Journal of Computer Science

Original Research Paper

One Time Password (OTP) Solution for Two Factor Authentication: A Practical Case Study

[1]Oscar Efrén Acosta Mayorga and [2]Sang Guun Yoo

[1]Centro de Posgrados, Pontificia Universidad Católica del Ecuador Sede Ambato, Ambato, Ecuador
[2]Departamento de Informática y Ciencias de la Computación, Escuela Politécnica Nacional, Quito, Ecuador

Article history
Received: 07-08-2024
Revised: 17-10-2024
Accepted: 23-12-2024

Corresponding Author:
Oscar Efrén Acosta Mayorga
Centro de Posgrados, Pontificia Universidad Católica del Ecuador Sede Ambato, Ambato, Ecuador
Email: oacosta.mayorga@gmail.com

Abstract: Currently, static passwords are no longer secure, they expose accounts and data, more advanced and sophisticated approaches are required to guarantee user authentication and operations in digital systems. In this context, this study investigates how in a practical application, addressing security challenges in user authentication using One Time Passwords (OTP) can strengthen two-Factor Authentication (2FA), the use of a sec factor for authentication has gained popularity and using an OTP adds an extra layer of protection, strengthening security beyond the traditional password. Furthermore, it is important to highlight that currently transactions between existing systems are largely carried out through smartphones with specific applications due to convenience and widespread presence in modern life. The objective of this study focuses on generating an OTP solution for two-factor authentication in a practical case based on an applied research methodology. The application of this methodology results in an effective system, improving two-factor authentication by providing security for user access with static passwords, even if someone gains access to the user's password, they still need the OTP code to access. This article presents an authentication system for 2FA that includes generation and delivery of the access code based on events (HOTP) or time (TOTP) and uses mobile and email applications for OTP code delivery. The entire authentication process from enabling 2FA on the user account to verifying and validating is comprehensively covered.

Keywords: Authentication, Two-Factor-Authentication, 2FA, One-Time-Password, OTP, TOTP, HOTP, OTPAUTH

Introduction

The safety of digital applications that involve users and computers is critical and essential component of the modern digital era. Authentication is vital in the process of verifying and validating user's identity in order to access a system, platform or specific information.

Most software applications in the authentication process are based mainly on the use of usernames and passwords, resulting in an ineffective and vulnerable method of cyberattacks that are increasingly complex and sophisticated Verma *et al*. (2023). Relying solely on static passwords as an authentication method is often not a sufficient mechanism to counter emerging network security risks Mahdad and Saxena (2023). The absence of an additional security layer in authentication process leaves accounts vulnerable and risk of being compromised by threats such as brute force attacks and phishing Williamson and Curran (2021).

One of the most effective methods of increasing security to improve applications access is to implement two-Factor Authentication (2FA) Marmolejo Corona *et al*. (2023). Two-Factor Authentication (2FA), a subset of Multi-Factor Authentication (MFA), enhances security beyond the traditional use of a username and password Williamson and Curran (2021). 2FA requires the user to present two distinct authentication factors to gain access to a system or resource. The first one typically involves a password or PIN known solely by the user. The second factor can be a One-Time Password (OTP) generated by an authenticator device or application Kirvan *et al*. (2023).

One-Time Passwords (OTPs) represent an efficient solution to strengthen two-Factor Authentication (2FA), since generating passwords based on time or counters reduces the risk of unauthorized access to accounts and systems Reese *et al*. (2019). This unique and dynamic form authentication has been implemented in a variety of environments, including educational institutions, military,

government agencies and private companies Kamau and Mgala (2022). Although OTPs have been widely adopted in international settings and certain national sectors, it is worth mentioning that their interaction with this 2FA mechanism at the local level in everyday applications is not so widespread.

Despite the benefits of two-Factor Authentication (2FA), significant challenges still persist in its implementation in multiple contexts, including low awareness about the importance of 2FA, insufficient knowledge and absence of customized OTP solutions than are not economically accessible. All of this increases the risk and vulnerability of information systems, making them more susceptible to possible attacks and unauthorized access Tirfe and Anand (2022). Therefore, the effective deployment of practical two-factor authentication solutions incorporating one time password as an additional security layer is crucial for enhancing information confidentiality. However, while OTP implementations offer an additional layer of security, their architecture on both the client and server side is not free from threats.

The purpose of this study is to improve the security of traditional login systems by means of two-Factor Authentication (2FA) through a practical solution based on One-Time Passwords (OTP). The proposed solution will be evaluated in a practical environment, focusing on its effectiveness in mitigating risks associated with unauthorized access and its efficiency in terms of usability and performance. The solution is based on an applied research methodology, leading to an effective OTP-based 2FA system. Covers the generation of one-time passwords, the methods for their delivery, advantages and disadvantages associated with their use, as well as the solution in an affordable practical environment. This approach improves on two-factor authentication process by providing a secure access method for users with static passwords, even if someone gains access to the user's password, an additional OTP code is required to access the account adding an extra layer of security.

Implementing 2FA in an on-premises environment is important to improve cybersecurity. 2FA through an OTP can help protect a system and data from cyber-attacks Williamson and Curran (2021).

Related Work

Under the context of authentication through Multi Factor Authentication MFA, it is ensured that any method used in one way or another is much more secure than using only a username and password Williamson and Curran (2021). Two-Factor Authentication (2FA) through the use of One Time Passwords (OTP) has been highlighted as an effective and robust method in protecting sensitive data and preventing unauthorized access. When attempting to log in or make a transaction, the user is given a temporary code that must be entered to complete identity verification, this adds an additional layer by reinforcing security and ensuring the individual is who they claim to be Ali and Ismail (2018).

From this perspective and conceptual framework, previous research works such as Marmolejo Corona *et al*. (2023) have shown that 2FA methods are created from Single Factor Authentication (SFA) usually referred as username or email address and access password which increases the security level as an additional layer to verify the user's identity. Following this approach, in Reese *et al*. (2019) several 2FA methods are analyzed, it is reflected that some of these may be less familiar to users, one of them is the hardware-based Universal 2nd Factor (U2F) device. In addition, the referenced study identifies common errors that could exist in the configuration of third-party applications for OTP generation, for example, unsuccessful access attempts when scanning Quick Response (QR) codes or registering access Keys (Keys) that use Time-based One-Time Password (TOTP) M'Raihi *et al*. (2011) or HMAC-based One-Time Password (HOTP) M'Raihi *et al*. (2005). Notably, Reese *et al*. (2019) highlighted people's familiarity with Short Message Service (SMS) authentication due to its ease of use. However, this can lead to security issues as text messages can be intercepted or redirected or delivery delays can occur.

According to Shukla *et al*. (2019) OTP solutions are essential in two Factor Authentication (2FA) process, but threats from client side (such as phishing attacks or device manipulation) and server side (brute force attacks or code interception) highlight the need for additional measures.

Mobile devices play an important role in exchanging information and accessing applications through 2FA and OTP based authentication. Sending SMS is one of the most commonly used methods for two-factor authentication. Users register their mobile number and receive SMS during the system authentication process Ali *et al*. (2020), however, it is possible that this method faces security problems immersed in the roaming of each country or in the visibility of the code on the phone or in SIM identification theft or problems associated with loss or non-delivery of messages Bruzgiene and Jurgilas (2021).

Reese *et al*. (2019); Tirfe and Anand (2022); Yin *et al*. (2020) mentioned that the choice of OTP generation method depends on many factors, such as security level, ease of implementation, cost, portability and ease of use. The most common thing today for 2FA authentication is to use a smartphone, either by sending the code via SMS Matelski (2022) or by using specific mobile applications that generate codes or receive notifications for successful authentication in 2FA Reese *et al*. (2019). Clear examples of mobile applications to generate one-time passwords are google authenticator, microsoft authenticator and FreeOTP. These applications implement the Time-based OTP (TOTP) M'Raihi *et al*. (2011) and HMAC-based

OTP (HOTP) M'Raihi *et al*. (2005) algorithms to generate OTP codes offline, thereby minimizing exposure to spoofing attacks Ozkan and Bicakci (2020).

Email is commonly one of the many methods in which OTP is delivered. A dynamically generated code is sent to the address that the user has registered or linked in the system. Bruzgiene and Jurgilas (2021). This method's security can be compromised if an unauthorized third party has access to the user's email account. In Addition, should be considered for codes to be delayed by sending them to an email address or the possibility of messages being classified as spam, which affects user experience and authentication effectiveness Ma *et al*. (2019).

OTP passwords are generated using different algorithms, one of them is time-based which generates a code that changes periodically Time-based One-Time Password (TOTP) M'Raihi *et al*. (2011) and another based on the security algorithm Hash HMAC-based One-Time Password (HOTP) M'Raihi *et al*. (2005). However, both HOTP and TOTP process require that the client and server must share a secret key, which can create security risks if the server is compromised. Yin *et al*. (2020). Despite this, it is one of the methods that, due to its non-connection dependency characteristic, strengthens and stands out compared to other methods Reese *et al*. (2019).

The unique time-based codes delivery is subject to external factors such as network coverage for text messages and calls and internet connectivity for email or messaging applications. If the user lacks any of these resources the code will not reach to user's device, resulting in the inability to login and verify identity, in this regard, the HOTP method provides a more flexible option for users who want to verify their identity, as they are not restricted by time and can enter the code at any time Lumburovska *et al*. (2021).

Similarly, there are other physical 2FA mechanisms for authentication or transactionality, one of them is to use tokens such as a USB device or a key object, bank card or smart card, even biometric solutions are used as alternatives to reinforce security, they are not widely used since they require an acquisition cost Binbeshr *et al*. (2023).

In short, password less authentication (SFA) methods need to do more strengthen user identity as the cyberattacks increasingly target easily accessible systems Matelski (2022). From this perspective, OTP codes provide an additional security layer by generating temporary unique values that are used only once in the transaction, these codes have various generation methods as indicated in Table (1).

Concerning the OTP generation methods, it is concluded that time-based methods are the most common, they occupy a shared key or token, they are easy to implement and use, some methods are based on mathematical algorithms, they are more secure, but more complex. To implement, other methods are based on hardware and the OTP is generated from the physical device using a mathematical algorithm or a hash function, however the problem with these lies in the cost.

Regarding the delivery method of OTP codes, it can be concluded that SMS is a fast delivery method, the drawback is that the passwords sent by this means can be intercepted or not have the reliability of delivery due to conditions of the telephone network, in addition include shipping costs which is reflected in an increase in expenses. Email is the most common delivery method, but it can take a while to arrive, making it a problem if the user needs to authenticate quickly. OTP mobile applications are a secure and convenient way to deliver passwords, these are generated on the user's device, which makes them more difficult to be intercepted, however, they require a secret key that can be identified if the server is engaged.

Table 1: OTP generation methods

Method	Description	Reference
HMAC-based One Time Password (HOTP)	This uses a hash value and a counter that is incremented each time a password is generated	Reese *et al*. (2019); Ali and Ismail (2018); M'Raihi *et al*. (2005); Bruzgiene and Jurgilas (2021); Yin *et al*. (2020); Lumburovska *et al*. (2021)
Time-based One-Time Password (TOTP)	Uses time, along with a counter, to generate a password	Reese *et al*. (2019); Ali and Ismail (2018); M'Raihi *et al*. (2011); Bruzgiene and Jurgilas (2021); Yin *et al*. (2020); Lumburovska *et al*. (2021)
Physically Unclonable Functions (PUF)	Generates unique keys based on physical characteristics of hardware devices	Uysal and Akgun (2023)
Merkle tree-based One-Time Password (MOTP)	Generate passwords based on Merkle tree structures	Yin *et al*. (2020)
Genetic algorithm	Generates passwords based on evolution and randomness from an initial population	Ali and Ismail (2018)

Implementation of 2FA is becoming more common across platforms. Although there are various works and applications to implement two-Factor Authentication (2FA) systems using OTP codes, the creation of real cases of this type can promote competition and innovation due to the simplicity and usability of user authentication and learning opportunities, therefore, this study helps to be familiar with the concepts and practical implementation of 2FA systems with OTP codes, allows us to be updated on real-world security challenges, in addition, by acquiring new skills and capabilities in understanding 2FA with OTP codes for authentication improves user security awareness. In this study, HOTP and TOTP are used as OTP generation methods and as OTP delivery methods the email, third-party mobile applications and a custom mobile application associated to the user account, all of this in a real-life case focused on 2FA user authentication.

Materials and Methods

The methodology considered for this study is based on Iterative and Incremental Development (DII) that allows constant evolution through short development cycles Solano-Fernández and Porras-Alfaro (2020), allowing constant adaptation to new requirements and continuous improvement of the product. This approach allowed the project to be divided into short iterations, each focused on developing a specific set of functionalities. In each iteration, the following activities were performed: Planning, design, development, testing and evaluation. At the end of each iteration, an increment of the product was obtained, which was evaluated to obtain feedback and make necessary improvements.

The study is divided into small iterations. In each cycle, a functional version with new features or improvements is delivered. Between each iteration, the results are analyzed, comments are collected as a development guide for the next iteration. Different parts of the software are integrated and tested continuously Yacelga *et al*. (2021). Figure (1) indicates the phases of this methodology applied in this study.

The Iterative and Incremental Development (IID) approach applicable to this study consists of the following phases. In a first phase, security requirements, one-time password generation methods and transport or delivery protocols are analyzed. The goal of the sec phase is to establish the infrastructure of the OTP system, including the selection of OTP generation algorithms, database structure and verification logic for OTP codes. In the third iteration, the development phase focuses on implementing the OTP software, creating the core functionality to securely generate, send and verify one-time passwords. Finally, in the fourth iteration, the OTP application is refined through security and usability testing, incorporating feedback to improve the user experience and ensure the robustness of the system.

It is worth mentioning that each phase is accompanied by deliverables according to requirements known as increments, which allows for continuous feedback and improvement as indicated in Fig. (2). The application of this methodology solves the correction of errors in case they exist, resulting in successful authentication to advance to the next increment.

Proposed Solution

Based on what was indicated in the previous section, the solution proposal includes incremental phases that help meet the objective of this study. It should be noted that before carrying out the phases, it is important to define the architecture of the system, which ranges from the user's interaction with the specific authentication system, where a token associated with their account is enabled and generated for later use on the mobile device or delivered via email depending on whether HOTP or TOTP method is used, to the interaction with the OTP code entry on the host system for successful authorization.

The general architecture of the solution is presented in the diagram in Fig. (3), where the process flow is reflected with 5 steps described below. However, this architecture can present vulnerabilities if the appropriate measures are not implemented. Therefore, in this proposal for the interaction between backend and frontend, SSL certificates and request control limits have been incorporated to optimize the user experience and facilitate the integration of the processes of generation, sending and validation of OTP codes.

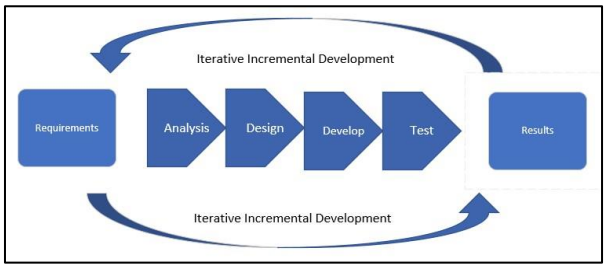

Fig. 1: Phases-incremental and iterative development methodology

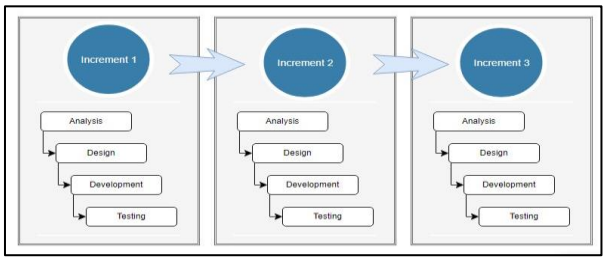

Fig. 2: Increments and phases-iterative and incremental development methodology

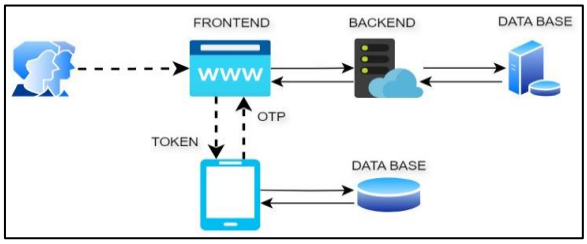

Fig. 3: General architecture-OTP solution for 2FA authentication

Login and 2FA Setup

The user logs into the host system using common credentials (username and password). Once recognized this activates two-factor authentication When 2FA is activated, the user must choose the OTP delivery method, either email or mobile app.

For mobile applications that generate temporary access codes, the host system provides a unique identifier in two formats: A QR image and a string code of letters and numbers. The mobile application facilitates reading the QR code or allows the registration of the string sequence, subsequently generating the OTP code based on the provided data. Among the mobile applications that stand out according to a comparative analysis carried out based on the last 5 years in google trends for OTP authenticator apps, google authenticator, microsoft authenticator and FreeOTP stand out. These are applications from well-known companies in the technology sector, such as Google, Microsoft and Red Hat, they are free and adhere to the RFC 4226 and RFC 6238 standards M'Raihi *et al.* (2005; 2011).

OTP Code Generation

As a dynamic code generation method, both HMAC-based One Time Password (HOTP) and Time-based One Time Password (TOTP) are used. This selection is based on the literature review whose results are summarized in Table (1) and also because they are RFC document standards M'Raihi *et al.* (2005; 2011) defined by the IETF. HOTP and TOTP are authentication protocols that provide a high level of security by generating one-time passwords using a secret key linked with the user account and a cryptographic algorithm, such as SHA-1, SHA-256 or SHA-512.

For HOTP, a counter is used that increments by one with each new validated code, the delivery is carried out by email when the system validates the user credentials provided during logging phase. In contrast, TOTP uses the current time in sec as the time value to generate the unique code which is provided by the mobile application.

In both cases, libraries that implement standards from M'Raihi *et al*. (2005; 2011) were used.

Mobile App Linking

The user installs the 2FA verification mobile application on their smartphone. The mobile application asks the user for their credentials to validate it. Once they have access to the app, they must examine the QR code generated in step 1 or they are asked to manually enter the encoded text string. The mobile app links the user account to the token and preserves the access identifier securely.

Once the mobile application is added, the app has the possibility of generating codes that have a limited duration and are usable only once. Using the application usually generates new OTP codes automatically and shows them to the user on a clear and easy-to-use screen.

Authentication with OTP Code

When the user needs to identify themselves in the host system, in addition to their username and password, they have to enter the OTP code delivered by email or generated from the mobile app.

Validation and Verification

The host system ensures the reliability of user identification by comparing the entered username and password with those stored in the database, ensuring that the OTP number provided is correct, has not expired and matches the one generated for the user and the access token. If the validation is successful, the host system gives the user the opportunity to transact on it, otherwise, the system displays an error message and prevents access.

Table 2: OTP delivery methods

Method	Description	Reference
SMS	The password is sent by SMS to the device linked to the user	Marmolejo Corona *et al*. (2023); Reese *et al*. (2019); Ali and Ismail (2018); Bruzgiene and Jurgilas (2021); Ma *et al*. (2019); Raddum *et al*. (2010); Aparicio *et al*. (2024)
Email	The access password is sent by email to the user	Reese *et al*. (2019); Ali and Ismail (2018); Bruzgiene and Jurgilas (2021); Ma *et al*. (2019)
Mobile app	The access code is generated in the user's mobile application	Reese *et al*. (2019); Ma *et al*. (2019); Lumburovska *et al*. (2021); Raddum *et al*. (2010); Papaspirou *et al*. (2023)
Hardware device	The password is generated on a physical device in the user's possession	Williamson and Curran (2021); Reese *et al*. (2019); Lumburovska *et al*. (2021); Bruzgiene and Jurgilas (2021)

DOI: 10.3844/jcssp.2025.1099.1112

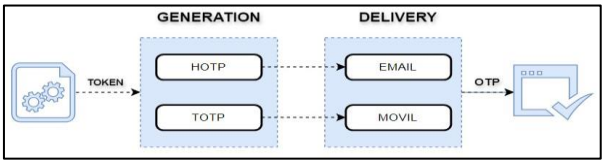

Fig. 4: OTP code generation and delivery service

Thanks to the overall system architecture, each increment addresses specific functionalities that progressively strengthen system security, the objective in first increment was to rely on HMAC to generate the OTP with an email delivery method, however, phishing attacks, email account compromise, delivery delays, interception with sniffing tools and automated attacks to decrypt sent codes, make it necessary to look for more secure and robust alternatives such as time-based OTP, so in the second and third increments, the analysis, design, development and testing is carried out with TOTP, that is, time-based OTP and a custom mobile application is also defined to generate the OTP code and interact with the host system. This modular approach enables iterative validation and controlled deployment of each component within an operational environment. Figure (4) provides a visual representation of the relationship between the OTP generation process and the delivery methods adopted in each development phase.

First Increment

Analysis

By having an authentication system using username and password, it is necessary to enable two-factor authentication, for which, the main requirement is to send a unique code to the user email account. The system must generate unique 6-digit OTP passwords based on HTOP for each user at the time of authentication, they must be sent by email, subsequently the system must request and validate the OTP codes entered by the user and finally the system must register authentication activities including date, time and access location.

Design

In the web interface, 2FA authentication enables with email sending option, then in each login entry an OTP code must be provided in addition to the username and password. In the authentication process, the service that is running on the application server obtains the request to generate the OTP code using HMAC with a one-way encryption hash, it must store the token and the counter in the database and must send the code to the email address associated with the user's account. The process to generate the unique 6-digit code with HOTP involves receiving a counter and the secret key or token. Each time the user generates a new request; the counter is incremented by 1. The token and the counter are validated on the server so that match the request, as described in Fig. (5).

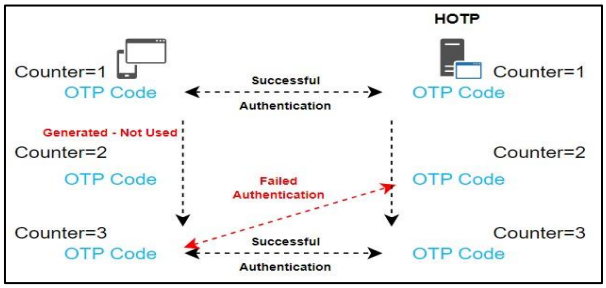

Fig. 5: HOTP authentication

Development

The development is based on the previously carried out design, uses the technology stack of a PostgreSQL Database, Backend services through JAVA EE running on a Wild-fly application server and PHP MVC running on an Apache Web Server. The service responsible for generating, delivering and validating the OTP code is developed using the Java programming language, it utilizes the jotp library which implements HMAC-based and Time-based algorithms Delamar (2020), this service communicates with PostgreSQL database to store information and allows information exchanges through a RESTful API.

Using the secret key and counter, an HMAC is generated which is then truncated to produce a 6-digit numeric code. The generated OTP code is sent to user email address, the user enters the received OTP code, the server compares the received OTP code with the internally generated values. If they match, the counter is updated and the authentication is successful. In case of discrepancies, the server may attempt to synchronize the counter up to a certain limit before returning an error message. Both the user and the authentication server must maintain the same counter value. Each authentication attempt increments the counter on both sides.

Testing

This section analyzes the performance of the OTP generation method, which is of utmost importance, since server load, network latency and the complexity of the OTP generation algorithm can affect the user experience. It is important to ensure that the HOTP-based mechanism is a fast method in the authentication process. The test uses the server-side System.currentTimeMillis() statement as a basis for recording execution times and measuring the performance of the HOTP algorithm.

The test were repeated multiples times simulating different points of authentication such as, windows, web and mobile applications and the average times were recorded to get the best estimate of the system. Tools such as Apache JMeter, Postman and BurpSuite, contributed to sending data in the authentication process and thus measure the generation time of the OTP code based on HMAC.

Table 3: Average HOTP code generation times

Interaction	OTP Generation time [ms]
1	224
2	35
3	25
4	26
5	84
6	47
7	34
8	26
9	23
10	17
Average	54.1

The loading periods were defined as the time required to generate an OTP code. To obtain a more accurate estimate of the system's performance, the average generation times were recorded as shown in Table (3). The tests were performed in a controlled concurrent environment with 100 users interacting 10 times simultaneously; the spawned threads typically have higher times in the first iteration, but when they are reused after completing the OTP code generation, their times decrease significantly. It is worth mentioning that some threads are not reused, so the average time tends to increase in subsequent interactions, as shown in repetition 5.

Second Increment Analysis

Integration with cross-platform mobile applications that implement one-time passwords becomes an effective way to interact with 2FA. The apps generate HOTP codes based on HMAC and TOTP based on time, in addition, they implement QR code reading that facilitates the link of the user account through a shared secret and they work offline, which guarantees continuity of access even in environments with limited or no connectivity. Although the delivery of codes by email was a security standard, it has become obsolete given the sophistication of current cyber threats, which is why authentication based on modern applications to guarantee the security of its users arises as a requirement in this increase.

These results demonstrate the efficiency and scalability of the HMAC algorithm in concurrent environments, making it ideal for use as a method for 2FA authentication. The process flow, as well as the result of this increase can be visualized in Fig. (6).

The requirement for this solution must include 2FA using free applications such as Google Authenticator or Microsoft Authenticator or FreeOTP since they provide a high level of trust and support as they are developed and maintained by recognized entities in the technology sector. In addition, economic barriers must be eliminated without additional costs and be accessible from multiple platforms.

Design

The system architecture allows new services to be generated with access token information or QR codes to be read in the mobile app. The server is designed to handle authentication process in both generation and validation requests using a RESTful API with PHP MVC interface. On the client side, when viewing the QR code or token through a web form, the Google Authenticator or Microsoft Authenticator or FreeOTP application is used for scans the data and stores the shared secret. Once the user account is linked, depending on whether HOTP or TOTP was chosen as the generation algorithm, from the mobile app generates temporary access codes to be entered together with the username and password in the web interface. The backend validating the credentials and code and either grants or denies access to transact in the host system. The design of the transactionality and fluidity of the process can be seen in Fig. (7).

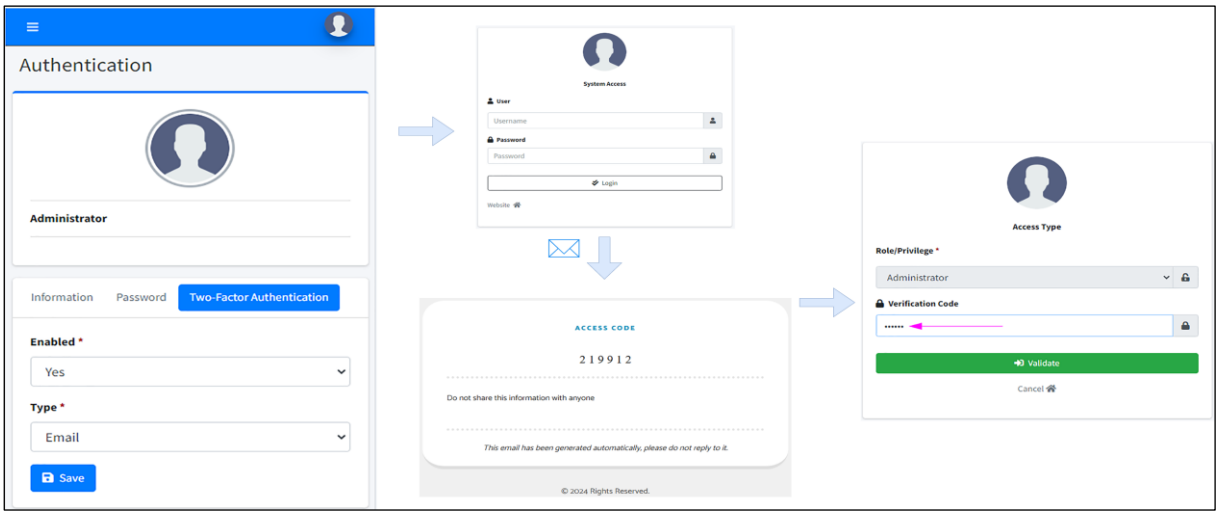

Fig. 6: Flow-results increment 1

Development

Interfaces are implemented to display tokens and QR images linked to REST services running on an application server, the google AUTH library is used to generate the shared secret to be saved in the database and using the zxing library, an image is generated in Base64 format, which contains the QR code based on HOTP or TOTP to be scanned by the Google Authenticator or Microsoft or FreeOTP Authenticator mobile application.

The generated QR uses otpauth format like *otpauth://type/label?parameters* as mention in Google (2024); Hoffman (2011) where type indicates the type of OTP, either HOTP or TOTP, the label represents a user account identifier and parameters contains the issuer and the secret key as a base. The applications are designed to read and identify the standard to be generated.

In the authentication process, after validating the user credentials, the OTP code must be provided, which according to the generation method is carried out from the mobile application; On the server, it is validated that the code corresponds and is not expired. If it is correct, access is given to transact in the system. Otherwise, there are options to cancel or re-enter the OTP code.

Testing

Unit tests were carried out for the generation of tokens and QR images, generation times were tested with multiple users at the same time. Integration tests were carried out to verify the complete 2FA Authentication Flow, here communication errors between components were detected and corrected, which guaranteed a fluid and correct interaction as in Fig. (8).

Mobile applications such as Google Authenticator and Microsoft Authenticator and FreeOTP executed on Android and iOS platform were used to generate OTP codes, which were entered in the authentication process and were successfully validated. Users installed and associated their accounts with these free mobile applications. Security tests where focused on identifying and mitigating vulnerabilities in the system with sensitive information when generating the QR code.

Third Increment Analysis

The data flow by having a mobile application that generates OTP codes for successful authentication and linked to the user account and mentioned in the previous increments, is presented as an essential requirement of this phase. However, the functionality of the solution was heavily dependent on third-party applications, which could introduce vulnerabilities or changes in the policies of the developing companies.

Design

Thanks to the previous increments, it is possible to reuse the backend architecture and add new features for enabling 2FA and authentication via OTP codes. A fundamental part is the URI format of the QR code, whose components are explained in Table (4). From the mobile application, when reading the QR image, account data linked with the shared secret is stored in the device's local SQLite database to subsequently generate the OTP code for authentication in host system.

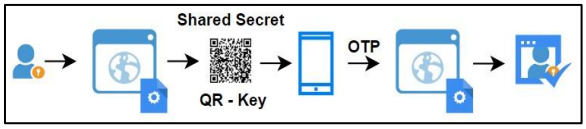

Fig. 7: OTP authenticator app

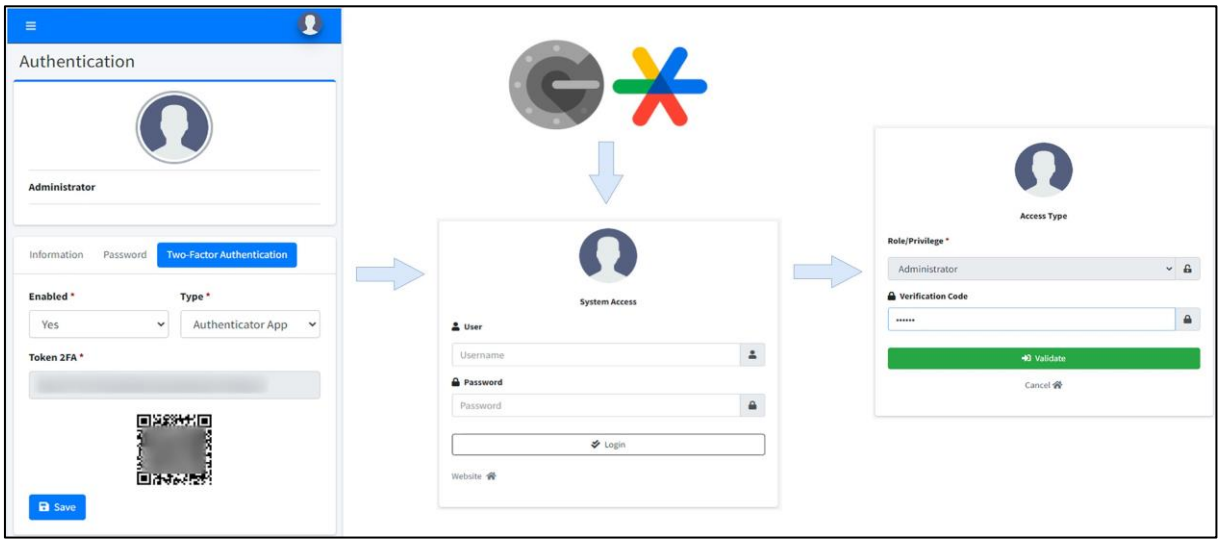

Fig. 8: Flow-results increment 2

DOI: 10.3844/jcssp.2025.1099.1112

Table 4: QR code URI components

Component	Description
Scheme	OTP URI scheme
Type	OTP type, can be HOTP or TOTP
Label	Identifier composed of the issuer and account name
Secret	Secret key in Base32 format
Issuer	Provider name or service name
Algorithm	Hash algorithm SHA1, SHA256 or SHA512. (Optional, default SHA1)
Digits	Number of digits in the OTP code, can be 6 or 8. (Optional, default is 6)
Counter	HOTP initial counter
Period	OTP code generation time. (for TOTP only, default 30 sec)

The design mockups in the mobile application are given by Fig. (9), where the user interface is clear and simple when linking accounts by scanning QR codes or registering tokens, every time the user enters the account the current OTP code is presented to be linked to the authentication system.

By having user stories where the actors involved in this process interact with the host system, customized communication flows are defined for each use case: Login, enabling 2FA, linking the mobile application, generating OTP codes and authentication with OTP code. The mobile application reads QR codes according to the otpauth format in Google (2024); Hoffman (2011), in which the type is identified for the generation of OTP either through HOTP or TOTP, stores encrypted information locally and is accessible through passcode, it was developed in Flutter with a focus on the Android system.

Development

Considering the overall process flow for successful authentication, new options were developed in the host system with 2FA enablement and integration to the mobile application. The user requests in the web interface running on a Nginx server linked to local services developed using Jakarta REST that their account has support for two-factor authentication, selects the type of code to validate based on counter or based on time and generates the QR code together with a token or secret in Base32, the considerations for the QR code are based on Table (4) while the options for registering the token are defined in the mobile application. The programming languages used up to this point at the backend and frontend level for web with client-server architecture are Java, Php and JavaScript, including the Laravel and Vue as frameworks.

The mobile application was developed in Flutter focused on the Android platform, it has a local SQLite database and maintains connection with RESTful services. It maintains a security layer to access and generate new OTP codes. The user scans the generated QR code and the application stores the associated account in its database, the display of the temporary access code is presented on a screen according to the selected account.

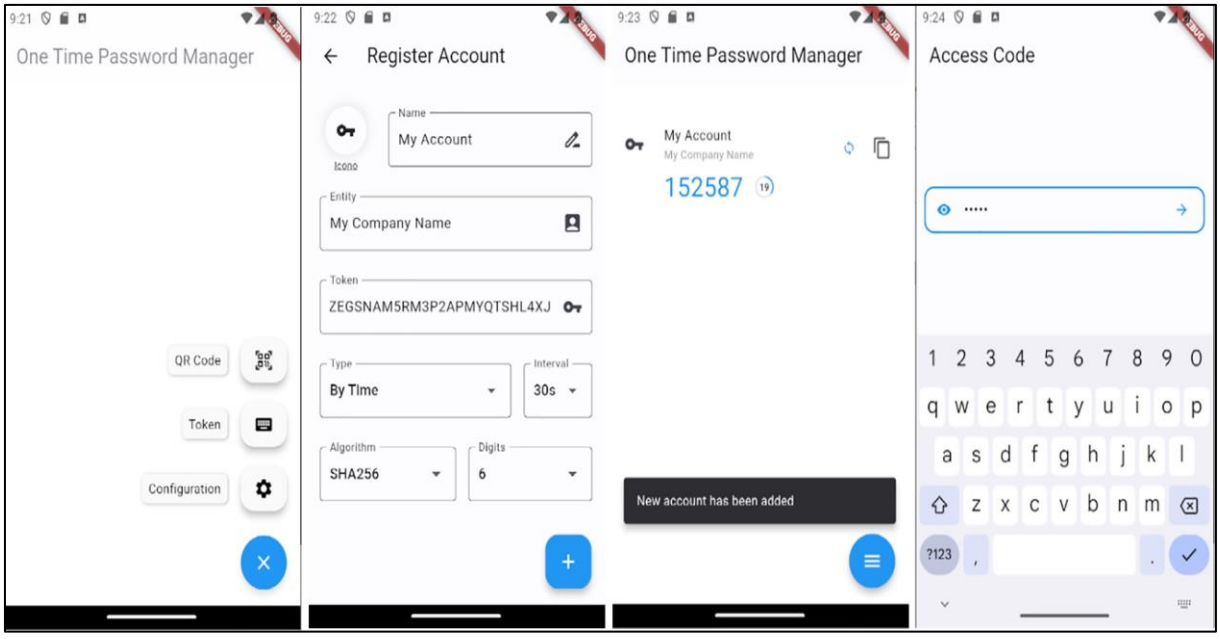

Fig. 9: Mobile app

Table 5: Tests-owasp mobile top 10 2024 Foundation (2024a-b)

Risk	Test	Resource	State
M1: Weak server-side controls	Lack of authentication and authorization controls on the server	Burp suite	Low
M2: Insecure data storage	Absence of resource protection in local device storage	Mobile security framework, SQlite database browser	Medium
M3: Insufficient transport layer protection	No network encryption and communications using HTTPS certificates	Burp suite, wireshark	Low
M4: Unintended data leakage	Incorrect handling of static data in code	Mobile security framework, SonarQube	Low
M5: Poor authorization and authentication	Scarcity of authentication and access authorization mechanisms for the application	Mobile security framework	Low
M6: Broken cryptography	Nonexistence of cryptographic algorithms	Mobile security framework	Low
M7: Client-side injection	Inadequate SQL injection protection	Flutter packages	Low
M8: Security decisions via untrusted inputs	Insufficient validation of user inputs	Mobile security framework, SonarQube	Low
M9: improper session handling	No session tokens	Mobile security framework	Medium
M10: Lack of binary protections	Deficiency in code obfuscation	Flutter	Low

Testing

For this increment, two types of base tests were carried out, one applied to the security of the mobile application and another aimed at the functionality and effectiveness of the 2FA authentication process through the participation of a group of users with existing accounts in the host system:

1. Both static and dynamic analysis were performed to ensure the protection of sensitive data and its integrity in transactionality, carrying out tests of the mobile application according to Owasp Mobile top 10 update 2024 Foundation (2024a), These tests have 10 critical points that ensure compliance with best practices and security standards. Table (5) shows the OWASP Mobile Top 10 2024 points, the approach of the tests carried out, the resources used and the results obtained, considering the OWASP Risk Rating Methodology Foundation (2024b), where the value is low if it is less than 3 (Low), medium if it is less than 6 (Medium), high if it is less than 9 (High) and critical if it is 10 (Critical)
2. Referring to the current process of adding 2FA authentication, according to the user stories in the analysis section, there are 44 users (teachers) with specific profiles in the existing system who need to interact with sensitive information and use OTP codes to access them. These users with an average age range of 42 years, with a minimum level of university education, 80% were unaware of the 2FA authentication method and of these, 90% were unaware of the name of one-time codes (OTP), however, they have used it in other access methods

Results and Discussion

Each increment was focused on overcoming the security drawbacks of the previous one, following an iterative and continuous improvement approach in a successful authentication. Throughout the different iterations, a continuous improvement in the functionality and security of the application is observed. It was detected that the DII methodology allowed greater flexibility and adaptability to the requirements of each stage.

In the first iteration, a simple enable and use functionality was implemented with generation of codes based on HMAC and delivery by email, increasing the protection of unauthorized access to the application. This method is easy to configure for users who want to increase security in a simple and fast way to their authentication. While this solution allows us to introduce a basic level of additional security to traditional credentials, there are still some glaring vulnerabilities. Delays in email delivery due to external factors, such as mail server congestion or network issues, impact the user experience. Additionally, the risk of email interception and the possibility of messages being filtered as spam reduce the reliability of the system. These issues highlighted the need for more robust, real-time solutions and prompted the search for alternatives in the second increment.

The first increase reflects user usability because it is simple and easy to configure and use, compared to generic email-based 2FA implementations, this increment incorporates customized token lifetimes and centralized server-side validation, reducing exposure to token reuse and phishing vectors, the sec increase reflects that users prefer multiplatform and zero-cost apps, they have no delays in code delivery, they work offline and are more secure when generating the code since it changes from

time to time which makes it less prone to brute force attacks, this approach decouples token generation from any vendor-specific cloud infrastructure, improving both privacy and interoperability additional, the server module is extended to validate HOTP or TOTP tokens against synchronized time windows, enhancing security boundaries without modifying the client's authentication workflow. However, the features offered by these apps do not satisfy the needs of users to manipulate data in transactions subsequent to authentication. Table (6) presented, highlights the advantages of a custom solution over free 2FA Apps. While free tools such as Google Authenticator, Microsoft Authenticator and FreeOTP offer a fast and affordable solution, they lack the flexibility, security and customization that larger organizations or those with specific security requirements need. This increment provides full control over token generation behavior, logging, and error handling routines, allowing fine-tuned integration with internal identity management systems. A custom application not only provides greater control over security and user experience, but also allows for integration and scalability that is much more aligned with business demands.

In response to the observed limitations, the sec increment, images in QR format are generated containing a shared secret which are read by free multi-platform mobile applications, by generating authentication codes directly on the user's device, these apps reduced the delay issues and security vulnerabilities associated with sending codes via email. However, new challenges arose, such as reliance on third-party apps, which, while reliable, do not provide direct control over security or user experience. In addition, managing compromised accounts and devices is difficult, as the apps did not provide an easy way to revoke access.

Finally, in the third iteration, a proprietary, adaptable and customizable mobile application was implemented that has access to system accounts and allows adding external accounts based on the Google (2024); Hoffman (2011) standards, the data control and deletion from the host system improves with respect to other mobile apps of the second increment.

Adding 2FA authentication with OTP codes provides additional security mechanisms, based on the analysis of Tables (1-2), the system supports both counter and time generation, has delivery methods by email or through multiplatform mobile applications, both online and offline and a free custom-made app.

Two-Factor Authentication (2FA) processes are implemented in several stages to enhance system security and significantly reduce the risk of unauthorized access. The modularity of the approach ensures that each phase can be independently evaluated and optimized, enabling continuous improvement and refinement of the authentication process. Initially, the user enters his or her credentials, which are verified by the host system. Once these credentials are verified, an OTP code is generated using two possible methods: HOTP using an incremental counter or TOTP based on time. HOTP sends the code to the user's email, while TOTP generates it on the mobile app by a shared secret. The user receives this code and enters it into the system to continue with the authentication. The system verifies the OTP code; if it is correct and up to date, access is granted. This process ensures that only authenticated users have access by combining the information that user knows (credentials) and the information they have (OTP). Implementing 2FA via HOTP, TOTP along with a trusted verification architecture not only improves transaction security and data integrity, but also achieves continuous and adaptive system improvement through iterative development in a successful authentication. Table (7) shows technical values for HOTP and TOTP that were successfully used in the OTP solution process flow for 2FA authentication.

Table 6: Comparison matrix: Free 2FA mobile apps vs custom app

3	Free 2FA apps (Google Authenticator, Microsoft Authenticator, FreeOTP)	Custom 2FA app
Flexibility and control	With no control over development or customization. They only allow the generation of OTP	Complete control over design, functionality, and security. Customizable to fit specific needs
User experience	Standard interface for all users, no customization option. Functionality limited to OTP generation	Fully customizable interface. Enhance the experience with notifications, deep integration
Support and maintenance	They depend on updates from third-party developers. No control over the release schedule or features	Control over technical support, maintenance and updates. Rapid response to problems and vulnerabilities
Scalability and adaptability	Scalable, but not tailored to specific requirements. Fixed functionalities	Highly scalable and adaptable to new requirements. Support for new authentication factors as needed
Implementation and maintenance	Free or low-cost, without customization or control options	Custom development, it results in a more robust and adaptable solution

Table 7: HTO and TOTP analysis

Criteria	HOTP	TOTP
RFC	RFC 4226	RFC 6238
Generation	HMAC with a counter value as initializer	HMAC with time as initialization parameter
Connection	Purely online process	No connection required
Synchrony	Same counter between server and client	Same time between server and client
Validity	Codes do not expire until used	Codes are generated at certain intervals of time
Use	Easy to use, event-based	More secure with time generated codes

Table 8: Results-OTP generation and delivery method

Generation	Delivery	Users	%
HOTP	Email	21	48
HOTP, TOTP	Free app	16	36
HOTP, TOTP	Own app	7	16
	Total	44	100

Functional testing applied at each increment along with security and usability testing in which users initiate the authentication and 2FA enablement process provide a successful flow in the generation, delivery and validation of OTP codes. The security and reliability of using a time-based temporary code generator is improved compared to the event-based method, although the HOTP-generated email delivery method improves user acceptability as it is simple and does not require interaction with third parties, communication is entirely dependent on external servers, which makes access difficult and slow.

Additionally, acceptance tests were carried out to identify the most used and enabled method according to the complete development of the system. The results are shown in Table (8), where it is concluded that 48% of users prefer the OTP code delivery method via email due to its simplicity in configuring authentication options for 2FA. Meanwhile, 36% use valid applications implemented in official stores from recognized providers to save and generate the time-based access code. This is because they prefer free apps over paid ones, and these apps are available on multiple platforms. Only 16% use their own app. The variation in percentages is influenced by the trade-off between security and usability, where multi-factor authentication, complex passwords or access restrictions make the system more secure, but often at the cost of a less fluid or more complicated user experience.

The results highlight the architectural, algorithmic, and operational distinctions inherent to each OTP method, underscoring how their selection can be aligned with specific system requirements, delivery constraints, user interaction models, and authentication workflows. This analysis informs the design and implementation of adaptive and context-aware authentication mechanisms by exposing implementation-level considerations relevant to developers, security engineers, and system architects operating in diverse deployment environments.

Conclusion

Based on the literature review, the study of the two Factor Authentication process 2FA that was put into practice provides several important findings about the security and efficiency of this authentication method. By requiring a second authentication factor such as an OTP code sent via email or generated by a mobile app, the risk of unauthorized access is significantly reduced even if the user's credentials have been compromised.

On the client side, phishing attacks remain a common threat, where attackers trick users into entering their OTP codes on fraudulent sites. On the server, incorrect time synchronization between the client and the server can allow replay attacks, where old code can be reused if not handled properly.

Common methods such as HMAC-OTP (HOTP) and Time-based OTP (TOTP) are used to generate temporary passcodes. HMAC-OTP (HOTP) uses a counter that increments with each new authentication code. This method is implemented using SHA -256 and sent via email, ensuring that each code is unique and can only be used once. However, its reliance on counter and email delivery can lead to delays and potential timing issues. Time-based OTP (TOTP) generates the OTP code with the current time as the value. Implementing TOTP through mobile applications provides greater flexibility and speed in code delivery.

Using a QR code represented by a common URI format, the apps simplify the process of setting up and using 2FA. In addition, the clear and user-friendly interface of the mobile app allows you to use OTP codes easily and efficiently.

The existing system architecture ensures reliable verification of OTP credentials and codes. By verifying that the OTP has not expired and matches the OTP generated for a specific user, the system ensures user authenticity and prevents unauthorized access. This

enables secure transactions and reduces the risk of fraud and identity theft.

Iterative development and continuous improvement for incremental approaches in systems development enable continuous and adaptive improvement. The first increment focused on implementing HOTP via email, while subsequent increments focused on TOTP with mobile applications. While the implementation helps improve key security aspects, continuous education and training of users are also essential. Guidance and support must be provided to ensure proper configuration and use of 2FA.

Ongoing user education and training is critical to ensuring that 2FA is configured and used correctly, thereby maximizing its effectiveness and minimizing issues that can arise due to misuse or lack of knowledge.

Acknowledgment

We sincerely thank the Centro de Posgrados, Pontificia Universidad Católica del Ecuador Sede Ambato, Ambato, Ecuador, for the academic opportunity in a higher master degree level as well as the services provided.

We appreciate the efforts of the editorial team in reviewing and editing our study and we are grateful for providing the resources and platform necessary to share our findings with a wider audience through this publication. We believe that our One Time Password (OTP) solution offers an important advance in the security of electronic transactions and the protection of online information.

Funding Information

This study was conducted without external financial support. No grants, institutional funding, or third-party financial contributions were received for the conception, execution, or publication of this manuscript.

Author's Contributions

Oscar Efrén Acosta Mayorga: Responsible for the research and development of this article, including designing, implementing and testing the software used in the study. Conducted a comprehensive review of the relevant literature. Wrote the full manuscript, responsible for the integration of findings and preparation of the study for submission and review.

Sang Guun Yoo: Supervisor of the entire research project, expert strategic guidance throughout the process as well as ensuring the technical quality of this study and giving the final verdict on the suitability and accuracy of the study performed.

The collaboration between both authors ensured the quality and scientific rigor of the article.

Ethics

The authors confirms that the manuscript is original, created by the authors, contains unpublished material, has not been submitted elsewhere, accurately represents their own research, is well-contextualized within existing literature and there are no ethical issues involved.

References

Ali, A. N. M., & Ismail, M. M. (2018). Acquiring the Clouds Using Otp. *IJRCS - International Journal of Research in Computer Science*, *1*(4), 15–17.

Ali, F. A. B. H., Hanza, M. Z. B. M., & Mohd Sukri, M. A. B. (2020). Two Factor Authentication by Using SMS for Web Based Application. *International Journal of Information Technology Infrastructure*, *9*(6), 21–24. https://doi.org/10.30534/ijiti/2020/02962020

Aparicio, A., Martínez-González, M. M., & Cardeñoso-Payo, V. (2024). App-Based Detection of Vulnerable Implementations of OTP SMS APIs in the Banking Sector. *Wireless Networks*, *30*(7), 6451–6464. https://doi.org/10.1007/s11276-023-03455-w

Binbeshr, F., Por, L. Y., Kiah, M. L. M., Zaidan, A. A., & Imam, M. (2023). Secure PIN-Entry Method Using One-Time PIN (OTP). *IEEE Access*, *11*, 18121–18133. https://doi.org/10.1109/access.2023.3243114

Bruzgiene, R., & Jurgilas, K. (2021). Securing Remote Access to Information Systems of Critical Infrastructure Using Two-Factor Authentication. *Electronics*, *10*(15), 1819–1869. https://doi.org/10.3390/electronics10151819

Delamar, A. (2020). *Jotp -2fa Otp Utility in Java*. https://amdelamar.com/jotp/

Foundation, O. (2024a). *Owasp Mobile Top 10 — Owasp Foundation*. https://owasp.org/www-project-mobile-top10/

Foundation, O. (2024b). *Owasp Risk Rating Methodology*. https://owasp.org/wwwcommunity/OWASP Risk Rating Methodology

Google. (2024). *Key uri Format. GitHub Repository*. https://github.com/google/googleauthenticator/wiki/Key-Uri-Format

Hoffman, P. E. (2011). *Requirements for Internet-Draft Tracking by the IETF Community in the Datatracker*. https://doi.org/10.17487/rfc6293

Kamau, J., & Mgala, M. (2022). A Review of Two Factor Authentication Security Challenges in the Cyberspace. *International Journal of Advanced Computer Technology*, *11*(5), 1–6.

Kirvan, P., Loshin, P., & Cobb, M. (2023). What is Two-Factor Authentication (2fa)? *Identity and Access Management*. https://authy.com/what-is-2fa/

Lumburovska, L., Dobreva, J., Andonov, S., Trpcheska, H. M., & Dimitrova, V. (2021). A Comparative Analysis of Hotp and Totp Authentication Algorithms. which One to Choose? *Security and Future*, *5*(4), 131–136.

M'Raihi, D., Bellare, M., Hoornaert, F., Naccache, D., & Ranen, O. (2005). *HOTP: An HMAC-Based One-Time Password Algorithm*. https://doi.org/10.17487/rfc4226

M'Raihi, D., Machani, S., Pei, M., & Rydell, J. (2011). *TOTP: Time-Based One-Time Password Algorithm*. https://doi.org/10.17487/rfc6238

Ma, S., Feng, R., Li, J., Liu, Y., Nepal, S., Diethelm, Bertino, E., Deng, R. H., Ma, Z., & Jha, S. (2019). An Empirical Study of SMS One-Time Password Authentication in Android Apps. *Proceedings of the 35th Annual Computer Security Applications Conference*, 339–354. https://doi.org/10.1145/3359789.3359828

Mahdad, A. T., & Saxena, N. (2023). SoK: A Comprehensive Evaluation of 2FA-Based Schemes in the Face of Active Concurrent Attacks from User Terminal. *Proceedings of the 16th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, 175–186. https://doi.org/10.1145/3558482.3590183

Marmolejo Corona, I. V., Serrano Manzano, G. A., Bautista Aguilar, F. A., & Santiago Gonzalez, Y. F. (2023). Seguridad en Sistemas de Autenticación: Análisis de Vulnerabilidades y Estrategias de Mitigación. *XIKUA Boletín Científico de La Escuela Superior de Tlahuelilpan*, *11*(22), 39–43. https://doi.org/10.29057/xikua.v11i22.10802

Matelski, S. (2022). Humancomputable Otp Generator as an Alternative of the Two-Factor Authentication. *EICC '22: Proceedings of the 2022 European Interdisciplinary Cybersecurity Conference*, 71–64. https://doi.org/10.1145/3528580.353284

Ozkan, C., & Bicakci, K. (2020). Security Analysis of Mobile Authenticator Applications. *2020 International Conference on Information Security and Cryptology (ISCTURKEY)*, 18–30. https://doi.org/10.1109/iscturkey51113.2020.9308020

Papaspirou, V., Papathanasaki, M., Maglaras, L., Kantzavelou, I., Douligeris, C., Ferrag, M. A., & Janicke, H. (2023). A Novel Authentication Method that Combines Honeytokens and Google Authenticator. *Information*, *14*(7), 386–423. https://doi.org/10.3390/info14070386

Raddum, H., Nestås, L. H., & Hole, K. J. (2010). Security Analysis of Mobile Phones Used as OTP Generators. *Information Security Theory and Practices. Security and Privacy of Pervasive Systems and Smart Devices*, *6033*, 331–324. https://doi.org/10.1007/978-3-642-12368-9_26

Reese, K., Smith, T., Dutson, J., Armknecht, J., Cameron, J., & Seamons, K. (2019). A Usability Study of Five Twofactor Authentication Methods. *15th Symposium on Usable Privacy and Security (SOUPS 2019*, 357–370.

Shukla, V., Chaturvedi, A., & Srivastava, N. (2019). A New One Time Password Mechanism for Client-Server Applications. *Journal of Discrete Mathematical Sciences and Cryptography*, *22*(8), 1393–1406. https://doi.org/10.1080/09720529.2019.1692447

Solano-Fernández, E., & Porras-Alfaro, D. (2020). El Modelo Iterativo e Incremental Para el Desarrollo de la Aplicación de Realidad Aumentada Amón_RA. *Revista Tecnología En Marcha*, *33*(8), 165–177. https://doi.org/10.18845/tm.v33i8.5518

Tirfe, D., & Anand, V. K. (2022). A Survey on Trends of Two-Factor Authentication. *Contemporary Issues in Communication, Cloud and Big Data Analytics*, *281*, 285–296. https://doi.org/10.1007/978-981-16-4244-9_23

Uysal, E., & Akgün, M. (2023). P/Key: PUF Based Second Factor Authentication. *PLOS ONE*, *18*(2), 0280181. https://doi.org/10.1371/journal.pone.0280181

Verma, S., Singh, M., Chaturvedi, K., & Tripathy, B. K. (2023). An Efficient Multifactor Authentication System. *Computational Intelligence in Pattern Recognition*, *725*, 109–122. https://doi.org/10.1007/978-981-99-3734-9_10

Williamson, J., & Curran, K. (2021). The Role of Multi-Factor Authentication for Modern Day Security. *Semiconductor Science and Information Devices*, *3*(1), 16–23. https://doi.org/10.30564/ssid.v3i1.3152

Yacelga, A. R. L., Espinoza, J. L. A., & Vásquez, R. A. D. (2021). Aplicacion De La Metodolog´´ıa Incremental ´En el Desarrollo De Sistemas de Informacion.´. *http://scielo.sld.cu/scielo.php?pid=s2218-36202021000500175&script=sci_arttext*, *13*(5), 175–182.

Yin, X., He, J., Guo, Y., Han, D., Li, K.-C., & Castiglione, A. (2020). An Efficient Two-Factor Authentication Scheme Based on the Merkle Tree. *Sensors*, *20*(20), 5735–5967. https://doi.org/10.3390/s20205735