Journal of Network and Information Security
11 (1) 2023, 20-27
http://www.publishingindia.com/jnis

Security Vulnerabilities of Popular Multifactor Authentication Methods and a Remedy

Shushan Zhao
Department of Computer Electronics & Graphics Technology, Central Connecticut State University,
New Britain, CT-06050, U.S.A.
Email: shushanz@ccsu.edu

Abstract: Authentication is of paramount importance for online services. Many online services are still using password as single authentication method, but this is not considered secure any more. Many others have switched to multifactor authentication mechanism. Nowadays many online service providers use One-time Password (OTP) as a supplementary authentication method to verify identity of the user. There are two major methods to generate OTPs: Time-based One-time Password (TOTP) and HMAC-based One-time Password (HOTP). We notice that there are several limitations or weaknesses with both. In this work, we first show some security vulnerabilities of TOTP and HOTP, then we present security improvement methods. We analyze and discuss the security features of proposed solution. The solution is generic to all platforms and operating systems, and our analysis demonstrates that it addresses security vulnerabilities of them.

Keywords: Multifactor authentication, OTP.

I. Introduction

Today we are living in a world full of online services — online shopping, online banking, online chatting etc., and recently we have upgraded above mentioned online services to mobile online services. All these services rely on a basic security service --- authentication, and it's facing more and more challenges nowadays.

The traditional and most popularly used method --- a password known only to its creator and owner --- has the inherent weakness that users must balance between security and convenience: a secure enough password is not easy to remember, and an easy-to-remember password is not secure enough. Password attacking is the main target of hackers for years, and there are many tools free online for this purpose. Most of the attacks use a password dictionary or a hashcode database (so-called rainbow table) and search for a match of password or hashed password. There are many tools to launch the attack, such as "John-the-Ripper", "medusa", "hydra" etc., and there are many tools to generate and customize the password or hashcode dictionaries, such as "crunch", "cewl", "rtgen" etc. With development of computational power and storage resources, attacks not possible or feasible before are now becoming possible and feasible. For example, on a DEC station 3100 in 1989, even if only words of 3 to 5 lower case characters are considered, testing 3000 passwords would require over 25 CPU hours [7]; while on a modern computer, testing this number of words takes less than 1 second. For another example, anyone can buy a password cracking dictionary of 15 GB containing 1,493,677,782 popularly used passwords for $5 (4). The financial and technical cost is now affordable by most individuals. This leads to the fact that passwords are getting easier and easier to break, and less and less secure.

Many online service providers have realized the trend of decreasing security of passwords and resort to other complementary methods. Security questions are another knowledge-based method, but they do not prevent family members or friends of a user to know the answers, and are also subject to shoulder surfing from strangers. A better solution is multifactor authentication, i.e. combining knowledge-based methods, possession-based methods and inherence-based methods. It's just a matter of time that online services will switch to some kind of multifactor authentication method.

Inherence-based authentication methods are also known as biometric methods. They rely on inherent and unique characteristics of a specific user, including static characteristics such as face, fingerprint, iris etc., or dynamic characteristics, such as voice, walking pattern, etc. These methods provide reliable security assurance. However, there are several concerns on them: extra hardware is required to scan or input the biometric data; people might concern about their privacy and are reluctant to give their data to an online service provider; some methods, such as iris recognition, are very intrusive [17].

Possession-based authentication methods are the most popularly used methods in multifactor authentication to complement password authentication. There are basically two types of methods in this category: hardware-possession and software-possession. Hardware-possession devices include hardware tokens and cellphones. Hardware tokens require extra cost and are hard to distribute. Cellphones are not considered as extra cost nowadays, and they can be used to receive authentication

verification in SMS messages, voice calls, or push-messages. However, cellphone methods require a side-channel from cellular network operators. The cellular network service might not be available or convenient to use when the user is traveling. And this also might cause extra delay and extra security issues. For example, a victim's cellphone SIM card can be duplicated through SIM swap fraud [14]. A portable 3G base station can be set up to intercept 3G communication, including SMS messages [10]. Shushan Zhao [18] proposes a scheme to improve SMS verification code security.

Software-possession methods can be a program, or an application installed on a computer or a smartphone, which generates a one-time password/passcode (OTP) to authenticate the user each time. OTP generators are either counter-based (or HMAC-based, HOTP) [8] or time-based (TOTP) [9]. Both HOTP and TOTP have limitations and weaknesses: TOTP requires synchronization and re-synchronization between the client and the server. HOTP is subject to differential analysis or dictionary attacks. This will be elaborated in Section II.

In light of the security challenge of so many solely password-protected online services and weaknesses and limitations of current OTP authentication methods, we find the imminent demand of a countermeasure to address these security issues. We propose security improvement on OTP methods. Specifically, we adjust existing HOTP algorithm and remove security vulnerabilities in it. The modified algorithms are not subject to differential analysis or dictionary attacks. The modification is only inside the kernel of the algorithm and does not affect its interface and application in so many existing online service applications.

The rest of this paper is organized as follows: Section II reviews background and related work. Section III demonstrates vulnerabilities of existing and popularly used OTP methods. Section IV presents improvements on security of HOTP methods. Section V discusses the security features of the proposed solution. Section VI concludes the paper.

II. Background and Related Work

Counter-based OTP (or HMAC-based, HOTP) [8] or time-based OTP (TOTP) [9] are the two most popularly used OTP methods specified in IETF RFC documents. Let's review some important specifications in the documents which this work is based on.

A. Counter-Based One-Time Password Algorithms

A counter-based one-time password algorithm, named as HMAC-based one-time password (HOTP) algorithm in the document) is described in RFC4226 [8].

A counter on client side and a counter on server side count the number of events, such as connection establishment, sequentially. The counters are supposed to be equal on both sides, and are passed to the following function:

$$HOTP(K, C) = Truncate(HMAC(K, C))\&0x7FFFFFFF \quad \text{Eq. (1)}$$

where K is the secret key shared between the client and the server, $0x7FFFFFFF$ sets the result's most significant bit to zero for the result to be convenient to use.

B. Time-Based One-Time Password Algorithms

A time-based one-time password algorithm (TOTP) is described in RFC6238 [9].

TOTP is based on HOTP with a timestamp replacing the incrementing counter:

$$TOTP = HOTP(K, T) \quad \text{Eq. (2)}$$

where T is an integer and represents the number of time steps between the initial counter time $T0$ and the current Unix time. More specifically, $T = (Current\ Unix\ time - T0)/X$, where the default floor function is used in the computation. Finally,

$$TOTP\_Value = TOTP \bmod 10^d \quad \text{Eq. (3)}$$

where d is the desired number of digits of the one-time password.

In the literature, it's already documented that existing TOTP and HOTP methods have following limitations or weaknesses [8], [19], [20]: Counter-based OTP requires synchronization and re-synchronization between the client and the server, because connection counters may be different on two sides when a connection was canceled or failed or reconnected. "Although the server's counter value is only incremented after a successful HOTP authentication, the counter on the token is incremented every time a new HOTP is requested by the user.

Because of this, "the counter values on the server and on the token might be out of synchronization" [8].

Time-based OTP requires synchronization and re-synchronization between the client and the server because of clock drift. To compromise with biased counter or time and transmission delay, they allow an acceptable size of window of different counter or time values. If this window size is too small, authentication for legitimate users tend to fail (false negative); if this window is too large, it gives the attacker the opportunity to bypass the authentication (false positive).

As is stated in National Institute of Standards and Technology (NIST) Special Publication 800-63-3 [12], multi-factor OTP authenticators contain two persistent values. The first is a symmetric key that persists for the device's lifetime. The second is a nonce that is either changed each time the authenticator is used or is based on a real-time clock. In this document, NIST also defines assurance levels and requirements of these values and algorithms to prevent various potential attacks. There are several industry standards, such as FIDO standards [5].

Unfortunately, many personal and business users either are unaware of these guidelines and standards or just bypass them because they are too heavy to implement.

III. Security Vulnerabilities of HOTP and TOTP

Both HOTP and TOTP are subject to some security vulnerabilities. We can think of and list some attacks exploiting them in this section.

A. Differential Attack

We notice that HOTP and TOTP applications implementing RFC 4226 [8] and RFC 6238 [9] are subject to potential differential attacks. The core function of counter-based OTP and time-based OTP is Eq. (1) and Eq. (2), where C is the counter counting number of connections between the client and server, T is a counter of time steps elapsed since start of an epoch. As we can see here, the input values C and T of successive logins are deterministic and closely related.

The security analysis of the two algorithms is on basis of random oracle model, assuming the hash function is a truly random function, "meaning its input-output values are indistinguishable from those of a random function in practice".

Unfortunately, this assumption does not always hold in practice, especially when successive input values are closely related with only one bit difference as in this case, which leads to at least these unexpected facts:

- If the input is short enough to be put in one block, then the one bit change of two successive input values is not passed in chain to affect other blocks as is designed in HMAC.
- If the input is short enough to be put in one block, say the counter size is 8 bytes, then after 264 calculations, the counter will be reused, and the output OTP will be repeated.
- If the input is more than one block, then if the attacker finds a collision in the first m blocks of two input strings, then he/she can append anything after *m-th* block of an input, but still generate the same output OTPs.
- The closely related input enables some specific types of attacks. There is not much entropy in the values of counter. Counter is incremented by 1 each time, and in most cases, there is only one 1 bit difference in two consecutive values. Consequently, consecutive OTPs are outputs of closely related inputs, and are subject to differential analysis.

  For example, a fault model attack relies on flipping two control bits to reduce the round number to break SHA-512 algorithm. By means of this attack, some certain data block can be extracted completely [16]. Differential Power Analysis (DPA) is another example: The attacker executes the cryptographic algorithm several times with different inputs and gets a set of power consumption traces, each trace being associated to one value known by the attacker. Then a statistical tool is used to compute the correlation between the acquired power consumption traces and predictions made by the attacker [6] [1].

With this knowledge, it is possible for some attacker to design some differential attacks to guess with large probability the output of the next input that is only one bit different than the current or previous input. For SHA-0, it is already known that a change in one bit can be corrected by complementary changes in six other bits [2]. For SHA-1, if it is truly random, 280 operations should be required to find a collision (according to Birthday Paradox), but it is proved that in reality with 263 operations, a collision can be found [18].

Moreover, recent analysis shows the weaknesses of crypto processors as root of secret and passwords generation [13]. An electromagnetic-wave side-channel issue was discovered on NXP Smart MX / P5x security microcontrollers and A7x secure authentication microcontrollers, with CryptoLib through v2.9 [11]. In [3], Cohney *et al.* empirically evaluate the side-channel resistance of common PRG implementations and find that hard-learned lessons about side-channel leakage from encryption primitives have not been applied to PRGs, at all abstraction levels. These findings make the differential attacks and many others easier.

In light of these facts, instead of relying on security of hash functions, we should eliminate possibility of differential attacks by randomizing the input to the hash function. The notion to increase security of a hash function by randomization is presented in some previous works. For example, in [15], the author proposes an algorithm named AKG (Algorithm of Key Generator) to use some tables to generate random keys instead of fixed key to be used in HMAC. There are 9995 tables each one consists of 9995*9995 cells. Each table uses algorithm NRG (Number Random Generator) to get the random numbers from 0 to 9994 that are used as indexes into the table to determine random keys. This idea can be applied in the context of OTP generation, but its drawback is large amount of memory consumption. Since we are considering a multi-factor authentication method that can be run on mobile devices, we are concerning about limited memory resource, and would try to propose some algorithms that are more memory-efficient than this algorithm.

B. Dictionary Attack

RFC4226 [8] specifies that K in Eq. (1) is from a master key that "will be stored at the server only" (naturally the password) and a "public piece of information that identifies uniquely the HOTP device", and "The HOTP client (hardware or software token) increments its counter and then calculates the next HOTP value HOTP client. If the value received by the authentication server

matches the value calculated by the client, then the HOTP value is validated.

In this case, "the server increments the counter value by one." and "A counter *C* is initialized to 0 ..... Both maintain a counter *C*, initially zero, and the user authenticates itself by sending *ALG(K;C)* to the server. The latter accepts if this value is correct."

Being aware of this implementation on authentic server and client, the attacker can launch a dictionary attack as follows:

1. Calculate a tentative key *K* from one of the most popular passwords.
2. Increment a counter *C* from *0* to a moderate number.
3. Calculate HMAC values and corresponding OTPs for this *K* and all *C*'s in Step 2.
4. Repeat Step 1-3 for different passwords, as long as computational and storage resources permit. Get tables such as Table I for password "123456", Table II for password "abc123", and Table III for password "password", and so on.
5. Sort the OTPs in all tables in ascending order and maintain a lookup table as is shown in Table IV. It's reasonable for use to assume that the OTPs are not kept secrets after being used since they are supposed to be used only once. In reality, the OTPs can be caught by capturing network traffic, shoulder-surfing, hidden cameras etc.
6. When the initial key *K* is weak and the counter *C* is small, the generated OTPs from *HOTP(K, C)* can easily be searched and found in a dictionary similar to Table IV. For example, the attacker collected three consecutive used OTPs: 076435, 939005, and 715483. Looking up these OTPs from Table IV, it's with very high probability that the password used to calculate the OTPs is "123456". By analyzing used OTPs, user's secret key can be cracked, and the attacker can calculate subsequent OTPs for future use.

Table I: Pre-Calculated OTPS for Password "123456"

Password	Counter	HMAC Value	OTP
123456	0	0f1c196aed1a8b5dc24798c4c4be7208ebad7fa5	341122
123456	1	78dfb85c00bad9eb98ff979426ed54477d56c28d	239869
123456	2	0fd90974e6b200575a71f230b29650de728b4575	883162
123456	3	7dc048261bfb4b19d81710eb3c1ed55c2f393dfd	299247
123456	4	b4f02794d6a8e9c25010f4ae7e54a461d0d6d30f	076435
123456	5	e9342f220ac9ba30fde8ef830e82aa17b13e9035	939005
123456	6	6d2fcbe53a83ca1bc61302a4fb08972026670c54	715483
123456	7	d2540abfe132f63eddaca43ecb6da2e27d43d1de	268547
123456	8	40d6ee0a742711aa5326e9f9a73f8ee6679c1d47	239869
123456	9	ebcb5f3bfb33f5ad3a76c7ed7931b44bedbc1c6d	092521
123456	...	.....	.....

Table II: Pre-Calculated OTPS for Password "abc123"

Password	Counter	HMAC Value	OTP
abc123	0	9ba6e3dc93ed05fd59b5927b245ad88cf207ff4e	632007
abc123	1	9edce9cc93d5a466298171b3163d04d0f47cce00	794252
abc123	2	5b512c411d57b7f2f9c9ae34172db44f9a759361	854749
abc123	3	c41f3780c7a3a13dd5eee13bc3f428483ab9a473	083553
abc123	4	058f32b1144f6cad2a854e0900832a403d0cd49f	742804
abc123	5	49efc0a00917a4d2642a3ed841e0c21bd54374df	961268
abc123	6	af145547702c4d96d555edecd1d8c86c06c76d07	079917
abc123	7	2c90ffaa9934d57f30fa6716f80c9b8e96eaa8d5	406960
abc123	8	954c1599d1dab465c3b6587d7ab33eba1b32911d	748891
abc123	9	1e358f6338d633c9df5ea5c7c3ade9514ba84811	589496
abc123	...	.....	.....

Table III: Pre-Calculated OTPS for Password "Password"

Password	Counter	HMAC Value	OTP
password	0	6ec730a828884e4f6491a5c1ab94687ba57a6702	326792
password	1	945f05c84f1260dd906529834736878e7451c175	338064
password	2	d686c76063e2ec032aa2f776261c074dd6dcafd8	323254
password	3	6ec522344a59097703809c5d973b62c154c9b24e	837321
password	4	6a6a6478faf0c408ab570ca0c0b14131f9a7564b	499201
password	5	0034ab752aa45c7766d14622fae593a139cc23c5	039654
password	6	c8cd7448e33b558c5a26da7d3a437764294389df	425865
password	7	07c9c291390a85a5830df909562af5bfd48f5e94	990885
password	8	192571fbab09e4dea98c2fc45d4b9e32caabcb97	169775
password	9	5430745170a94b81e4a4f135dcb85bc975c23071	929392
password	...	.....	.....

Table IV: Lookup Table from OTPS to Password

OTP	Password	Counter
039654	password	5
076435	123456	4
079917	abc123	6
083553	abc123	3
092521	123456	8
169775	password	8
239869	123456	1
268547	123456	7
299247	123456	3
323254	password	2
326792	password	0
338064	password	1
341122	123456	0
406960	abc123	7
425865	password	6
499201	password	4
589496	abc123	9
632007	abc123	0
715483	123456	6
742804	abc123	4
748891	abc123	8
794252	abc123	1
837321	password	3
854749	abc123	2
883162	123456	2
899501	123456	9
929392	password	9
939005	123456	5
961268	abc123	5
990885	password	7
.....	.....	...

IV. Proposed Solution

We see in Section III vulnerabilities of existing and popularly used OTP methods. To deter differential attacks and dictionary attacks to OTP algorithms, we suggest adding randomness into the input of the hash function in use, in order to break determinism that is essential in the pre-calculated tables.

A. Random IV Based Counter

We can initialize counter *C* from a random IV that is determined the first time the authentic user registers into the server, and start counting from that point in subsequent logins.

As is illustrated in Fig. 1,

OTP1 = HMAC(K, IV),

OTP2 = HMAC(K, IV + 1),

OTP3 = HMAC(K, IV + 2),

.....

The client and server maintain the IV as if it were the counter as in RFC 4226. If a login is successful, both sides increment the counter and use it in the next OTP calculation.

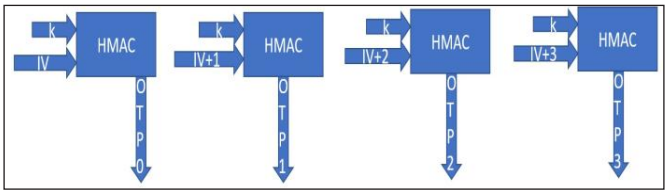

Fig. 1: Random IV Based Counter in HOTP

B. Chaining IV

Further from above Random IV based Counter, we can use last-time output hash code as a random IV for this time OTP calculation. As is illustrated in Fig. 2, first OTP

OTP0 = HMAC(K, IV)

is random, and is used as random IV in

OTP1 = HMAC(K, OTP0).

Similarly,

OTP2 = HMAC(K, OTP1),

OTP3 = HMAC(K, OTP2),

.....

The client and server start with a random IV that is agreed at very first time. In later communications, if a login is successful, both sides store the HMAC output as if it were the counter as in RFC 4226, and use it in the next OTP calculation.

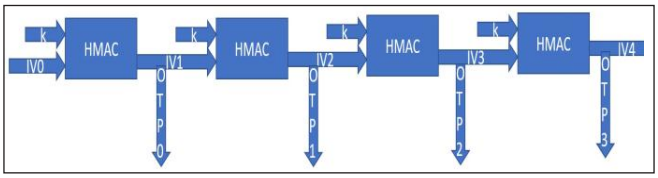

Fig. 2: Chaining IV in HOTP

V. Security Analysis and Discussion

Using random IV based counter, OTPs generated with same passwords are not deterministic any more, and are not subject to dictionary attacks. As is shown in Table V and Table VI, IVs used do not start from 0, and are different and unpredictable. Even for a same simple password, taking large space of IVs into account (compared to small and predictable counters), it is infeasible to generate a table similar to Table IV to search for OTPs.

Nevertheless, it is noted that random IV based counter mode is still subject to differential attacks theoretically, because the inputs to HMAC function are still close related. Using chaining-IVs, OTPs generated with same password are not deterministic any more, and the inputs to HMAC function are not related any more, as is shown in Table VII. The generated OTPs are not subject to differential attacks or dictionary attacks. The security level is even higher.

Table V: OTPS for Password "123456" with Random IV 66832847

Password	Counter	HMAC Value	OTP
123456	66832847	27a724ac2357a0dfaf53c951817b33485d2cab27	325769
123456	66832848	b1f960aa02a4ec90aadd51b2fd3cbbd1c034457d	941888
123456	66832849	3203885ad6779c642a7f9dc73a135390bdf51ef8	006535
123456	66832850	4c322a3a00944db4093dc742ec188fa741990655	636681
123456	66832851	8a8e7504a0f34f4a6ad88884ba77f378a1569739	341882
123456	66832852	196c454fcca5c3dbd3e1590fbfcd8804646ea6d5	592787
123456	66832853	ee665f9f47c3c0795b996dc7d0e61b16e1ebbc62	274115
123456	66832854	b02510cc7c8845821c29a1223351f92ceab591b9	425907
123456	66832855	40d50701218912682d2fd7fc50d43a1e37babd7d	094967
123456	66832856	2ae7e4930d82546961a6e5523547399d5969450a	886407
123456	...	.....	.....

Table VI: OTPS for Password "123456" with Random IV 82791155

Password	Counter	HMAC Value	OTP
123456	82791155	f330b6e96289e0c66230080fbced2cbf4b18bfa4	203142
123456	82791156	ffc2432069a72d75354026fe6e494d04f8da0292	197671
123456	82791157	daf6a566a11e021e77926caf87e5c4abac64298f	718121
123456	82791158	8a80b79d277bbc8513a1972e6a2c4a9f7292088d	087986
123456	82791159	69a614024dc383c615461cfa55bb7aa1aabd0ac4	658886
123456	82791160	142ffe5de83c9fe9db32b37537bbc04e8ff27ec5	113051
123456	82791161	b98684fd7bf50add2ba8036e572c91eeab40b49d	761323
123456	82791162	9002090429c331149ae093f38e077e3c98da755e	900506
123456	82791163	6dfb3340791dd4e25ec7f2bdb12d01e20158779c	033186
123456	82791164	5fd9bc82b09415cba6371d70e0b92e4ab1ea0ec4	011275
123456	...	.....	.....

As for TOTP, since T in Eq. (2) is the number of time steps since the Unix epoch, and is not starting from 0, it is better than the counter C used in Eq. (1). But T is still predictable for a given time period, and successive values of T are still related; thus, TOTP is subject to differential attacks. In this regard, we suggest T should be replaced with $T \oplus IVn$, so that $TOTP = HOTP(K, T \oplus IVn)$ where IVn is what is explained above.

The existing HOTP and TOTP both require a "random oracle model". Security results in the random oracle model do not necessarily imply security in the real world. The proposed solution realizes the vulnerabilities of relying on "random oracle model" in the real world, and mitigates this reliance, thus is more secure in the real world.

Table VII: OTPs for Password "123456" with Random IV 46db991 in Chaining Mode

Password	Counter	HMAC Value	OTP
123456	46db991	0b3962302e627a6de78a511eeb5a848edc83cea8	117982
123456	0b3962302e627a6de78a511eeb5a848edc83cea8	d3bac25f730579c462c9413a9fef92e9833f519f	209105
123456	d3bac25f730579c462c9413a9fef92e9833f519f	9ae69dd0bea5537cdf10443d3440334f9f309097	993476
123456	9ae69dd0bea5537cdf10443d3440334f9f309097	26ec160080f8ecbc2bae215da515d055e8f8d941	381248
123456	26ec160080f8ecbc2bae215da515d055e8f8d941	e1b7d4275c0d275fee8d799280d72549f1975fc3	344103
123456	e1b7d4275c0d275fee8d799280d72549f1975fc3	457ac6cf38e1405c5d750c91aa55de691493e544	286172
123456	457ac6cf38e1405c5d750c91aa55de691493e544	c8a4c2544ddfb66d7a981b13c57cdbd1d6179a12	821215
123456	c8a4c2544ddfb66d7a981b13c57cdbd1d6179a12	81bb3786ef378441a545a7ea7a85b0bac3617a31	494767
123456	81bb3786ef378441a545a7ea7a85b0bac3617a31	a9ec8730bd6120b8aad4f036900e73ee93a5a4ea	624014
123456	a9ec8730bd6120b8aad4f036900e73ee93a5a4ea	919bf7350b4088c6eb12fea8adba20e5370632fb	474016
123456	...	.....	.....

VI. Conclusion

Authentication is an indispensable and paramount component of an online service. Password-based authentic-cation itself is not secure enough anymore. One-time passwords are used in many online services as supplementary authentication method. However, we see vulnerabilities in existing TOTP and HOTP authentication methods used by many online services. We demonstrate vulnerabilities and potential attacks to them. In light of this, we propose a simple solution to overcome the vulnerabilities, by adding randomness and unpredictability to TOTP and HOTP algorithms.

With the proposed solution, security level of TOPT and HOTP is improved, while application interface and performance remain unchanged. We believe this is a novel, feasible, and promising solution for online service authentication in current situation and the near future.

References

[1] S. Belaid, L. Bettale, E. Dottax, L. Genelle, and F. Rondepierre, "Differential power analysis of HMAC SHA-2 in the Hamming weight model," In *2013 International Conference on Security and Cryptography (SECRYPT),* 2013, pp. 1-12.

[2] F. Chabaud, and A. Joux, "Differential collisions in SHA-0," In *CRYPTO*, 1998, pp. 56-71.

[3] S. Cohney, A. Kwong, S. Paz, D. Genkin, N. Heninger, E. Ronen, and Y. Yarom, "Pseudorandom black swans: Cache attacks on CTR DRBG," In *2020 IEEE Symposium on Security and Privacy (SP)*, 2020, pp. 1241-1258, doi: 10.1109/SP40000.2020.00046.

[4] Crackstation, "Crackstation's password cracking dictionary," 2016. [Online]. Available: https://crackstation.net/buy-crackstation-wordlist-password-cracking-dictionary.htm, accessed on 01/10/2023

[5] FIDO Alliance, "FIDO alliance input to the national institute of standards and technology (NIST)," 2017. [Online]. Available: https://www.nist.gov/system/les/documents/2020/09/08/Comments-800-63-009.pdf, accessed on 01/06/2023

[6] L. Guo, L. Wang, and D. Liu, "A chosen-plaintext differential power analysis attack on HMAC - SM3," In *2015 11th International Conference on Computational Intelligence and Security (CIS),* 2015.

[7] D. Klein, "Foiling the cracker: A survey of, and improvements to, password security," *Programming and Computer Software*, vol. 17, pp. 1-10, 1992.

[8] D. M'Raihi, M. Bellare, F. Hoornaert, D. Naccache, and O. Ranen, "RFC 4226 - HOTP: An HMAC-based one-time password algorithm," *IETF RFC 4226*, 2005.

[9] D. M'Raihi, S. Machani, M. Pei, and J. Rydell, "RFC 6238 - TOTP: Time-based one-time password algorithm," *IETF RFC 6238*, 2011.

[10] C. Mulliner, R. Borgaonkar, P. Stewin, and J.-P. Seifert, "SMS-based one-time passwords: Attacks and defense," In *2013 10th International Conference on Detection of*

Intrusions and Malware, and Vulnerability Assessment, 2013.

[11] NIST, "NIST national vulnerability database," 2017. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2021-3011, accessed on 01/12/2023

[12] NIST, "NIST special publication 800-63B: Digital identity guidelines," 2017. [Online]. Available: https://pages.nist.gov/800-63-3/sp800-63b.html

[13] M. Schwarz, M. Lipp, C. Canella, R. Schilling, F. Kargl, and D. Gruss, "ConTExT: A generic approach for mitigating spectre," In *2020 Symposium on Network and Distributed Systems Security (NDSS)*, 2020.

[14] Sedicii, "Preventing mobile phone SIM swap frau," 2017. [Online]. Available: https://www.sedicii.com/

[15] S. H. Shaker, "HMAC modification using new random key generator," *Iraqi Journal of Computers, Communication and Control and Systems Engineering*, vol. 14, no. 1, pp. 72-82, 2014.

[16] A. Shoufan, "A fault attack on a hardware-based implementation of the secure hash algorithm SHA-512," In *2013 International Conference on Reconfigurable Computing and FPGAs (ReConFig)*, 2013, pp. 1-7, doi: 10.1109/ReConFig.2013.6732292.

[17] E. Vasiete, Y. Chen, I. Char, T. Yeh, V. Patel, L. Davis, and R. Chellappa, "Toward a non-intrusive, physio- behavioral biometric for smartphones," In *Proceedings of the 16th International Conference on Human-Computer Interaction with Mobile Devices and Services, MobileHCI'14*, 2014, pp. 501-506.

[18] S. Zhao, "Improvement on security of SMS verification codes," *Software Engineering Research, Management and Applications. Studies in Computational Intelligence Book Series*, vol. 845, 2019, pp. 189-203, Springer (ISBN: 978-3-030-24343-2).