A Comparative Analysis of HOTP and TOTP Authentication Algorithms. Which one to choose?

Lina Lumburovska, Jovana Dobreva, Stefan Andonov, Hristina Mihajloska Trpcheska, Vesna Dimitrova
Faculty of Computer Science & Eng. Ss. Cyril and Methodius University
Skopje, R. N. Macedonia
jovana.dobreva@finki.ukim.mk

Abstract: *Giving the right access, limiting resources, and recognizing a user's identity are important steps that need to be taken into consideration before entering a certain network. These steps are executed by authentication and authorization. In this paper, we put our focus on authentication algorithms HOTP and TOTP as two algorithms for generating one-time passwords. A one-time password is an automatically generated string of characters - a password that is meant to be used only once. This password is only valid for one login session or transaction. Due to its randomness and usage (only once), it leads to higher security outputs, and that is why this type of password is used in authentication algorithms. We will analyse both algorithms and their working way and will present the obtained results and their usage in practice. The main characteristic is that the HOTP algorithm uses only hash functions and the TOTP algorithm uses time above the hash. To check when each algorithm is better to use, we need to know the given environment and circumstances. In this paper, we will try to answer the question" Which one is better at a particular time?". Depending on many factors that we analyse through the sections, we are going to make conclusions that will be useful for future planning of good security passwords.*

Keywords: *authentication, security, one-time passwords, HOTP, TOTP, hashing functions*

1. Introduction

The first step which needs to be taken when we want to access a certain network and the services it offers is the authentication process. Authentication as a process includes recognizing a user's identity, which means determining whether someone or something is who or what it declares itself to be based on a set of credentials. The most common way to check credentials is by checking the username and associated password. If certain credentials (username and password) match the credentials stored within the authentication server, besides authentication there is one more step that needs to be taken, the authorization process that checks if the user has the right permission to access the required services. Authentication and authorization are two processes that are connected, and in many situations, they depend on each other, but in this paper, we will make a comprehensive description and analysis of the authentication process [3, 1].

Username often consists of the individual's first and last name, i.e., it represents the user's identity, and the chosen password is something that is determined by the user and has to remain secret. Often users make Tweak passwords to be easy to remember, which also means they can be easily guessed. Therefore, authentication based on entering username and password does not provide adequate security when accessing systems with sensitive data. If you have noticed, the most common rules for a password are min length of characters, at least one special character, upper and lower letter, and number. However, this leads to higher security and passwords that can hardly be guessed. Due to the vulnerabilities mentioned above, which are brought by authentication, the basis of username and password is changed using a combination of different independent authentication factors. Roughly speaking, these authentication factors can be separated into three groups: something you know, something you have, and something you are. For example, when using a username and password, we are talking about the first group - something you know because the user must know his credentials to execute his request. Then, when talking about something that a user owns, we have the second group - something you have, such as some physical device, i.e., one-time physical password generators and smart cards. The last group - something you are described with the biometric parts of the human body such as fingerprints, face and voice patterns, or any other parts of the human body [5].

In recent years, we have often talked about the principles of strong authentication, which most often refers to two-factor authentication and multi-factor authentication. This is called strong authentication because the users prove their identity with at least two independent authentication methods that belong to different groups. Two-factor authentication can also be used with username and password and ensures a higher level of service security. This authentication method makes it very difficult for attackers to gain access to user devices for online accounts because just knowing the user's password is no longer enough to verify the identity since the attacker does not have the other part of the authentication process. However, as a part of two-factor authentication, we usually talk about the security of devices that a user owns, and the software is responsible for the identification [16]. In addition to secure devices in physical form, software solutions are also easier to use. These solutions generate passwords according to some predefined algorithms, which will be explained in the following sections.

The paper is partitioned into a few sections, structured as follows. The second section is providing a brief introduction to authentication algorithms and one-time passwords in general. The HOTP and TOTP algorithms are explained in the third and fourth sections accordingly, while in the fifth section we give a brief overview of their similarities and differences. In the sixth section, we present the practical usage of both algorithms supported with real-life examples, within a simple analysis with improvements of a more modern hash functions. In the end, we gave a conclusion of the analysis of both algorithms.

2. Authentication Algorithms and One-time Passwords

This section describes the implementation and differences between HOTP (HMAC-based One-Time Password) and TOTP (Time-based One-Time Password) algorithms, as two different algorithms for generating one-time passwords. We can see that both of them have something in common besides the fact that they are authentication algorithms. The main similarity between the two algorithms is the generation of a one-time password. As its name says, a one-time password is an automatically generated string of characters - a password that is meant to be used only once. This password is only valid for one login session or transaction, for a limited period. Due to its randomness and usage, it leads to higher security outputs and that is why this type of password is used in authentication algorithms [13, 8].

One of their biggest advantages is that these passwords are static, and they are resistible to replay attacks, which means that if a potential attacker records some past one-time passwords, he cannot use them to log in to the system afterward, because the

passwords will no longer be valid. The one-time passwords are beneficial when a user has logged in to more than one system. Therefore, if he uses a one-time password, he will need a different password for each device and if a hacker finds one of the passwords, he cannot use that password to login to the other systems [7].

To sum up, the security of the one-time password method is high, but it does not mean that this method is unbreakable. The one-time password methods are vulnerable to a man-in- the-middle attack, but we will not dive into details about the attack's side as it is not part of the scope of this paper.

3. HOTP Algorithm

HOTP (HMAC-based One-time Password) is a one-time password algorithm based on HMAC (hash-based message authentication code). This algorithm was found by David et al. in 2005 [9, 25].

Firstly, we will briefly describe hash functions, since this algorithm is based on hash functions as one of the fundamental building blocks of modern cryptography. These are functions that convert an arbitrarily long message into a sequence of bits of a certain length. In combination with asymmetric algorithms, they are the most commonly used for encryption and digital signing. Modern hash functions must be deterministic and computationally efficient. Also, known as one-way functions, which means that the hash value is calculated from the input, and it is not possible vice versa i.e., to efficiently calculate the input from a given hash output value. The most frequently used hash functions belong to the SHA (Secure Hash Algorithm) family and these features additionally have the property that a small change in the message causes a well- perceived change density (the old and the new digests do not look similar) [26].

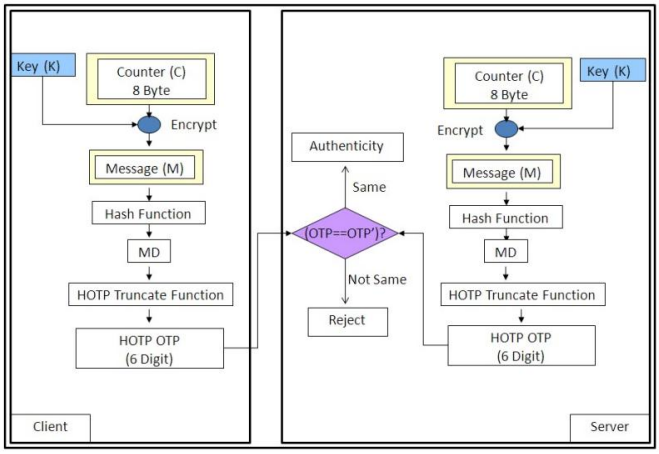

Fig. 1 *Scheme of HOTP algorithm [23]*

Although, hash functions are the message authentication mechanism that plays an important role in the algorithms for generating one-time passwords. Hash-based message authentication code (HMAC) is a mechanism for calculating a message authentication code involving a hash function such as MD5 (Message Digest), SHA-1, or other hash function in combination with a secret key. HMACs are almost like digital signatures. They both enforce integrity and authenticity. They both use cryptographic keys, and they both employ hash functions. The main difference is that digital signatures use asymmetric keys, while HMACs use symmetric keys. HMAC uses two passes of hash computation. The secret key is first used to derive two keys – inner and outer. The first pass of the algorithm produces an internal hash derived from the message and the inner key. The second pass produces the final HMAC code derived from the inner hash result and the outer key. HMAC is described using the following equation:

$$\mathrm{HMAC}(K, m) = \mathrm{H}((Kj \oplus opad) \parallel \mathrm{H}((Kj \oplus ipad) \parallel m)) \quad (1)$$

where H is the hash function, m is the message to be authenticated, K is the secret key, K_j is a block-sized key derived from the secret key, K; either by padding to the right with 0s up to the block size, or by hashing down to less than or equal to the block size first and then padding to the right with zeros, *opad* is the block-sized outer padding and *ipad* is the block-sized inner padding [19, 6].

HOTP is an algorithm for generating a one-time password that works based on a message authentication mechanism. It can be implemented by any hardware or software developer to create an interoperable authentication device or software. An incremental counter and a static symmetric key, known only to the security device and the verification server, are used to generate one-time passwords. For the HOTP value, it is used the HMAC-SHA-1 algorithm that accepts an arbitrarily large data set and returns a value of a certain length of 160 bits as an output value:

$$\mathrm{HOTP}(K, counter) = \mathrm{HMAC\text{ - }SHA\text{-}1}(K, counter) \quad (2)$$

where K is the common secret between the client and the server and counter is the 8-byte counter value. This counter must be synchronized between the HOTP generator (client) and the HOTP validator (server). The value obtained must then be truncated using dynamic truncation function (DT) specialized for HOTP. So, from the first step the HOTP value is 160bits (20-byte string). Then, we generate a 4-byte string by using DT function such

$$Sbits = \mathrm{DT}(\mathrm{HMAC\text{ - }SHA\text{-}1}(K, counter)) \quad (3)$$

and it returns a 31-bit string. We recompute the HOTP value as

$$Snum = \mathrm{StToNum}(Sbits) \quad (4)$$

where we convert the string to number. The purpose of the dynamic offset truncation technique is to extract a 4-byte dynamic binary code from a 160-bit (20-byte) HMAC-SHA-1 result. Due to the fact that the final value must be at least 6-digits (or 7 or 8 or more) we use modular operation $Snum \bmod (10^{digit})$, where *digit* is the selected number of digits to be outputted (6,7,8 or more) [10].

Although the server's counter value is only incremented after a successful HOTP authentication. Also, 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 the token might be out of synchronization, and resynchronization of the counter is required. This can be solved by setting a parameter s on the server, which defines the size of the look- ahead window. The server can recalculate the next s HOTP- server values and check them against the received HOTP client. Synchronization of counters in this scenario simply requires the server to calculate the next HOTP values and determine if there is a match. Optionally, the system may require the user to send a sequence of (per example, 2, 3) HOTP values for resynchronization purposes, since forging a sequence of consecutive HOTP values is even more difficult than guessing a single HOTP value [10].

The HOTP algorithm is vulnerable to brute force attacks due to the use of the truncated HMAC-SHA-1 value, which means that the authentication server must be able to detect and prevent such an attack. The first option is to use a limit on the number of requests by setting the damping parameter D, which defines the maximum number of possible attempts to validate a single password. Another option is to implement a delay scheme that works so that after the *i*-th failed attempt, the authentication server waits for the increasing number of seconds D_i. Therefore, if $D = 5$, after the first attempt, the server should wait for example 5 seconds, and after the second failed attempt 5 * 2 = 10 seconds and so on. Authentication between the client and the certifier must

take place via secure channels and using appropriate security mechanisms, such as using a session identifier to protect the user's privacy and avoid repetitive attacks. The main problem of the HOTP algorithm is that the generated password is valid for a long time or at least until the next authentication attempt, which means that if the attacker overrides the password, he can use it at any time. This shortcoming is solved by the TOTP algorithm within which every password has limited validity.

Using the parameter *D*, the security of the algorithm is increased because after each unsuccessful attempt the number of seconds to wait is increased. On the other hand, the usage of truncation has its disadvantages because it leads to more insecure implementation. This can be handled if the value of the digits is increased (for example, more than 8), so it becomes less vulnerable to brute force attacks in real time. The security also depends on the parameter s which ensures the server does not go on checking HOTP values forever (causing a denial-of-service attack) and also restricts the space of possible solutions for an attacker trying to manufacture HOTP values.

SHA-1 has a weakness concerning collisions [21]. But HMAC resistance does not rely on resistance to collisions. Indeed, HMAC is proven secure as long as the hash function which it uses is a Merkle-Damgard function which itself relies on an internal "compression function" which behaves like a PRF(Pseudorandom function family). The known weakness of SHA-1 voids the proof, but nobody knows how to turn that into a weakness on HMAC/SHA-1. Empirically, we have the example of MD4 (Message Digest): MD4[18] is extremely broken with regards to collisions, with a near-zero cost (computing a collision for MD4 takes less time than actually hashing the two colliding messages to verify that it is, indeed, a collision), and HMAC/MD4 is also broken, but with a quite non-trivial cost of 258 plaintext/MAC pairs (and that's a forgery attack, not even a key recovery attack), making it utterly non-applicable in practice. If we have the same kind of ratio for SHA-1, then HMAC/SHA-1 is still very safe. To sum up, using the default SHA-1 for the HOTP is still a valid option.

4. TOTP Algorithm

TOTP (Time-based One-time Password) is a one-time password algorithm which uses the current time as a source of uniqueness [11, 24]. The TOTP algorithm is a version of the HOTP algorithm that works based on time and is calculated according to the common function (2), except that in this case the counter is replaced by the value of T, which is derived from the time reference. The HMAC-SHA-256 or HMAC-SHA-512 algorithm may be used instead of the HMAC-SHA-1 algorithm to calculate a one-time password. In general, TOTP is defined as:

$$\text{TOTP} = \text{HOTP}(K, T) \tag{5}$$

where T represents the number of time steps from the initial time counter T_0 and the current UNIX time. More specifically, T is defined as:

$$T = (\text{CurrentUNIXTime} - T_0)/X \tag{6}$$

where X represents the number of time steps in seconds (default value for X is 30 seconds) and is a system parameter, and T_0 is the UNIX time at which we start counting time steps (default value is 0) [12].

When the authentication server receives a one-time password (OTP), it does not know precisely when it was created. Due to network delays, the gap (number of time steps from time T0 onwards) between the time when OTP was created and the time the verification system receives the OTP may be too large. Therefore, the verification system should typically set a policy for an acceptable OTP transmission delay time. The larger this interval is, the more exposed it is to the possibility of an attack. It is strongly recommended that the time lag should be less than the size of a one-time step.

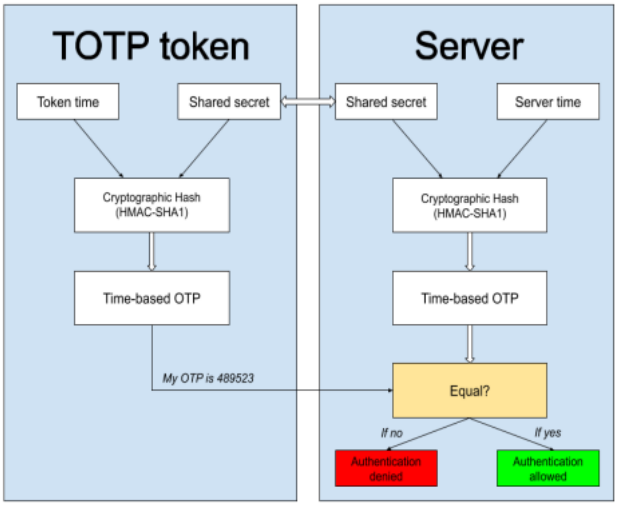

Fig. 2 *Scheme of TOTP algorithm [22]*

The larger interval that was mentioned before is, decreases the security of the algorithm. According to equation (3), TOTP is calculated from HOTP which means that the security of this algorithm depends on the HOTP parameters. Choosing the right HOTP parameters and how they increase/decrease security was described in the third section and those rules are the same for the TOTP algorithm.

5. The Main Differences between Algorithms

The previous two sections were written to explain both algorithms in detail and this section will give an overview of both algorithms and explain what the difference between algorithms is and compare them according to their possibilities.

5.1.HOTP Analysis

1) The algorithm uses a counter, as one of the goals is to include an algorithm for generating one-time passwords based on the HMAC value in devices to store large amounts of data, such as USB keys and SIM cards.
2) The algorithm should be economical in terms of implementation in hardware by minimizing the requirements related to battery, the number of buttons, the computational power, and the size of the screen in case the device uses it [10].
3) The algorithm works with tokens that do not support any numeric input but can also be used with more advanced devices such as secure PIN inputs.
4) The value displayed on the device should be easy to read and enter; it should also be of reasonable length but not less than six characters. It is also desirable that the HOTP value is exclusively numerical, which allows easy entry into devices such as telephones.
5) The mechanism for resynchronizing the counter is suitably chosen (a good example of how to choose it is described in section 3). Each time the user requests it, it should be increased on the device, and on the server, the counter is increased only upon successful authentication based on a valid HOTP value.
6) The algorithm uses a symmetric key known only to the authentication server and client. To ensure greater security against some known attacks on the symmetric keys, and by the fact that this key is hardcoded on the

device, the recommendation is to use the symmetric key that is at least 128 bits long.

5.2. TOTP Analysis

1) To generate a TOTP value, the client (physical or software one-time password generator) and the certifier (authentication or validation server) must know whether they are capable of obtaining the current UNIX time. The most common way to obtain this is to count the number of seconds elapsed since midnight UTC of January 1, 1970 [12].
2) The client and the certifier either share a key or have information based on which they generate a key (for example, using symmetric cryptography).
3) The algorithm uses HOTP as the main building block.
4) The client and the certifier should use the exact value of the time step X, which illustrates the duration of the password in seconds.
5) Keys can be stored on devices with tamper-resistant properties that are protected against unauthorized access. Those devices use hardware encryption, are activated only when required: the key is decrypted when needed to verify an OTP value and re-encrypted immediately to limit exposure in the RAM for a short period.

5.3. Analysis Conclusion of Both Algorithms

HOTP (also known as Event-based OTP) is the original one-time password algorithm and relies on two pieces of information. The first is the secret key, which is known only by the token and the server that validates submitted OTP codes. The second piece of information is the moving factor, which in event-based OTP, is a counter. The counter is stored in the token and on the server. The counter in the token increments when the button on the token is pressed, while the counter on the server is incremented only when an OTP is successfully validated. TOTP is based on HOTP where the moving factor is time instead of the counter. This is the main difference between both algorithms TOTP uses time in increments called the timestep, which is usually 30 or 60 seconds. This means that each OTP is valid for the duration of the timestep [12]. Both OTP schemes offer single-use codes, but the main difference is that in HOTP a given OTP is valid until it is used, or until a subsequent OTP is used. In HOTP, there are several valid "next OTP" codes. This is because the token button can be physically pressed, and the counter on the token immediately increases. However, this action is done without the resulting OTP being submitted to the validating server. For this reason, HOTP validating servers accept a range of OTPs. Specifically, they will accept an OTP generated by a counter within a set number of increments from the previous counter value stored on the server. This range is referred to as the validation window. If the token counter is outside the server's range, the validation fails, and the token must be re- synchronized. So clearly, in HOTP, there is a trade-off to be made. The larger the validation window is, the less likely the chance to re-sync the token with the server can be done, which is inconvenient for the user. Notably, the larger the window is, the greater the chance of an adversary guessing one of the accepted OTPs through a brute-force attack is. In contrast, in TOTP, there is only one valid OTP at any given time - the one generated from the current UNIX time [4].

6. Practical Usage

One-time password generators are divided into HOTP and TOTP generators. They are divided into physical devices and software generators, where physical devices are usually smaller devices that create a one-time password when the button is pressed or after entering the PIN code and display it on the screen. The user then logs in using this password. Software generators work on the same principle, except that in this case an application such as e.g., Google Authenticator, which mimics the behavior of a physical generator, is installed on a device such as e.g., mobile phone.

Fig. 3. *(a) OTP generator protected by PIN (b) OTP generator not protected by PIN [2]*

One way to generate an OTP is by using grid cards or transaction numbers of lists. These methods offer low investment costs but are slow, difficult to maintain, easy to replicate and share, and require the users to keep track of where they are in the list of passwords. Another, more convenient way for users is to use an OTP token, a hardware device capable of generating one-time passwords. Some of these devices are PIN protected which offers an additional level of security. The user enters the one-time password with other identity credentials, and an authentication server validates the logon request. Although this is a proven solution for enterprise applications, the deployment cost can make the solution expensive for consumer applications. Because the token must be using the same method as the server, a separate token is required for each server logon, so users need a different token for each Web site or network they use. Last, more advanced hardware tokens use microprocessor-based smart cards to calculate one-time passwords. Smart cards have several advantages for strong authentication, including data storage capacity, processing power, portability, and ease of use. They are inherently more secure than other OTP tokens because they generate a unique, non-reusable password for each authentication event, store personal data, and do not transmit confidential or private data over the network [17, 15].

One-time password generators are one of the most commonly used authentication factors today. No matter how secure the use of such devices may seem at first glance, this method of two-factor authentication has some vulnerabilities. One of the main problems is the possibility of a man-in-the-middle attack in real-time, where a real-time attacker misuses the data of a user who wants to log in to a service. The attacker imitates the appearance of the target website and thus obtains all the necessary authentication data of the user when he logs in to the service on the fake website. This information is then immediately transmitted to a legitimate website and allows the attacker to report on behalf of the victim. Such attacks can be largely prevented by the user never using the public network to access sensitive data, and the use of VPN provides even better security, as communication takes place through a secure tunnel. Also, another useful comment here is never to put your secret data on the service that communicates over the unprotected channel. Using hypertext transfer protocol secure (https) is always the right choice.

6.1 Improvements with SHA-256

In this subsection, we focused our research on analyzing HOTP and TOTP algorithms with HMAC-SHA256. We explained that there is no mistake if both algorithms are used with HMAC-SHA-1, but because there are more modern hash functions, in this subsection we decided to expand our sights and we give a short overview about the performance analysis if both algorithms are used with HMAC-SHA-256.

Hash function SHA-256 comes from the family SHA-2, which is an upgraded and more secure version of the hash function SHA-1. The SHA-2 family consists of more hash functions, which depends on

the output they provide, and in this analysis, we chose the SHA-256 which outputs 256 bits. In this case, we can see the difference between SHA-1 where we had 160 bits compared to 256 bits. This is the first indicator where the security of the hash function is increased due to the extended length. For example, if we compare according to the brute force attack, where L is the number of bits in the message digest, finding a message that corresponds to a given message digest can be done in almost 2^L evaluations. SHA-1 has 160 bits message digest and SHA-2 has 256 bits message digest. Accordingly, SHA-1 has 2^{160} evaluations and SHA-2 has 2^{256} evaluations to find the hash value. This may seem a small difference, but the machine that performs the action will require much more performance to implement such an attack, even though for more than 80 bits it is still impossible. Then, we have the problem with the collision, finding two different messages that result with the same message digest is about $2^{L/2}$ evaluations using the birthday attack. This attack is much faster than the brute force attack, so the evaluations for SHA-1 are 2^{80} (as if 80 bits message digest) compared to the evaluations for SHA-2 which are 2^{128} (as if 128 bits message digest). To sum up, collision is the reason why the security (the strength) provided by the hash function is divided by 2 from the number of bits that the algorithm has as a message digest. SHA-1 has 80 bits security and SHA-2, 128 bits security. Theoretically, to break 80 bits security, a machine may require 1 day, to break 84 bits security, a machine may require 12 days, to break 89 bits security, a machine may require 1 year, and so on. In this example, we can see that there is a clear difference between 80 bits and 128 bits message digest which one more time increases the security of the SHA-2. With much better security, HMAC is more secure with SHA-2 than with SHA-1 [20].

According to the analysis, HMAC-SHA-1 is still valid and secure, but as time passes and new technologies are discovered, it is recommended to switch to more modern and secure versions. HOTP and TOTP in our analysis are used with HMAC-SHA-1, but in the next analysis, we will continue our work on how they will be implemented with HMAC-SHA-256 according to the above-written analysis and also with all the newest lightweight hash functions standardized by the NIST 2021 [14].

7. Conclusion

To sum up, everything that was written and explained in this paper, both algorithms are secure and used in practice with their advantages and disadvantages.

From a secure perspective, TOTP is better to use. Importantly, the validating server must be able to cope with the potential for time-drift with TOTP tokens to minimize any impact on users. The fact of adding an extra factor that needs to be met increases the security of the code. On the other hand, the sending of the one-time codes depends on external factors, such as broadband coverage (for SMS and calls) and internet connection (for email or messaging apps). If the user lacks any of them, the code won't arrive at the user's device, and they will be incapable of entering the code and verifying their identity. In this case, the user will need to ask for an extra code. Even when all the external platforms are working correctly, if the user doesn't enter the OTP quickly, the code won't be valid either. Regarding this matter, HOTPs can be a friendlier way of verifying users, since they are not limited by the timesteps and can enter the code whenever they want to. Unfortunately, this is a less secure option when compared to time-based OTPs. Whatever type of one-time code we are using, we can be sure that multi-step authentication processes are an efficient way of onboarding users. Using one-time passwords is a way of reinforcing forms based on passwords, verifying the user's phone number or email account. The chances of fraud or failure when using one-time passwords in two-factor authentication are positively low.

References

[1] *Authentication Vs Authorization- What's The Difference?*https://www.ilantus.com/blog/authentication-vs-authorization-whats-the-difference/. Accessed: 23.11.2021.

[2] *Authentication devices*. https://www.hidglobal.com/system/files/doc_eol_expired_files/iam-activid-otp-tokens-br-en.pdf Accessed: 20.11.2021.

[3] Marc Briceno et al. *Advanced authentication techniques and applications*. US Patent 10,270,748. Apr. 2019.

[4] *HOTP vs TOTP: What's the Difference?* https://www.microcosm.com/blog/hotp-totp-what-is-the-difference. Accessed: 20.11.2021.

[5] Christophe Kiennert, Samia Bouzefrane, and Pascal Thoniel. "Authentication systems". In: *Digital identity management*. Elsevier, 2015, pp. 95–135.

[6] Hugo Krawczyk, Mihir Bellare, and Ran Canetti. *HMAC: Keyed hashing for message authentication*. 1997.

[7] Ricardo Margarito Ledesma. *Systems and methods for one-time password authentication*. US Patent 10,587,613. Mar. 2020.

[8] Chung-Huei Ling et al. "A Secure and Efficient One-time Password Authentication Scheme for WSN." In: *Int. J. Netw. Secur.* 19.2 (2017), pp. 177–181.

[9] David M'Raihi et al. "Hotp: An hmac-based one-time password algorithm". In: *The Internet Society, Network Working Group. RFC4226* (2005).

[10] David M'Raihi et al. "Hotp: An hmac-based one-time password algorithm". In: *The Internet Society, Network Working Group. RFC4226* (2005).

[11] David M'Raihi et al. "Totp: Time-based one-time password algorithm". In: *Internet Request for Comments* (2011).

[12] David M'Raihi et al. "Totp: Time-based one-time password algorithm". In: *Internet Request for Comments* (2011).

[13] Matthew Nichols. *Generation of randomized passwords for one-time usage*. US Patent 10,282,526. May 2019.

[14] *NIST Lightweight Cryptography Standardization Process*. https://csrc.nist.gov/Projects/lightweight-cryptography/finalists. Accessed: 20.11.2021.

[15] *OATH Authentication Tokens*. https:// cpl. thalesgroup. com/ access - management / authenticators / oath - tokens. Accessed: 20.11.2021.

[16] Aleksandr Ometov et al. "Multi-factor authentication: A survey". In: *Cryptography* 2.1 (2018), p. 1.

[17] *One Time Password (OTP, TOTP) : definition, examples*. https://www.thalesgroup.com/en/markets/digital-identity-and-security/technology/otp. Accessed: 20.11.2021.

[18] Ronald L Rivest. "The MD4 message digest algorithm". In: *Conference on the Theory and Application of Cryptography*. Springer. 1990, pp. 303–311.

[19] M Rogobete and O Tarabuta. "Hashing and Message Authentication Code Implementation. An Embedded Approach". In: *Scientific Bulletin" Mircea cel Batran" Naval Academy* 22.2 (2019), 296A–304.

[20] Marc Stevens. "Real-world Cryptanalysis". Second AMSec Workshop (2019). https://www.amsec.org/wp-content/uploads/2019/10/Stevens.pdf Accessed: 23.112021

[21] Marc Stevens, Pierre Karpman, and Thomas Peyrin "Freestart collision for full SHA-1". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2016, pp. 459–

483.

[22] *TOTP Algorithm Explained*. https://www.protectimus.com / blog / totp - algorithm - explained/. Accessed: 23.11.2021.

[23] Suratose Tritilanunt, Napat Thanyamanorot, and Nattawut Ritdecha. "A secure authentication protocol using HOTP on USB storage devices". In: *2014 International Conference on Information Science, Electronics and Electrical Engineering*. Vol. 3. IEEE. 2014, pp. 1908–1912.

[24] Mariano Luis T Uymatiao and William Emmanuel S Yu. "Time-based OTP authentication via secure tunnel (TOAST): A mobile TOTP scheme using TLS seed exchange and encrypted offline keystore". In: *2014 4th IEEE International Conference on Information Science and Technology*. IEEE. 2014, pp. 225–229.

[25] Andrea Visconti and Federico Gorla. "Exploiting an HMAC-SHA-1 optimization to speed up PBKDF2". In: *IEEE Transactions on Dependable and Secure Computing* 17.4 (2018), pp. 775–781.

[26] *What is a hash function? Definition, usage, and examples*.https://www.ionos.com/digitalguide/server/security/hash-function/ Accessed: 20.11.2021.