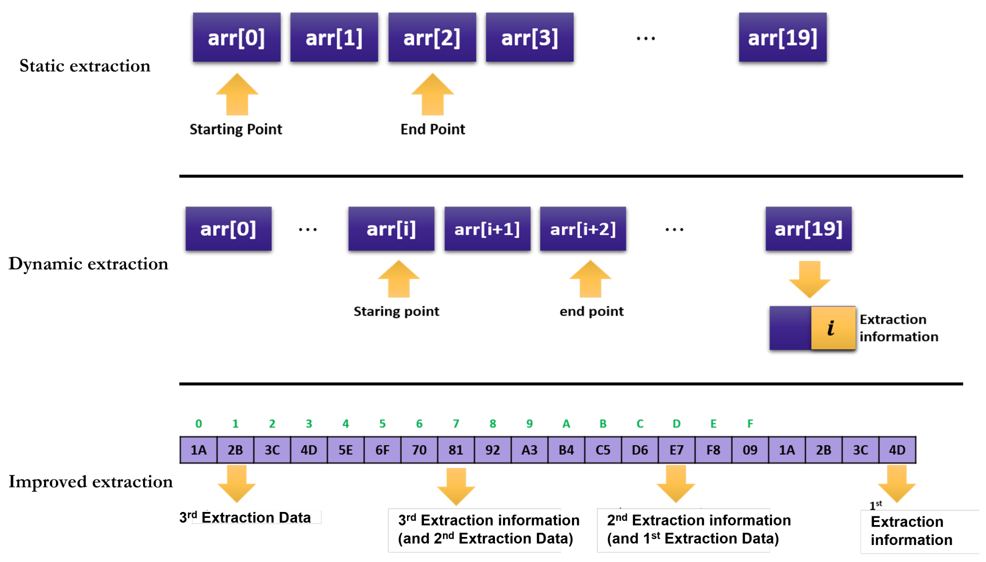
Static extraction
arr[0]
arr[1]
arr[2]
arr[3]
…
arr[19]
Starting Point
End Point
Dynamic extraction
arr[0]
…
arr[i]
arr[i+1]
arr[i+2]
…
arr[19]
Staring point
end point
i
Extraction information
Improved extraction
0 1 2 3 4 5 6 7 8 9 A B C D E F
1A 2B 3C 4D 5E 6F 70 81 92 A3 B4 C5 D6 E7 F8 09 1A 2B 3C 4D
3rd Extraction Data
3rd Extraction information (and 2nd Extraction Data)
2nd Extraction information (and 1st Extraction Data)
1st Extraction information

Figure 2. Type of extraction algorithms: static, dynamic, and improved.

4. Materials and Methods

4.1. Scenario of Vulnerability in OTP Extraction Function

The OTP vulnerability analyzed in Ref. [1] is shown in Figure 3.

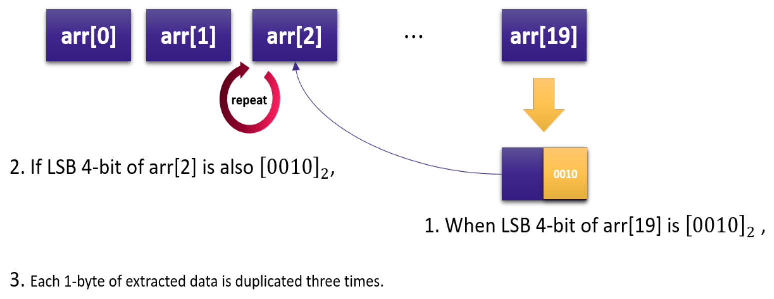
arr[0]
arr[1]
arr[2]
…
arr[19]
repeat
0010
2. If LSB 4-bit of arr[2] is also $[0010]_2$,
1. When LSB 4-bit of arr[19] is $[0010]_2$,
3. Each 1-byte of extracted data is duplicated three times.

Figure 3. Vulnerabilities in the improved dynamic extraction algorithm.

The improved dynamic extraction algorithms can be vulnerable to OTP generation. Figure 3 illustrates a situation where a vulnerability can occur in that case.

When using the improved dynamic extraction algorithm of Ref. [17], if the first extraction index and the next extraction index are the same, all values of the three-byte extraction data outputted will be duplicated. For example, if the four-bit LSB of the lowest data byte of the ciphertext is used in the initial extraction data, and the lowest data byte $arr[19]$ is 0×42, the extraction index is 0×02, which is the four-bit LSB, and the initial extraction datum is the value of $arr[2]$. Using the four-bit LSB of $arr[2]$ in the second extraction data, if the four-bit LSB of $arr[2]$ is also 0x02, and the second extraction datum will also be the value of $arr[2]$. Similarly, the last (third) extraction datum will also have the same value, resulting in the total three-byte extracted data being duplicated. The output format of the example is $arr[2]||arr[2]||arr[2]$.

4.2. Predictability of OTP Extraction Function Values

If the values of the extracted data are all duplicated as shown above, the OTP range will become a set of elements in the form of "$arr[i]||arr[i]||arr[i]$", where $0 \times 00 \leq arr[i] \leq 0 \times \mathrm{FF}$ and $0 \leq i \leq 15$. If we call this reduced set V, then $|V| = 256$. Even if we convert the elements of this set into decimal numbers using only the lower 6 digits, there will be no collision pairs, and the set will still have 256 elements. As a result, ∣V∣ shows a decreased rate of about 99% compared to [0, 999,999], and the range is rapidly reduced, making it easy to predict the range of possible random numbers that can be output. Ref. [1] also analyzes how frequently this phenomenon occurs.

As indicated in Figure 4, Ref. [1] conducted experiments on a single continuous random number sequence, applying three extraction algorithms handled in the TTA standard, converting them to decimal numbers, and outputting only up to a maximum of six digits. For example, if the OTP's three-byte data 0xE7812B are converted to decimal form, the resulting $15,171,883$ is outputted up to two digits, which is 83, and the possible range is $[0, 10^2 - 1]$. Alternatively, if it is outputted up to four digits, it becomes 1883, and the possible range is $[0, 10^4 - 1]$. Table 1 shows the experimental group, which applies an improved dynamic extraction algorithm to the sequence to demonstrate predictability, and the control group, which applies the other two algorithms to the sequence. The random variable X represents the frequency of occurrence of each 'element' within the range of decimal numbers outputted; that is, the frequency. If all elements have an equal probability of occurrence, each element within the range should be distributed evenly at $E(X)$. However, by looking at $MAX(X)$ in Table 1, we can see that the value of the experimental group, which is the 'improved dynamic', is significantly different from that of the control group. Threshold H is a value obtained experimentally in Ref. [1] and can clearly classify the experimental and control groups. While the samples of the control group appear somewhat similar to the expected value $E(X)$, the samples of the experimental group show a significant difference from the expected value. By using the improved algorithm, the probability of OTP results being mapped to 256 numbers is increased by $\frac{1}{16}$ [1]. As evidence of this, the number of 'elements exceeding the threshold' (i.e., $\#(X > H)$) is exactly 256, and none of the remaining elements of the control group exceed threshold H. Therefore, this algorithm is vulnerable to prediction attacks by attackers due to the increased predictability of certain numbers appearing with a high probability.

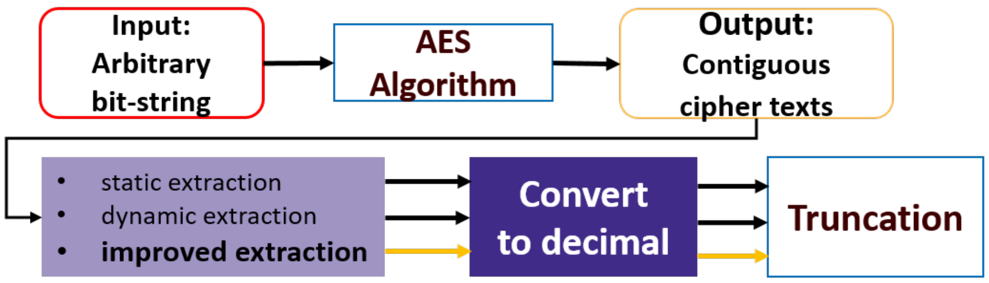

Figure 4. Truncation of OTP strings to varying lengths.

4.3. Analysis of Distinguishability of OTP Extraction Function

Expanding on the predictability experiment mentioned above, we propose a distinguishable security model. Due to the use of an improved dynamic extraction algorithm, the reduced range of V is $|V| = 256$. Thus, we construct an oracle that distinguishes between this extraction function and a random function by leveraging the characteristic of a higher probability of mapping to a specific number.

4.3.1. Distinguishable Security Model for the Proposed OTP Extraction Function

When comparing an arbitrary function family to a family of random functions with the same domain and range, if the two families are indistinguishable, then the arbitrary family satisfies pseudorandomness, which is called a pseudorandom function (PRF) [21]. Based on this, we define a family of random functions and a family of extraction functions with weak properties. The domain and range of the random and weak functions are presented in Table 2. To distinguish between the family of weak extraction functions J, a set of functions, called a function family, was constructed, which generalizes the extraction function defined by TTA and has the same properties.

Table 1. Results of the experiment in Ref. [1]: This table numerically shows 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 dataset
- Control groups: Static/dynamic algorithm datasets
- 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 uniformly occurs. ($\frac{\sum X_i}{|Range|}$)
- $MAX(X)$: 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	$\boldsymbol{E[X]}$		$\boldsymbol{MAX(X)}$	$\boldsymbol{H}$		$\boldsymbol{\#(X > H)}$		$\boldsymbol{\#(X \leq H)}$	$\boldsymbol{e}$
$[0, 10^1 - 1]$	$\approx 10^5$	-		-	-		-		-
$[0, 10^2 - 1]$	$\approx 10^4$	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	10,000	0.740
		dynamic	151		dynamic	0	dynamic	10,000	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	99,744	**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	999,744	**9.143**

Table 2. Definition of function families for analysis.

Glossary of Terms	**Symbols**	**Description**
Domain of Extraction Function	D	$D = Z_{2^{24}}$, 24-bit representation area
Range of Extraction Function	R	$X \ mod \ 10^6$, $(X \in D)$
Family of Random Functions	$Func(D, R)$	A set that includes all functions with domain D and range R
Family of Vulnerable Extraction Functions	J	A set that includes all functions with the domain D and range R, and has the vulnerable characteristics mentioned in "Section 4"
Randomly Selected Instance	g	A function randomly selected from J or $Func(D, R)$, (i.e., $g \leftarrow Func(D, R)$ or $g \leftarrow J$)

Representing the elements (i.e., functions) of function families in canonical form makes it easier to understand the elements of the family. Figure 5 illustrates the process of finding a function with the same properties as function family J. The extraction function starts with generating a 20-byte random sequence using a generation algorithm (random sequence generation algorithm) to extract a 3-byte OTP. The extraction index uses a total of 4 bits (i.e., 2^4 possible number representations), so there are a total of 16 extraction locations that can be used for OTP random number extraction. If the four indices that cannot be selected as extraction positions from the 20-byte sequence are listed as $(0,1,2,3),(0,1,2,4),\cdots,(16,17,18,19)$, the total number of cases is $\binom{20}{4}=\binom{20}{16}=4845$.

Furthermore, 4-bit data are used for assigning extraction indices in 4 randomly selected locations outside of the 16 extraction locations (i.e., 4-byte = 32-bit). The possible 4-bit positions that can be used for index purposes among the 32 bits are listed as $(0,1,2,3),(0,1,2,4),\cdots,(28,29,30,31)$, resulting in a total of $\binom{32}{4}=35{,}960$ cases. With this, all elements of the function family J can be represented in a canonical manner as $E_0^0, E_1^0, \cdots, E^0_{\binom{32}{4}-1}, \cdots, E_0^1, E_1^1, \cdots, E^{\binom{20}{4}-1}_{\binom{32}{4}-1}$, and the total number of elements in the function family J, in other words, $|J|=\binom{20}{4}\times\binom{32}{4}=174{,}226{,}200$. Based on this, we can define the selection of a function in Func J as randomly choosing a function (with identical probability for all elements) as $E_j^i \leftarrow J$.

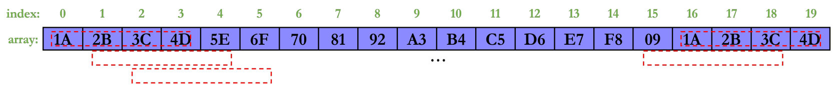

Figure 5. Conceptual diagram for generalizing the extraction function family.

Then, with regards to $\forall i,j\ (0\le i\le\binom{20}{4}-1,\ 0\le j\le\binom{32}{4}-1)$, the probability of $E_j^i \leftarrow J$ is $Pr[E_j^i\leftarrow J]=\frac{1}{\binom{20}{16}\times\binom{32}{4}}$. In contrast, the size of the random function family $|Func(D,R)|$ is $10^{6\times 2^{24}}$, so the probability of selecting a random function from $Func(D,R)$ is $Pr[g\leftarrow Func(D,R)]=\frac{1}{10^{6\times 2^{24}}}$. Since the selection of an instance from each function family has been explicitly described, we can define adversary A as an algorithm that distinguishes between the random function family and a specific function family in order to analyze distinguishability. Oracle selects an instance from a group of random functions or specific functions with a probability of $\frac{1}{2}$, and A inputs a finite number of queries ($x\in D$) to that chosen function by the Oracle [21,22]. A does not know which group of functions the Oracle has chosen, and while a finite number of queries are inputted, the Oracle retains the same instance (function) g without selecting again. The Oracle responds with $g(x)$, and based on the queries and responses, A outputs a 1-bit value $b\in\{0,1\}$, indicating whether it is *World* 0 (Oracle selects g as $g\leftarrow Func(D,R)$), or *World* 1 (Oracle selects g as $g\leftarrow J$).

4.3.2. Analysis of Distinguishable Security of the OTP Extraction Function

To represent the distinguishable model, the experiment performed by adversary A is defined as follows. The model defined in this section is illustrated in Figure 6.

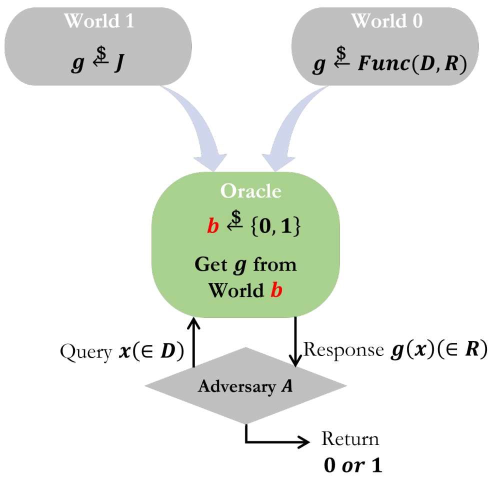

Figure 6. The Oracle model selects one instance randomly from the function family or extraction function family J.

Experiment 1	Experiment 0
$Exp_J^{prf-1}(A)$	$Exp_J^{prf-0}(A)$
$E_j^i \leftarrow J$	$E_j^i \leftarrow Func(D,R)$
$b \leftarrow A^{E_j^i}$	$b \leftarrow A^g$
$Return\ b$	$Return\ b$

To distinguish between the OTP extraction function and the random function, the adversary A algorithm is constructed, as shown in Algorithm 1 for the model defined in the previous section. Adversary A distinguishes between *World* 0 and *World* 1 using threshold H shown in Table 1, and the detailed process consists of five steps.

Algorithm 1: Algorithm for distinguishing the random function of adversary A^g

```
Input  : Queries $x_i$
Output: Decision $b$
1: $b \leftarrow 0$
2: for $i \leftarrow 0$ to $|V|$ do
3:     $cnt_{v_i} \leftarrow 0$
4: end for
5: for $i \leftarrow 0$ to $n$ do
6:     $y_i \leftarrow g(x_i)$
7:     if $y_i \in V$ then
8:         $cnt_{y_i} \leftarrow cnt_{y_i} + 1$
9:     end if
10: end for
11: for $i \leftarrow 0$ to $|V|$ do
12:     if $cnt_{v_i} \geq H$ then
13:         $b \leftarrow 1$
14:         Break
15:     end if
16: end for
17: return $b$
```

In step 1, A initializes the final output value of b to 0 (line 1). In step 2, A initializes a total of 256 cnt_{v_i} (count values for each element v_i of the reduced set V) to 0 to check how many times each v_i is output (line 2 to 4). In step 3, A inputs the query x_i to the oracle and stores the response $g(x_i)$ in y_i (line 5 to 10). If $y_i \in V$, then A updates the cnt_{v_i} initialized in step 2 to $cnt_{v_i} + 1$. A repeats steps 3 and 4 for n times (for example, 2^{20} times, as shown in Table 1). Based on Table 1 analyzed in Ref [1], adversary algorithm A succeeds in distinguishing. According to Table 1, static and dynamic extraction functions equivalent to random functions output all elements within the range (Z_{10^6}) in similar frequencies to the expected value, while the improved dynamic extraction function corresponding to the vulnerable function family J outputs the elements of set V ($V \subset Z^{10^6}$) by more than 35 times the expected value. Therefore, if Oracle chooses *World* 0, arbitrary cnt_{y_i} values do not exceed H, and if Oracle chooses *World* 1, arbitrary cnt_{y_i} values are considered to exceed H. Hence, adversary A can distinguish *World* 0 and *World* 1 with high probability by verifying the existence of the cnt_{v_i} output more than threshold H.

5. Results and Discussion

5.1. Results

At this point, from the perspective of adversary A, the attack advantage $Adv_J^{prf}(A)$ of distinguishability between the random function family and the extraction function family J is defined as follows [22,23].

$$\begin{aligned}&Adv_J^{prf}(A)\\&= Pr[Exp_J^{prf-0}(A) = 0] - Pr[Exp_J^{prf-1}(A) = 0]\\&= Pr[A^g = 0 | g \leftarrow Func(D,R)] - Pr[A^g = 0 | g \leftarrow J]\end{aligned} \quad (1)$$

$Adv_J^{prf}(A)$ represents the difference between the probability that A determines the outcome of experiment 0 as 0 (the success probability of distinguishing) and the probability that A determines the result of experiment 1 as 0 (the failure probability of distinguishing). Since $g(x_i)$ outputted by $g \leftarrow J$ in lines 11 to 17 of Algorithm 1 can be distinguished accurately, it is only necessary to consider the case of distinguishing failure, $J \subset Func(D,R)$. That is, since $|J| = \binom{20}{4} \times \binom{32}{4} = 174,226,200$, the advantage of A is:

$$\begin{aligned}&Adv_J^{prf}(A)\\&= 1 - \frac{174,226,200}{10^{6\times 2^{24}}}\\&\approx 1 - 10^{-3\times 2^{23}}\\&\approx 1.\end{aligned} \quad (2)$$

Depending on the values derived in this way, the OTP extraction function defined in Ref. [17] is distinguishable from random functions, because the adversary can directly construct effective algorithms with an advantage that is almost 1. Therefore, financial institutions and mobile systems that use this vulnerable extraction algorithm in their OTP generation mechanism should review whether there are similar vulnerabilities; it is critical to replace it with other extraction algorithms that have not yet been found to have vulnerabilities.

5.2. Discussion

We analyzed the characteristics of vulnerable extraction functions that can be utilized during the OTP generation process and proved their security from a distinguishability perspective.

Ref. [1] constructed a sufficient amount of OTP datasets as extraction methods. Based on numerous datasets, Ref. [1] demonstrated that as the OTP extraction method used in actual systems becomes more similar, the randomness of the data decreases. On the other hand, this paper demonstrated the adversary's advantage and the corresponding provable security with theory alone, without a sufficient amount of practical datasets. This vulnerable extraction function has experimentally increased predictability and theoretically increased distinguishability by compromising the randomness that is the source of security in all crypto and OTP systems.

From the perspective of generating "secure OTP values", the solution we propose to mitigate the problem is as follows:

Use static or dynamic algorithms: Not only should non-improved algorithms be used, but the specifications related to this algorithm should be removed from the standard. If this algorithm continues to be specified in the standard, the institution developing OTP may mistakenly use this vulnerable algorithm as the strongest algorithm by seeing the phrase "improved". If this algorithm is used in the institution using OTP, the algorithm is easily analyzed based on the adversary's attack model presented in this paper. This increases the probability of predicting OTP values. OTP is widely used because it is not reused, has randomness, and cannot predict the next value based on current information. However, as in the results of this paper, if the value is distinguished by the algorithm, the OTP output can be predicted. In this case, the advantages provided by OTP disappear. Therefore, improved algorithms that eliminate these advantages should be removed from the standard, and static and dynamic algorithms should be used instead.

6. Conclusions

Currently, OTP is used in various ways, such as multi-factor authentication. However, the OTP standard in Korea was established around 2010 and has not been updated since then [24]. In this paper, we analyzed the security of the extraction algorithm, which is part of the OTP generation mechanism defined in the TTA's OTP-related standard, from the perspective of distinguishability, to clarify its probable security. Adversary A's algorithm in this paper has a limit of using the experimentally obtained threshold H and preparing no less than 2^{20} queries. However, it is significant that we identified the vulnerability of the function that may still be used in the field. Although 5G and next-generation cryptography are being studied, the current reality is that we still have to rely on existing cryptographic systems. Therefore, it is important to continue paying attention to existing cryptographic systems such as OTP to make them reliable until alternative technologies emerge.

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

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable for studies not involving humans or animals.

Informed Consent Statement: Not applicable for studies not involving humans.

Data Availability Statement: Not applicable for studies.

Acknowledgments: This work was supported by the Institute for Information & Communications Technology Promotion(IITP) grant funded by the Korean government (MSIP) (no. 2020-0-00085, Research on Security Technology for 6G Telecommunication with 5G+).

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

References

1. Kim, H.K.; Han, J.H.; Park, C.I.; Yi, O.Y. Analysis of Vulnerabilities That Can Occur When Generating One-Time Password. *Appl. Sci.* **2020**, *10*, 2961. [CrossRef]
2. Kim, H.K.; Kim, D.H.; Yi, O.Y. Convolution Neural Network-Based Sensitive Security Parameter Identification and Analysis. In *Wireless Communications and Mobile Computing*; Hindawi: London, UK, 2022; Volume 2022.
3. 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]
4. 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.
5. Menezes, A.J.; Katz, J.; Van Oorschot, P.C.; Vanstone, S.A. *Handbook of Applied Cryptography*; CRC Press: Boca Raton, FL, USA, 1996.
6. Stinson, D.R.; Paterson, M. *Cryptography: Theory and Practice*; CRC Press: Boca Raton, FL, USA, 2018.
7. 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.
8. Barker, E. *NIST Special Publication 800-57 Part 1 Revision 4, Recommendation for Key Management Part 1: General*; NIST: Gaithersburg, MD, USA, 2016.
9. 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 Special Publication; NIST: Gaithersburg, MD, USA, 2018; pp. 5–39.
10. Choi, H.; Ju, W.H.; Kim, H.E.; Yeom, Y.J. *Guideline for the Collection and Application of Noise Sources on Operating Systems*; TTAS.KO-12.0235/R2; Telecommunication Technology Association: Seongnam, Republic of Korea, 2020.
11. Kim, H.E.; Ju, W.H.; Choi, H.; Yeom, Y.J. *Guideline for Testing Noise Sources Used in Software Cryptographic Modules*; TTAK.KO-12.0341; Telecommunication Technology Association: Seongnam, Republic of Korea, 2018.

12. 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, Republic of Korea, 2010.
13. Jaehoon, N.; Ujin, G. *TTAK.KO-12.0120: Assurance Level of One-Time Password Authentication Service*; TTA: Seongnam, Republic of Korea, 2009.
14. 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.
15. M'Raihi, D.; Machani, S.; Pei, M.; Rydell, J. Totp: Time-Based One-Time Password Algorithm. Internet Request for Comments. Available online: https://tools.ietf.org/html/rfc6238 (accessed on 1 March 2011).
16. Barker, E.; Kelsey, J. *Recommendation for Random Bit Generator (RBG) Constructions*; Technical Report; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2016.
17. Gu, H. *TTAK.KO-12.0193: Algorithm Profile for One-Time Password*; TTA: Seongnam, Republic of Korea, 2012.
18. Haller, N.; Metz, C.; Nesser, P.; Straw, M. A One-Time Password System. Network Working Group Request for Comments. 1998; Volume 2289. Available online: https://tools.ietf.org/html/rfc2289 (accessed on 1 March 2020).
19. 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).
20. Huiwon S., Ujin Gang, S.S. *TTAK.KO-12.0100: Security Requirements for OTP Key Management*; TTA: Seongnam, Republic of Korea, 2009.
21. Kim, N.Y.; Kang, J.S.; Yeom, Y.J. Provable Security of PRF-based Key Derivation Functions according to Input Types of Counters. *J. Korea Inst. Inf. Secur. Cryptogr.* **2015**, *25*, 547–557.
22. Bellare, M.; Rogaway, P. *Introduction to Modern Cryptography*; CRC: Boca Raton, FL, USA, 2005; pp. 59–67.
23. Park, H.J. A Study on the Security Evaluation for Cryptographically Secure Random Number Generators. Ph.D. Thesis, Department Financial Information Security, Kookmin University, Seoul, Republic of Korea, 2020.
24. 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.