information

Article

A Novel Authentication Method That Combines Honeytokens and Google Authenticator

Vassilis Papaspirou [1,†], Maria Papathanasaki [2,†], Leandros Maglaras [3,*,†], Ioanna Kantzavelou [1,†], Christos Douligeris [4,†], Mohamed Amine Ferrag [5,†] and Helge Janicke [6,†]

[1] Department of Informatics and Computer Engineering, University of West Attica, 12244 Athens, Greece; vpapaspyrou@uniwa.gr (V.P.); ikantz@uniwa.gr (I.K.)
[2] Department of CS and Telecommunications, University of Thessaly, 35100 Lamia, Greece; mpapathanasaki@uth.gr
[3] School of Computing, Edinburgh Napier University, Edinburgh EH10 5DT, UK
[4] Department of Informatics, University of Piraeus, 18534 Piraeus, Greece; cdoulig@unipi.gr
[5] Technology Innovation Institute, Masdar City 9639, Abu Dhabi, United Arab Emirates; mohamed.ferrag@tii.ae
[6] Cyber Security Cooperative Research Centre (CSCRC), Edith Cowan University, Perth, WA 6027, Australia; h.janicke@ecu.edu.au
* Correspondence: l.maglaras@napier.ac.uk
† These authors contributed equally to this work.

Abstract: Despite the rapid development of technology, computer systems still rely heavily on passwords for security, which can be problematic. Although multi-factor authentication has been introduced, it is not completely effective against more advanced attacks. To address this, this study proposes a new two-factor authentication method that uses honeytokens. Honeytokens and Google Authenticator are combined to create a stronger authentication process. The proposed approach aims to provide additional layers of security and protection to computer systems, increasing their overall security beyond what is currently provided by single-password or standard two-factor authentication methods. The key difference is that the proposed system resembles a two-factor authentication but, in reality, works like a multi-factor authentication system. Multi-factor authentication (MFA) is a security technique that verifies a user's identity by requiring multiple credentials from distinct categories. These typically include knowledge factors (something the user knows, such as a password or PIN), possession factors (something the user has, such as a mobile phone or security token), and inherence factors (something the user is, such as a biometric characteristic like a fingerprint). This multi-tiered approach significantly enhances protection against potential attacks. We examined and evaluated our system's robustness against various types of attacks. From the user's side, the system is as friendly as a two-factor authentication method with an authenticator and is more secure.

Keywords: honeytoken; authentication; security; encryption; threat modeling; two-factor authentication

check for updates

Citation: Papaspirou, V.; Papathanasaki, M.; Maglaras, L.; Kantzavelou, I.; Douligeris, C.; Ferrag, M.A.; Janicke, H. A Novel Authentication Method That Combines Honeytokens and Google Authenticator. *Information* **2023**, *14*, 386. https://doi.org/10.3390/info14070386

Academic Editor: Arkaitz Zubiaga

Received: 24 April 2023
Revised: 3 July 2023
Accepted: 4 July 2023
Published: 7 July 2023

1. Introduction

The rapid spread of communication systems has made it clear that relying on a single password to protect online accounts is no longer sufficient [1]. Advanced methods for identifying users that take advantage of the unique features of each system have been developed.

There are several reasons why a single password is not sufficient to guarantee the security of an online account. These include users' inability to create strong passwords, the improper storage of passwords by service providers, and the risk of falling victim to phishing attacks. Masquerading, or impersonating a user, is a serious security threat to any computer device. To protect against this danger, user authentication is an essential part of any security infrastructure, serving as the first line of defense.

In our effort to enhance security measures, we incorporated a concept known as "honeytokens"—decoy passwords inserted into a password file aimed at detecting unauthorized access. If an attacker tries to use one of these honeytokens, an alert is triggered, signaling a potential system compromise.

This research work introduces a cutting-edge authentication method that combines the power of honeytokens and the convenience of Google Authenticator. It is a novel approach that offers a unique blend of security and usability, making it a perfect fit for any organization looking to protect its sensitive information. By using honeytokens, a set of false passwords are mixed in with the user's real password. This method makes it impossible for intruder's to determine the real password, even if they manage to obtain the password file. The inclusion of Google Authenticator, a widely used two-factor authentication technology, provides an extra layer of security, making it virtually impossible for unauthorized access. This integrated method is easy to use, even for people who are not tech-savvy, and offers maximum protection against major threats such as SIM swapping, stalkerware, and side-channel attacks.

Cybersecurity requires the use of multiple levels of protection to ensure that there is always a backup method if the first line of defense fails [2]. Multi-factor authentication (MFA) is a security technique that verifies a user's identity by requiring multiple credentials from distinct categories. These typically include knowledge factors (something the user knows, such as a password or PIN), possession factors (something the user has, such as a mobile phone or security token), and inherence factors (something the user is, such as a biometric characteristic like a fingerprint). This multi-tiered approach significantly enhances protection against potential attacks.

We propose a system that adds extra layers to the defense system, building on the existing two-factor authentication methods. Our approach introduces honeytokens in a two-factor authentication scheme and combines them with Google Authenticator. This novel system departs from traditional solutions and offers a new approach to two-factor authentication.

The proposed research offers several key contributions:

- The honeytokens approach, when paired with a not-so-common computing device as a second factor, can significantly boost security.
- The identification process is fast and efficient, utilizing the popular QR-code mechanism.
- The fusion and coordination of currently used, mainstream authentication technology.
- The "Two-Factor Honeytoken Authentication Mechanism" is a unique and integrated solution, which can be used with any authenticator.
- The method can be used with any authenticator, within the system described in this work utilizing Google Authenticator.
- The proposed system boasts low complexity, high flexibility, and a user-friendly design [3], making it accessible even for older individuals. It has mutual authentication, resistance to known attacks, and no smart-card-loss attack.
- Security evaluations have also revealed that it is resistant to various significant threats, including SIM swapping, stalkerware, and side-channel attacks, as well as others.

The paper is organized as follows: Section 2 presents the related research work on user authentication and two-factor authentication. Section 3 provides details on authentication from different perspectives, including a description of multi-factor authentication and different types of two-factor authentication. It also compares popular authenticators and discusses common issues with these methods. Section 4 shows a comparison of some authenticators' applications that already exist. We compare them regarding the features they have and their mechanisms. It has also a prior mechanism idea, which we used for this system (honeytokens), about other two-factor authentication systems and one-time passwords(OTPs). Section 5 describes the proposed system in detail. Section 6 presents a security analysis of the proposed system also with regard to other systems. Some attacks illustrate the operation of the proposed mechanism. Threat modeling is displayed as the main factor in every system's security, and measures that must be taken to be safer are

discussed. In Section 7, two-factor authentication is described as a service. It answers the questions "how it works for companies and what are the advantages and the disadvantages and how companies can benefit from that". In Section 8, future improvements and suggestions for better-layered security of our system are exposed. In Section 9, evaluation and criticism of the proposed method results in valuable conclusions. We calculate and describe the time that our system requires to accomplish some functions. We compare some existing password-hashing algorithms and how time depends on them. Finally, Section 10 suggests potential future work actions to expand on the findings of the research carried out and summarizes its outcomes.

2. Related Work

In the past years, a suggestion was made with three methods to significantly eliminate the probability of offline password guessing: (a) make use of a machine-specific function, (b) make use of distributed cryptography, and (c) utilize services like external password hardening (like Phoenix [4]). Nevertheless, each of these methods needs a very big amount of changes to the authentication protocols that they are part of on the server side. Method (a) is not scalable enough to be used for Internet-based services, making it unable to support the backup of password files with hashed values stored in a distributed manner. Method (b) requires changes to client-side systems, which is generally seen as undesirable, and method (c) has a single point of failure and discloses information about user behavior to outside parties. People often choose weak passwords or commonly used terms that can be easily located using a dictionary attack when the authentication system is not guiding their choices and is not monitoring their actions.

In 2013, A. Juels and R. Rivest [5] first suggested the use of honeywords to enhance hashed password protection against cracking. These honeywords serve as decoys that make it difficult for an attacker to determine which passwords are real and which are fake. Even if an attacker manages to access a file containing hashed passwords and tries to locate any password by trying to match different hashes, they will not be able to distinguish between genuine passwords and honeywords. To detect any attempts at password cracking, a "honeychecker" is triggered if an attacker tries to use a honeyword to access the system. This simple method is reliable in detecting brute force or dictionary attacks, and the idea behind it has since been fundamental for several research works that have expanded upon or strengthened it.

In [3], a new approach was proposed that aims to address issues generated by compromised servers and illegal users. Honeywords were used to improve the security–usability balance in the system. The authors performed user and sensor authentication to prevent node compromise.

The authors in [6] built upon the work of A. Juels and R. Rivest by addressing some gaps in their approach. They proposed a solution to prevent an attacker from altering the honeychecker code or the code running on the login server, which would trigger the alarm.

Furthermore, A. Juels and R. Rivest's work on honeywords was extended in [7]. The authors used honeywords not only to protect passwords but also to detect potential system intrusions. It is important not only to identify attackers but also to gather information about them and determine the root cause of the problem.

To improve upon traditional two-factor authentication methods and address the issue of insecure passwords and user behaviors that put computer systems at risk, the work in [8] presented a new, enhanced authentication system using honeywords. By detecting and preventing node capture attacks, this system aims to provide the highest level of secure and uninterrupted service while protecting against unauthorized access to resources and safeguarding personal information. The use of QR codes makes it easy to integrate this method into any platform or web application and allows for mobile access.

To address issues related to password protection, user behaviors that pose threats to sensitive systems, and 2FA mechanisms, the work in [9] aims at providing a high level of security and uninterrupted services while safeguarding systems and users from unau-

thorized access to resources and preventing privacy violations and personal information disclosure. By utilizing honeyword principles and incorporating QR codes, which make the implementation platform-independent, this method allows for mobile user access, and the selected strategy seeks to detect and prevent attacks.

In 2022, we further extended the research and performed specific tests against some common attacks to find out how strong the defense of our system is and what vulnerabilities still exist. We also conducted a comparison between the most well-known authenticators to reveal how their features affect the needs of a user. Each authentication mechanism has its own authenticator, and we examined their characteristics and operational features to find out how they work and justify the selection of one [10].

In this paper, we propose the use of honeytokens in an innovative two-factor authentication system. Previous attempts to fully eliminate the potential for offline password guessing, such as machine-dependent functions, distributed cryptography, and external password-hardening services have limitations. Among these are the need for significant changes to server-side authentication protocols, a lack of scalability, and single points of failure. Our approach with honeytokens aims at enhancing security while overcoming these limitations.

"Honeytokens" are broader in scope, as compared to honeywords. They can take various forms, such as decoy database records, files, or network endpoints, that are intended to attract and detect unauthorized access and use and, subsequently, notify system administrators of a breach.

3. Authentication and Authenticators

Authentication is the process of verifying the identity of a user, device, or system, typically through the use of a unique identifier such as a password or token. Authenticators are the tools or methods used to perform this verification, which are similar to a password, a fingerprint scanner, or a security token. The goal of authentication is to ensure that only authorized users can access, in a legitimate way, a system or resources.

There are various types of authenticators, each with its own set of advantages and disadvantages. Passwords are the most common form of an authentication identifier, which are widely used, but they can be easily guessed or stolen. Biometric authenticators, such as fingerprint scanners or facial recognition, provide a higher level of security, but they can be expensive and may not be suitable for all environments. Token-based authenticators, such as security tokens or smart cards, provide an extra layer of security, but they can be lost or stolen.

Authentication is an essential aspect of information security, and it is important to use the most appropriate authenticator for the environment, taking into account the cost, ease of use, and level of security required. Due to the aforementioned problems and drawbacks, the two-factor authentication method was introduced to advance security levels.

3.1. Multi-Factor Authentication

Multi-factor authentication (MFA) is a method of authentication that requires the use of two or more pieces of information for identification and provides an additional level of security. This method requires the use of a second electronic device, such as a mobile phone, tablet, or computer, for identification purposes.

Authentication is typically for applications and services that require users to register before accessing them. When a user requests access to a service, they must enter the password they have chosen during the registration process in order to confirm their identity. The password is considered as one factor from the group of *something the user knows* and serves as the first level of authentication in a one-factor authentication scheme. This helps to ensure that *only the user* who claims to access the service is able to do so.

In today's world, it is necessary to have additional levels of security in place due to the increasing sophistication and severity of attacks. This is especially important for accounts in critical infrastructure, such as banks, or platforms containing personal user data, as it is

essential to have advanced controls in place to verify the identity of anyone attempting to access these accounts.

A multi-factor authentication (MFA) approach is used by many applications where security is of major concern. Some of them are highlighted below [11,12]:

- In massive open online courses (MOOCs), it can be difficult to distinguish between the registered user and the actual user who is taking an exam or completing homework. As more universities adopt MOOC technologies, it is important to be able to verify student identities in a secure system. MFA is an effective solution for this purpose because it combines a variety of authentication factors and is both consistent and scalable. As universities move towards the concept of "metaversities" in the future, the use of novel and secure authentication mechanisms will become increasingly important. Multi-factor authentication can ensure that the registered user participating in an exam is the actual user, not a masquerader, by requiring the user to provide multiple forms of identification before gaining access to the exam system.
  In a threat model where exam fraud is a concern, MFA can be an effective security countermeasure against various types of attacks. For example, if an attacker gains access to a user's login credentials through a phishing attack or by guessing a weak password, MFA can prevent the attacker from accessing the exam by requiring an additional form of authentication, such as a fingerprint scan, facial recognition, or a one-time code sent to a registered mobile phone.
  Additionally, MFA can prevent multiple users from sharing a single login to take an exam. For example, if a student shares their login information with a friend, MFA can prevent the friend from accessing the exam without the additional authentication factors that only the registered user owns. Overall, MFA is an important security countermeasure that could ensure that the registered user is the actual user participating in an exam and could prevent exam fraud in various threat models.
- Online payments or money transactions must be fully protected like other bank applications with the best security methods. MFA can be used to quickly confirm the identity of genuine users. For larger amounts of money, stricter authentication measures may be required to identify users effectively. However, different amounts of money might adjust the selection and use of authentication factors.
- It is simple to incorporate multi-factor authentication (MFA) into the secure entry of all classes of electronic medical records, which contain highly confidential and sensitive data that must be protected. MFA can use various identity verification techniques, such as detecting a user's device, media, and environment, to provide more robust authentication.
- MFA can be further developed to accommodate various Internet computing capabilities, including the user level (such as administrators or visitors), the document level (such as a PDF containing an application form or a document with proprietary information), and the application level (such as financial applications, emails, or social media applications).

When implementing MFA, a number of aspects should be taken into account:

- To improve multi-factor authentication, devices with fingerprint recognition and a high-definition camera for retinal eye capture and analysis are needed. Additionally, software with a database that stores and manages fingerprints may be necessary.
- In order to successfully verify authentication factors, users should have a basic level of technical knowledge, particularly when using soft tokens that generate one-time passwords that expire after a certain period of time.
- Implementing improved multi-factor systems requires additional funding for development and maintenance. Most hardware-token-based systems are proprietary, and some vendors charge an annual fee per user. Replacing lost or damaged hardware tokens can also be costly. In addition to installation costs, multi-factor authentication often comes with significant additional service fees.

- MFA supporters argue that requiring additional factors beyond just a password can help reduce the harm caused by online identity theft and scams. However, MFA is still vulnerable to certain types of attacks such as man-in-the-middle and man-in-the-browser attacks. Moreover, MFA may not be effective against more modern threats such as smart and polymorphic malware, phishing, and ATM skimming.

3.2. MFA in Advanced Internet of Things

Smart devices have become a major part of our lives, leading to the creation of the Advanced Internet of Things (A-IoT) with advances in communication and computing, while these devices can enhance people's daily lives, unauthorized access to them can be dangerous. However, people also want easy interaction with these devices, which they use regularly, so it is important to ensure the security of their data while using them.

Unauthorized access to A-IoT systems poses serious security threats and the challenge of trustworthiness becomes a critical research problem in securing A-IoT systems. An access control process involves authentication and user identification. A-IoT devices can use advanced capabilities in modern society, especially if the user being identified has portable devices that can act as providers of identification codes (e.g., one-time passwords) or even as a means to collect biometric data or determine the location of the user.

The connection to the A-IoT system through short-range radio waves allows wearables to present the security credentials of their user, but this requires support from appropriate security protocols so that the platform can trust the data collected from the user's available equipment. At the same time, users can be confident that their sensitive personal information is not disclosed [13].

Firstly, wearables can act as a form of two-factor authentication. For example, a user can set up their wearable device to require both a password and biometric authentication, such as a fingerprint or a heartbeat, to access their accounts. This makes it more difficult for unauthorized users to gain access to sensitive information.

Secondly, wearables can track a user's movements and behavior patterns, which can be used to detect suspicious activity. For instance, if a user's wearable device suddenly shows activity in a location that is not typical for them, it could indicate that their account has been compromised, and the account can be locked down or additional security measures can be taken.

Thirdly, some wearables offer encrypted communication features, such as secure messaging or phone calls. This can help protect the user's communication from interception and unauthorized access.

In terms of practicality for mass/ordinary users in society, wearables offer a convenient and accessible form of enhanced security. They are easy to use, and many people already own and use wearable devices. This means that wearables can help improve the security of the general public without requiring them to undergo extensive training or purchase expensive equipment.

4. Authenticator Comparison

The number of authenticators is growing considerably, and some are becoming more common online and are particularly noteworthy for their ease of use and effectiveness [14]. Google Authenticator (GA) is the most widely used authenticator due to its many benefits. It is free and easy to use, and supports time-based one-time password (TOTP) and HMAC-based one-time password (HOTP) algorithms. It does not require an Internet connection (HOTP). However, some of its drawbacks include the inability to create backups and the lack of certain features.

Google Authenticator is an application that implements two-step verification services using the time-based one-time password (TOTP) algorithm and HMAC-based one-time password (HOTP) algorithms for authenticating users.

The application generates a six- to eight-digit one-time password, which users must enter in addition to their regular login information. Here is a brief overview of its architecture and mechanism:

- Initialization: The user initializes Google Authenticator by scanning a QR code or manually entering a secret key provided by a service provider (like Google, Dropbox, etc.). This secret key is then stored in the application.
- Token generation: When a user attempts to log in, Google Authenticator generates a unique time-based or HMAC-based one-time password (OTP) using a stored secret key. The password's validity is strictly time-limited.
- User authentication: The user must enter this OTP along with their regular login credentials within the time limit. The service provider verifies the OTP using the same secret key and allows access only if the OTP matches.

Via this process, even if a user's primary password is compromised, an attacker would still need the current valid OTP to gain access, thus adding an extra layer of security.

Authy two-factor authenticator (Authy 2FA) is another popular authentication mechanism with many similarities to GA. It is feature-rich, free, and easy to use, and also allows users to create backups and crypto-wallets. However, its multi-device synchronization capability can pose security risks because it does not require a password to use the app, making it difficult to maintain control across all devices. Additionally, Authy uses SMS as an authentication method, although 2FA solutions usually no longer make use of it. It is crucial to note that users of other authentication programs must have a rooted phone in order to transfer all of their tokens to Authy, which can compromise the overall security of the device [15].

For those who use Microsoft Services, or they use a Microsoft account, or operate the Windows 10 utilizing system on their mobile device, Microsoft Authenticator (MA) is a useful tool. One of its major advantages is that it offers authentication without passwords, with Microsoft apps to notify the user if it uses an unfamiliar environment. However, the application's somewhat complex user interface and the lack of features that would make it more attractive have limited its popularity [15].

Table 1 provides a summary of all aforementioned features examined in Google Authenticator (GA), Authy two-factor authenticator (Authy 2FA), and Microsoft Authenticator. The symbol ✓ represents the existence of a feature in the authenticator and the symbol ✗ represents its absence.

Table 1. Comparison the most used Authenticators.

FEATURES	GA	Authy 2FA	MA
Open source	✗	✗	✗
Free	✓	✓	✓
Widely adopted	✓	✓	✗
Lack of features	✗	✗	✓
Easy to use	✓	✓	✗
Data backup	✓	✓	✓
Network connection needed	✗	✗	✗
Multiple account support	✓	✓	✓
Cryptocurrency securing	✓	✓	✓
Microsoft Services compatible	✗	✗	✓
TOTP and HOTP use	✓	✓	✓

Authentication mechanisms use cryptographic methods to protect user credentials, but many users tend to use weak passwords or phrases that can easily be guessed through a dictionary attack [6]. Due to the improvements in GPU technology, it is now easy to crack a common password, e.g., by locating it as a word in a dictionary. This means that an adversary could potentially obtain user credentials through a dictionary attack.

4.1. Honeywords

Juels and Rivest [5] suggest the use of "honeywords" as a way to improve password security. This involves storing passwords differently by giving each user their own unique password and a few fake ones along. The fake passwords are named honeywords, and the whole set of honeywords plus the real password (correct password) is named sweetwords. If a user uses a honeyword during the login process, an alert will activate that the password database has been compromised.

To check if the password file has been compromised, a group of fake passwords, known as honeywords, are blended in with the user's actual password. The encrypted hash values of these passwords (real password and honeywords) are then stored in the password file. Even if an attacker manages to gain access to the file and obtain all of the hash values, they will still not be able to determine the true password. This clever technique is like leaving a decoy trail for intruders, keeping the real password safe and secure.

There are several significant advantages by using honeywords as a password security approach for a company. To begin with, the administrator does not have to do much work. With the creation of honeywords, the administrator can relax and wait for potential breaches to occur. The honeywords will likely trigger an alert, making it easy to detect a breach. Additionally, there is minimal impact on organizational structures. The only strain on linked systems is the login attempt itself and the honeywords index being sent to the honeychecker, making it an efficient and low-impact security measure. The implementation of honeywords hardly burdens the system. The only action required from the administrator is to respond to the alarm output. The company will also benefit from the added layer of security provided by the use of honeywords between the honeychecker and the computer system. If a single system component is compromised, the damage will be contained and mitigated. In the unlikely event that both components are hacked, a new hash file can easily fix the problem by changing the honeywords used, creating a new honeytrap for cybercriminals.

Honeywords are used as a red flag after the database is stolen. In our system, we are using honeytokens in order to defend the end user against security and privacy preservation attacks. Once an adversary has successfully accessed a mobile phone, the user will not be able to identify the correct OTP in order to be able to gain access to the system that is protected by the proposed 2FHA mechanism. In case the attacker enters one of the honeytokens, depending on the security level the administrators have set, the system can notify the user, lock the account or start a campaign of awareness to all users for tentative similar attacks against them too.

Honeywords are not a replacement for robust password management practices, but as cyber attacks become more frequent, they provide a valuable commercial solution for rapid and efficient data breach detection. It is important to remember that no password security system is completely foolproof. The weakest point in any security system is often people, who are easily tricked and manipulated. Honeywords can serve as an additional layer of protection to keep sensitive information safe from cybercriminals.

4.2. Types of Two-Factor Authentication

One-Time Passwords

In 1981, Leslie Lamport introduced one-time passwords (OTPs) as a means of logging into remote computing systems. OTPs are temporary and become invalid immediately after use, making them effective in providing security, especially against distributed client/service interactions and replay attacks [16]. Many authentication devices now use OTP technology for two-factor authentication, particularly in the banking industry. These devices use small hardware to generate unique passwords each time, while OTP technology has its benefits, it also has drawbacks such as the cost of purchasing and distributing the devices as well as maintenance costs. Users may also need multiple devices depending on the service they want to log in to, while OTP technology was popular in the past decade, it is now being replaced by TOTP technology in many cases.

A one-time password (OTP) [17] is a password that is valid for only one login session or transaction, on a computer system or other digital device. OTPs are used as an additional layer of security for user authentication to ensure that a password is not reused or shared among multiple users. They are typically generated by a secure server and sent to the user's device, such as a smartphone, via text message or push notification. OTPs can also be generated by a hardware token or software token, such as a mobile app. These tokens use algorithms to generate a unique OTP at set intervals, such as every 30 s. The OTP is then entered by the user along with his/her username to complete the login process.

Time-Based One-Time Passwords

One of the technologies that use the password is time-based one-time passwords (TOTPs) ([18,19]), which is quite similar to OTP, where a unique code is generated using the current time as input. Unlike OTP, TOTP can be conducted offline using an application and is hardware-free, making it widely adopted and easy to use. The offline capability is made possible by the shared secret key and system time. TOTP utilizes symmetric key cryptography, meaning that both the user and the application infrastructure use the same key to create and verify the token.

Universal 2nd Factor

Universal 2nd Factor (U2F) [20] is an authentication method that uses a key for multiple services. It requires the user to possess a physical device, typically connected to a USB port. When the user attempts to log into a service, they must also accept an encrypted signal on the USB device after entering their credentials correctly (e.g., by pressing a button on the device). The disadvantages of this method include the inconvenience of losing the device and losing access to various services if a backup device is not available. However, it is user-friendly in that it does not require charging or maintenance. Additionally, the keys are secure as private information in the key cannot be extracted.

HMAC-Based One-Time Passwords (HOTPs)

HMAC-based one-time passwords (HOTPs) ([18,21]) are like a secret code that is created based on a counter that increases every time an event happens. It is generated using hash-based message authentication codes. In this type of OTP, the user receives a unique token that is known only to them and the server and is based on a hash algorithm. The biggest factor that makes the difference between HOTPs and TOTPs is that the latter uses time as the variable factor, while HOTPs use a counter that is incremented with each event. HOTPs are more user-friendly as they do not have a time limit for entering the password, giving you the freedom to access your account at your own pace.

5. Proposed System

In this study, a new authentication method called "Two-Factor Honeytoken Authentication (2FHA)" is proposed. It combines honeytokens and two-factor authentication and works with Google Authenticator. When a new user logs in, he/she is required to provide two inputs: a password and the correct OTP, which is sent to his/her on a separate device (e.g., a mobile phone). This fulfills the 2FA requirement, and the user must enter both the password and the correct OTP in order to legitimately access the system. The user knows which OTP in the series is the correct one to enter, based on the position it is sent to him/her on the second device.

To create a new account, a user must first follow a simple registration process and complete its steps. During this process, the user is asked to choose a number from a predetermined range (e.g., 1 to 3). This number, which is agreed upon by both the user and the system, determines where the real OTP number should be placed within a series of false OTP during the registration process. Each time a user makes an attempt to log in, a new set of phony OTPs is produced with the real (correct) OTP correctly placed. For the program, which is pilot, the range of this sequence has been numbered to 3. Figure 1 illustrates the registration phase and the login phase of the 2FHA method. The following sections offer in-depth explanations of the registration and login procedures.

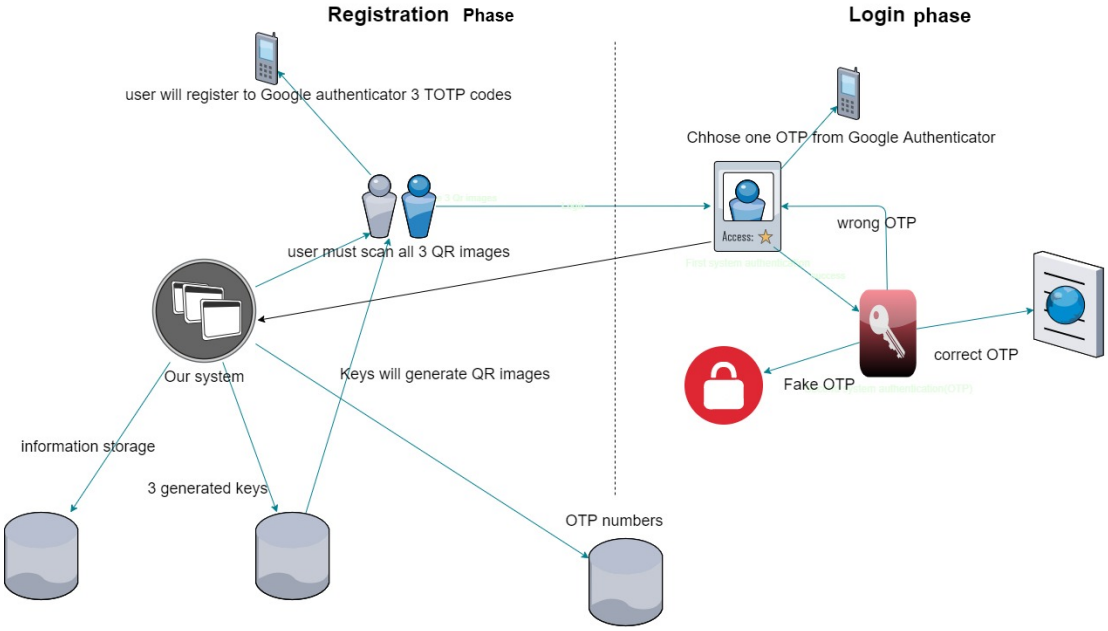

Figure 1. The layout and functioning of the proposed system.

Registration Phase

The registration process in the proposed system is designed with user simplicity in mind; while typical 2FA systems often require users to download additional software or applications, acquire physical devices like security tokens, or integrate with a mobile phone through a process that can be technically challenging for some users, our system's setup is straightforward. It solely requires users to create a password and pick a number (1–3) from a drop-down menu. The process is not only user-friendly but also eliminates many of the potential barriers inherent in other 2FA systems. Details are provided in Section 8: Evaluation.

More analytically, in the registration phase, the user should fill the registration form with some required information. She/he must first complete the mandatory fields (username, password, and OTP order number) and then the optional fields (first name and last name). The selection between mandatory and optional fields is a developer's choice and may change depending on the system.

Certain fields, like username and password, require enhanced security to prevent various types of attacks. As the user types these fields, a number of dots replace the actual content. Among the different defense mechanisms that could counteract attacks, the dot style was selected. The optional fields do not require advanced protection countermeasures. The phone number is a required field, and the user types the phone number to which they would like the OTP sent. Regarding the OTP order number, the user must choose one integer of the specified range of integers proposed to him/her. The right OTP will appear based on the user's choice. For instance, if a user selects option number 2, the second of the three OTP he/she receives during the login phase is the correct one. The username in the pilot system must be unique.

In order to register to Google Authenticator, the following process is required: Initially, three OTP codes will be generated by the system. The username is base32-encrypted, and based on this, three OTP codes are generated. The Python library pyotp is used for the production of the OTP codes, which is a library for generating and verifying one-time passwords. Our system demands uniqueness for the usernames, and also, even if an adversary obtains the username, he/she will not be able to reproduce the OTPs. Our system generates OTP codes based on time-based algorithms, such as time-based one-time passwords (TOTPs) or HMAC-based one-time passwords (HOTPs). These algorithms use a unique secret key, often referred to as the OTP seed, which is stored securely and associated

with each user account. The OTP codes are inserted in the QR images to be scanned for Google Authenticator. The QR images will be shown only in the registration phase and not in any other phase. At the same time, three OTP codes will be sent to the user's phone via SMS.

The user should scan all three QR images, instead of only one, in Google Authenticator. This precaution measure was implemented in order to avoid situations wherein the user only scans the correct QR and someone steals their phone. In this case, the thief could compare the SMS codes with the one in Google Authenticator and figure out the place of the correct OTP. Knowing the OTP order number, the user fills in the OTP field and continues. If the user wrongly types one of the other two OTP codes, then their account locks. If they mistype the OTP and enter any other code that does not belong to the generated codes, then they will be redirected to the login screen, and they will be notified of a failure to log in. If the user reaches three failed tries, then their account locks.

If a user registers more than one time, Google Authenticator can recognize the different usernames. So, the user will not be confused, as long as they remember which username has the right OTP number. In the interest of security, OTP codes will not be stored in the system's database, but instead, the selected OTP number will be safely kept in a separate database away from other user data. It is generally safer to store one-time password (OTP) codes in a separate database from the one that stores the username and password information. This is because, if an attacker gains access to the database containing the username and password information, they might also be able to access the OTP codes, compromising the security of the OTP authentication method.

By storing an OTP order number in a separate database with additional security measures, such as encryption and access controls, the risk of unauthorized access to the OTP order number can be reduced. Additionally, separating the OTP database from the user database can provide an additional layer of security in the event that one of the databases is compromised.

However, it is important to ensure that the OTP database is properly secured and that appropriate backup and recovery mechanisms are in place to prevent data loss in case of any issues. Ultimately, the choice of where to store OTP codes will depend on the system's specific security requirements and considerations.

Some key-security implementations are as follows:

Separate storage: The OTP order number of each user is stored separately from the main database that contains the genuine tokens and user data. Our system has incorporated this implementation.

Salted hashes: Honeytokens are stored as salted hashes. Salting involves adding a unique and random value (salt) to each token before hashing. This prevents attackers from easily identifying the genuine tokens. Our system also has this implementation.

Login Phase

In the login phase, the user completes a login form, which consists of two fields: the username and password. After successfully entering the login credentials, the user receives an SMS that contains the OTP codes. The codes appear in random order except for the correct one, which is always at the right place, chosen by the user during the registration phase. For example, if the user chooses OTP position number 2, then the correct OTP code will always be at position number 2. The other two will be in random order. Google Authenticator is quite the same. If the user scans the QR images from left to right, then the left one will be the first and the right one will be the third, in a three-OTP-code system.

Upon receiving the SMS, the user must complete the OTP field to gain access. The time-based nature of TOTP codes means that they have a limited validity period (usually 30 s to a few minutes) before they expire. We have a notification box that shows the user how much time the user has to write the correct OTP before it changes again. If the user types one of the honey OTPs, then their account locks. We define honey OTPs as those generated and sent to the user, but they are not correct. All other mistypes or six-digit numbers will be dealt with as mistypes. In case of a mistype, the user will be redirected to

the login form. Then, one failed try is counted. At three failed tries, their account will be locked. When an account is locked, an alert is sent by e-mail to the user.

Other Two-Factor Authentication Systems

Two-factor authentication mechanisms have gained popularity because they increase security; therefore, a great number of 2FA applications exist. Most of them use the same principle as in the proposed mechanism—a combination of OTP codes with SMS messages. The system varies and depends on the base of the application. The most popular implemented systems are listed in the sequel:

- Email-based 2FA;
- SMS-based 2FA;
- Voice-based 2FA;
- Software-token-/TOTP-based 2FA;
- Biometrics-based 2FA;
- As a push notification;
- Hardware-token-based 2FA.

Rather than sending a verification code via an application or an SMS, a more secure authentication method is to purchase a dedicated key-type MFA device. One that gains favor is the YubiKey 5C NFC. These keys generate codes that are sent when one becomes directly connected to an NFC, Bluetooth, or a USB port. Unlike smartphones, it has the advantage of being a single-purpose, security-enhanced device. However, why are they safer? A malware-infected application running on a phone could intercept the verification code generated by the phone's authenticator application. In contrast, security keys have no batteries or moving parts, and they are very durable. However, they are not as user-friendly as a smartphone.

Microsoft Authenticator works like most other authenticator applications. It generates a six- or eight-digit code on a rotating basis of about 30 s. When one signs in to an application or a service, at the final stage, the system requires a two-factor authentication code. They can then accesses the Microsoft Authenticator application to obtain these codes. This feature works a little differently on Microsoft accounts than on non-Microsoft accounts. The code can be used from this application to sign in without a password for a Microsoft account. It can also be configured to send the user push notification approvals. It also performs secondary verification using a phone's authentication method (fingerprint reader, PIN, or pattern). However, on all other account types in different applications (Facebook, Google, etc.), one must sign in with their username and password before they can add a code. Microsoft supports all websites that use the time-based one-time password (TOPT) standard. As a result, an application can regularly generate codes, and one can use them as needed. Most of the applications that a user signs into use this method, with the exception of a few banking applications.

Another one is the Authy authenticator. Soft tokens enable authentication everywhere. If a user is offline or lacks data and is unable to receive an SMS or push authentication on mobile or desktop, they can still sign in using their time-based one-time password (TOTP). A TOTP token is generated from a device-specific shared secret. This allows for per-device risk decisions compared to device-independent authentication methods, such as SMS and voice. SMS, voice, and email are universal forms of communication and verification, allowing you to reach large audiences without onboarding barriers. SMS, voice, and email authentication are vastly improved over using passwords alone, providing both international reach and accuracy.

6. Security Analysis

Security analysis is the process of evaluating the security of a system, network, or application to identify vulnerabilities and potential threats. It involves a thorough examination of the security controls and mechanisms in place, as well as an assessment of the system's overall architecture and design. Security analysts use a variety of tools and

techniques, such as penetration testing, vulnerability scanning, and threat modeling, to simulate real-world attacks and identify potential weaknesses. They also review log files and other system data to detect and analyze any unusual or suspicious activity. The goal of security analysis is to provide a detailed report of any vulnerabilities found, as well as recommendations for remediation and mitigation. This process helps organizations identify and address potential security risks before they can be exploited by attackers.

The login phase employs multi-factor authentication (MFA), which is an essential security component. It incorporates the use of something the user knows (like a password), something the user has (like an identity card or OTP number), and something the user is (like a fingerprint, if biometric measures are used).

The password, which is a knowledge factor, is hashed using Bcrypt, which makes it very resilient against various types of attacks, such as brute force and rainbow tables. Even if an attacker obtains the hash, it would be computationally expensive and time-consuming to reverse engineer the original password.

The second factor, the OTP number, is encrypted and changes after every successful login, making it a moving target that is challenging for attackers to exploit. This OTP number serves as an additional hurdle for any unauthorized user, even if they managed to obtain the correct password.

In the case of a biometric factor, like a fingerprint, it adds another layer of security. Instead of biometrics, we used the OTP order number, and its usage further reduces the chances of unauthorized access.

Finally, to guard against brute force or dictionary attacks, we have implemented rate limiting on login attempts. After a certain number of failed attempts, the user account is temporarily blocked, further strengthening the implemented security measures.

We are investigating how to make the user authentication system in this model more secure by using honeytokens. We have maintained all the important elements without making the login process more difficult or decreasing the effectiveness of honeytokens. This section discusses various types of attacks and how resistant our system is to them. According to research by Ferrag et al. (2020), our system is able to withstand these attacks [22].

Data Breach

Hashing algorithms cannot reverse the process of turning plaintext into a hash value, but they are still vulnerable to security breaches [23]. Attackers can use various techniques, such as dictionary attacks and brute-force attacks, to try to crack a hash value; while adding a salt value to the plaintext before hashing can increase security, it is not foolproof. The two-factor honeytoken authentication (2FHA) method can prevent security breaches by requiring the attacker to guess the correct one-time password (OTP) among a series of false ones.

Stalkerware

Attacker spyware can be installed on a user's phone, allowing the attacker to monitor activity, access information, and even listen in on conversations [24]. However, the 2FHA system is resistant to this type of spyware because, even if the attacker is able to read the OTPs sent to the user's phone, they will not be able to determine which OTP is the correct one among the false ones. To mitigate this threat, it is crucial to emphasize the importance of implementing strong security measures, such as frequently changing the OTP seed (OTP order number) or utilizing additional layers of authentication, to prevent unauthorized access, even in the presence of such observation. The attacker cannot see which OTP code in our system is at the OTP box, so they cannot understand which one the user is writing each time.

SIM Swapping

The 2FHA system is robust against SIM swapping attacks [25], a common method used to gain access to accounts for bitcoin communities, banks, social media, and email. These attacks involve swapping a SIM card to bypass two-factor authentication. However, the 2FHA system is resistant to this type of attack because only the legitimate user knows the location of the real OTP among the fake ones.

Video-Recording Attack

Our proposed method offers protection against video-recording attacks [26] by requiring the use of both username and password, as well as making it difficult for an attacker to scan QR images to gain access to Google Authenticator. These measures make it difficult for an attacker to successfully carry out this type of attack without the user noticing.

Guessing Attack

To defeat our system, an attacker must correctly guess not only the password but also the correct one-time password out of a series of three options. Even if they manage to gain access to the first layer of security [27], they must also correctly identify the true OTP. If they make the wrong choice, their attempts will be halted and the account will be locked. The likelihood of a successful attack decreases with longer passwords, making our system more secure.

Side-Channel Attack

A side-channel attack is a form of a security breach that gathers information from the physical implementation of a system, exploiting the data produced by the device during normal operation. Examples of such data include timing information, power consumption, electromagnetic leaks, or even sound, which could potentially reveal sensitive information about the user's login credentials or the system's internal processes [28]. However, in the context of the proposed system, even if an attacker was able to execute a successful side-channel attack to obtain the user's login credentials, they would still face the additional security layer of the one-time password (OTP). Unless the correct OTP is entered within the valid timeframe, unauthorized access to the system remains effectively blocked. This serves as an additional deterrent and obstacle for potential attackers, enhancing the overall security of our system.

Password Spraying

The "password spraying attack", a type of brute-force attack, involves a malicious actor repeatedly trying the same password on multiple accounts before moving on to another password and repeating the process [29]. This tactic is often successful because many people use similar passwords. In our system, this approach will not work because we have a limit of three wrong passwords per user. If a user enters their password incorrectly three times, their account will be locked, and they will be notified.

Among the parameters that were examined, usability, user experience, scalability, and flexibility were chosen for the security analysis of the proposed system.

Usability and user experience: Two-factor authentication systems should be easy to use and not overly burdensome for users. If the system is too complex or problematic, users may be less likely to use it or may resort to weaker authentication methods.

Scalability and flexibility: As the organization grows and the number of users increases, the two-factor authentication system should be able to scale to meet the demand. Additionally, the system should be flexible enough to accommodate different types of users and use cases, such as remote workers or third-party contractors.

We have examined a variety of potential attacks on our system and have found that our suggested approach is resistant to all of them. Each security layer we have implemented requires the attacker to pass through it before gaining access to a user account, making it extremely difficult for them to succeed, and while this approach is secure, it does not add any additional difficulty to the user's authentication process.

The attacks described above were examined in a virtual environment with the use of a pilot program that permitted the examination of simulation scenarios. Details regarding these attack simulations and the way they contributed to our security analysis scenarios are given below.

Video-Recording Attack

This attack includes a video camera, so we used a smartphone, and we recorded a video via Windows. From the attacker's side, the person who recorded the video, any attempt to steal the password of the user failed due to the password element in the HTML. For the OTP, in Google Authenticator, video recording does not work due to Google's

safety protocol, and neither does a screenshot. Information an attacker may target could not be stolen from the user, so the attack was not successful on the proposed system. We mentioned the screen-recording protection feature on a smartphone as an example of a measure that can safeguard against unauthorized screen-capture attempts carried out through software-based screen-recording methods. However, it is important to note that this protection does not directly impact the capabilities of an external video camera used for video recording attacks. The reference to smartphone screen-recording protection was intended to highlight the importance of comprehensive security measures in protecting sensitive information displayed on screens, recognizing that different types of attacks may require different countermeasures. In our system, the OTP box, which must be filled by the user, will be displayed as dots in the field edit box to counteract these types of attacks.

Guessing Attack

The guessing attack requires the attacker to guess the username, password, OTP number order, and the OTP. It is very difficult for this attack to succeed because of the many pieces of private information that should be guessed.

Brute-Force Attack

In a brute-force attack, the attacker tries to obtain the username and password fields. The limited number of attempted failures effectively protects the system. If a user fills in the wrong password three times, the account locks; thus, the attack works very poorly against this system.

Authentication systems can be vulnerable to various types of attacks, as shown in Table 2. However, our 2FHA system is resistant to these attacks. One potential weakness is the registration phase, where the user selects the correct OTP number. One solution could be to send the correct number to the user via a phone call or email, or a similar method used by banks for credit cards to protect against a man-in-the-middle attack.

Table 2. The resistance of the 2FHA system against various forms of attacks.

Attacks	Robust
Stalkerware	✓
SIM swapping	✓
Video-recording attack	✓
Guessing attack	✓
Side-channel attack	✓
Man-in-the-middle attack	✓
Brute-force attack	✓
Dictionary attack	✓
Password spraying	✓

Threat Modeling

Threat modeling is a technique for examining the risk built into a system's architecture. This method, which has its roots in application security circles, has been proven to work well for both IT and OT systems in a variety of situations (operational technology). For industrial sites, nuclear plants, wind turbines, and smart grids, we have successfully run threat models.

It is a means of "shifting left", which refers to ensuring that security is taken into account as early as feasible when building a system. However, TM may also be used to examine an existing system.

STRIDE [30] is the pinnacle of the strategy of threat modeling. Prior to any coding, it is essential to identify and eliminate any potential weaknesses. To construct networks, systems, and applications that are inherently secure, it is crucial to begin by embracing threat-modeling techniques. To guarantee the safety of application design, one can implement a risk framework known as STRIDE to achieve this goal:

1. Spoofing identity;
2. Tampering with data;
3. Repudiation;
4. Information disclosure;
5. Denial of service;
6. Elevation of privilege;

Spoofing Identity

This type of threat is characterized by an individual assuming the identity of another user. A typical example of this is when an attacker impersonates an administrator. In our case, the attacker, if they manage to take the role of admin, would not be a serious problem. The admin has permission to change the username and the password of each user but, to alter the password, will need the previous authenticated password to proceed. The admin cannot see the in-use password of the users; they are hashed. The admin does not have the authority to change the OTP number that each user has, and neither can see it. The OTP number is cryptography-enhanced or with some hashing method (Bcrypt, in our case). The role of the admin in our system is to see the usernames with read-only permission and nothing else. Our system mitigates this risk through the implementation of hashed passwords and encrypted OTP numbers. The role of the admin in our system is tightly controlled, limiting the potential for abuse of admin privileges.

Tampering with Data

Data tampering refers to the malicious alteration of information through illicit methods, such as elimination, manipulation, or editing. Data can be tampered with whether they are at rest or in motion. As data travel through digital channels, they can fall prey to interceptors looking to make changes. Data transfer is the backbone of electronic communication. Our system utilizes Bcrypt hashing, preventing illicit changes to passwords. Furthermore, OTP numbers are tied to individual users and any changes would require the corresponding password.

This threat is very interesting for our system. If the attacker can replace the OTP number with one of their own, they must have also the password to achieve the goal they want. The OTP number depends on the user (the OTP number is a foreign key to the user). The attacker must also replace the password of the user they want to access, but that cannot be performed so easily. The Bcrypt [31] method of hashing we used for the password does not allow the password to be changed with another.

Repudiation

A program or technology that lacks the necessary safeguards to diligently track and record user actions, leaving it susceptible to manipulation and false identification of new activities, is a prime target for a repudiation attack. By altering the authorship information of malicious user-performed operations, this attack may be exploited to incorrectly log data to log files. Similar to how spoofing mail messages are used, its use may be expanded to include generic data manipulation under others' names. In the event of this assault, the information recorded in log files may be deemed false or deceptive. Our system has a tracking history of user actions, like when the username was created and the exact time of every login and logout, so we can keep an eye on every user to prevent any malicious attack. We do not use API, and passwords and all the information is either hashed or encrypted, so the user cannot pretend to be someone else. The threat of "Repudiation" is countered by maintaining a detailed log of user actions. The log history, combined with the hashed and encrypted data, deters and prevents unauthorized actions and false identification.

Information Disclosure

Information disclosure, also known as information leaking, is when a website inadvertently reveals sensitive information to its visitors. This can include a wide range of data, such as information about other users, including usernames, financial details, and more. It is a careless mistake made by websites that can cause severe consequences if not addressed properly. "Information Disclosure" is significantly limited as our system does not use

APIs, and all sensitive information is hashed or encrypted. The potential for leakage of information is thereby minimized.

Our system is not built with API, so the leak information is limited a lot. The system also does not mention any database or tables or columns at any point. The only information that can be leaked is the username of the user but that also can hide it.

Denial of Service

A denial-of-service (DoS) attack is a malicious attempt to obstruct a legitimate user from utilizing resources to which they are entitled to access. This can disrupt an application's operations, data storage, and data-processing capabilities. For Django, the vulnerable versions of the web framework are 4.0–4.1.1 and 3.2–3.2.15. We have version 3.1.6, so we do not have to worry about DOS attacks. To counter "denial-of-service" (DoS) attacks, we have ensured the Django version used in our system is not vulnerable to such attacks. We utilize Django version 3.1.6, which is not susceptible to known DOS attacks.

Elevation of Privilege

When an application acquires privileges or powers that are not appropriate for it, this is known as an elevation of privilege. Many of the vulnerabilities for privilege elevation are also used against other threats. For instance, cunning buffer overflow attacks aim to create executable codes.

Lastly, for "elevation of privilege", our system is designed such that an application cannot acquire more privileges or powers than it should have. This, combined with other safeguards, mitigates the majority of vulnerabilities related to privilege elevation. We use Django version 3.1.6 for our system. Django is a high-level Python web framework that encourages rapid development and a clean, pragmatic design. Affected versions (2.1, 2.1.15; 2.2, 2.2.8) of this package are vulnerable to privilege escalation. Imagine you have a Django model admin that displays related models inline, where the user has read-only access to the parent model, but has editing permissions to the inline model. In this case, the user would be presented with an editing UI that allows POST requests to update the inline model. However, it would not be possible to directly edit the read-only parent model, but the parent model's save() method would still be called, potentially triggering side effects and activating pre- and post-save signal handlers. This situation can be tricky and should be handled with caution.

7. 2FHA as a Service

Two-factor authentication (2FHA) is a security measure that requires users to provide two forms of identification in order to access an account or system. One form of identification is typically a password, and the other is a unique, time-based one-time password (OTP) that is generated and sent to the user's phone via SMS.

Two-factor authentication (2FA) as a service is a security solution that allows companies to add an additional layer of protection to their user authentication process. It is typically delivered as a cloud-based service and can be integrated into an organization's existing systems and infrastructure.

With 2FA, users are required to provide two forms of identification when logging into a system or application: the first is a traditional form of authentication, such as a password, and the second is a unique, one-time code generated by a device or service, such as a mobile app or text message.

By requiring two forms of identification, 2FA makes it much more difficult for attackers to gain unauthorized access to an organization's systems and data. Additionally, 2FA service providers can offer additional security features, such as risk-based authentication, to further protect user and company data.

Using a 2FA as a service can help companies to meet regulatory compliance and industry standards, and also provides an added level of protection against cyber attacks, data breaches, and other security threats.

A company can benefit from implementing 2FHA in several ways:

Increased security

2FHA adds an additional layer of security to the login process, making it much harder for unauthorized users to gain access to sensitive information.

Reduced risk of breaches

With 2FHA in place, even if an attacker manages to obtain a user's password, they will not be able to access the account without also having the correct OTP, which is sent to the user's phone and is valid for a limited time.

Compliance

Multi-factor authentication is often a requirement for compliance with industry standards and regulations, such as the Payment Card Industry Data Security Standard (PCI DSS) and the Health Insurance Portability and Accountability Act (HIPAA).

User convenience

SMS-based 2FHA is a simple and convenient method for users to access their accounts without the need for additional hardware, such as a security token.

Better user experience

SMS-based 2FHA also makes it easy for users to recover their accounts in case they forget their password, as they can simply request a new series of OTP codes to be sent to their phone.

Cost-effectiveness

SMS-based 2FHA is relatively inexpensive to implement and maintain, and it does not require any additional hardware or software.

Please note that it is important to keep in mind that SMS-based 2FA has some security risks, as an attacker may intercept or redirect the SMS message to gain access to the account. A 2FHA mechanism can cope with such attacks.

Moreover, a 2FHA mechanism can be offered as a service to companies from a company whose main service is the creation of multiple OTPs for clients (let us name this company **"Factorization of Authentication: FofA"**). The companies that use the services of FofA can be separated into two groups: the first group includes companies that have their own 2FA mechanism (authentication system: TOTP server and hashed passwords), and the second groups are the companies that do not have any 2FA system.

FofA will be mainly connected with companies and not the end clients of them. Each message that will be exchanged between FofA and the companies will be encrypted through the use of asymmetric cryptography.

In the first case, company A will send the produced OTP to FofA through an API and FofA will generate the following OTPs, bundle them into a message, and send them to the client. The information that A needs to send to FofA is the mobile phone number of the user, the produced OTP, and the number that the user has picked as the preferred one (the valid OTP in the list of produced ones, e.g., 5). Figure 2 demonstrates the first case with a company that incorporates the proposed method in its already used 2FA system.

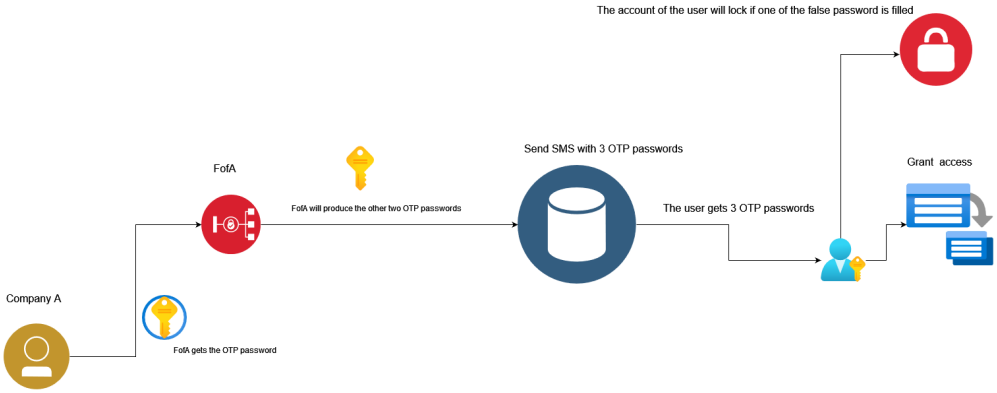

Figure 2. A diagram that shows how a company that already has a two-factor authentication system can work with our system.

In the latter case, company B will redirect the user to a website of FofA where they will need to enter the correct OTP out of the ones that FofA will send to their mobile phone. At the same time, company B will send the mobile phone number of the user and the preferred OTP number to FofA. FofA will produce all N OTPs, send them to the user, and wait for the correct answer from the user. Once the answer is received, FofA will check its validity and redirect the user back to the main website of company B while also sending a confirmation message to company B. Figure 3 shows how a company incorporates the proposed method although it does not use a 2FA system.

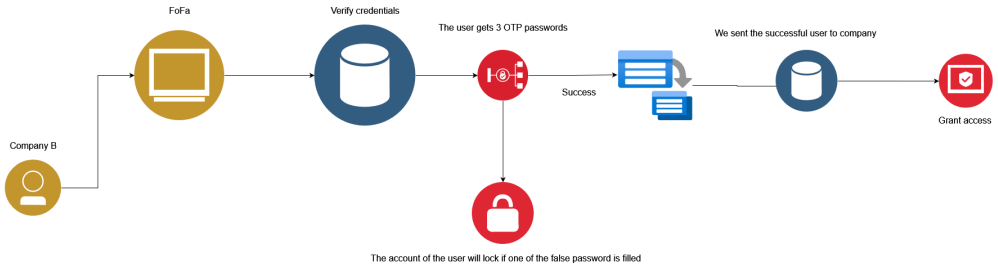

Figure 3. A diagram that shows how a company that does not have two-factor authentication system can work with our system.

Companies will have the option to choose from a variety of methods that are suitable for their operation. Clients of the companies can also have that option as well, depending on their device and how familiar they are with mainstream technologies, for example, if the clients do not have smartphones they cannot use Google Authenticator, so the only option they will have is obtaining their OTP numbers via messages. The variety of clients can be high, but the companies will not have to change their technology to adjust to mainstream technology. The information exchanged with the clients will be encrypted for safety purposes, and the companies will only store the data that are necessary for the operation. 2FHA can also be licensed to companies providing security services to their clients, packaged with consultancy to enable the integration of 2FHA into the licensee's incumbent software platform, for example, the Android operating system and Apple iOS, or security infrastructure components.

8. Evaluation

The response time is an important factor, especially for applications that produce OTP numbers and need to interact in real time with customers. We often receive an SMS containing the OTP code late; in some cases, even 30 s later than the creation of the code. Sometimes, if the application is very complex, it can impose further delays.

We used the response time parameter because the proposed system includes an SMS notification and Google Authenticator. Any delay in the SMS transmission might confuse the user and delay the transaction. The factors that affect the response time in the proposed system are the QR images and the hashing passwords. The most time-delaying elements are the QR images, as the system has to produce the corresponding images; thus, this is why additional emphasis should be given to this factor. The SMS factor does not depend on the system because it is used by a third-party tool called Twilio. The OTPs are built by the proposed system using the pyotp library, which is embedded in Python. The time required for the composition of one OTP is very small, and Table 3 shows the required time.

When using GA as a fallback option for OTP code delivery, it is crucial to establish clear rules or conditions that determine when the system will accept GA OTPs instead of SMS OTPs.

Timing window: We can define a specific time window within which both the SMS OTP and GA OTP will be considered valid. For example, if the SMS OTP is generated within the last 60 s, but the GA OTP falls within the current 30 s window, the system can

prioritize the GA OTP over the SMS OTP. For the pilot mode, we do not have this option, but this can be further enhanced in future roll-outs of the system.

Table 3. Time calculation for each set of the QR images.

QR Images	1 Calculation	2 Calculations	3 Calculations
3	0.373821496963501 ms	0.34929943084716797 ms	0.3513021469116211 ms
5	0.3903353214263916 ms	0.36231279373168945 ms	0.35980987548828125 ms
7	0.40434694290161133 ms	0.38182902336120605 ms	0.38633203506469727 ms

We tested our system with three, five, and seven QR images and calculated the time of the application for each run. We ran the program for each set three times and measured the time. With three QR images, the times were 0.373821496963501 ms, 0.34929943084716797 ms, and 0.3513021469116211 ms. With five QR images, the times were 0.3903353214263916 ms, 0.36231279373168945 ms, and 0.35980987548828125 ms. We can observe that the times are slightly bigger for the previous ones with three QR images. With seven QR images, the times were 0.40434694290161133 ms, 0.38182902336120605 ms, and 0.38633203506469727 ms. We can observe that the number is even bigger for five QR images. The reason is that the program needs more time to produce more QR images and OTPs. The difference is very small, and in real time, we would not understand any delay.

We tested the system in a Windows 10 environment, with 16 GB RAM at 2999 MHz, an NVMe M2 500 GB SSD, and a Ryzen 5-2600 CPU at 3.6 GHz. The IDE we used to do the test was PyCharm Community Edition 2022.3 and Python version 3.7.6. We calculated the time each process took using the time library. The system functionality was presented at the 14th EAI International Wireless Internet Conference (EAI WiCON 2021), and a demo of the pilot system was demonstrated and used by some users that tested it and realized how it works.

There are many different hash functions that can be used to hash passwords, and the speed at which they operate can vary significantly.

Some common hash functions include MD5, SHA-1, and SHA-256. In general, faster hash functions are less secure than slower ones because they can be more susceptible to attacks such as brute-force attacks, in which an attacker tries to guess a password by hashing many different possible passwords in a short amount of time.

Below, a high-level comparison of the speed of some common hash functions is presented:

1. MD5: fast;
2. SHA-1: faster than MD5;
3. SHA-256: slower than SHA-1.

Keep in mind that the actual speed of these hash functions can vary depending on the specific implementation and the hardware they are running on. Additionally, some hash functions may be optimized for certain types of hardware, such as GPUs or CPUs, which can affect their relative speeds.

In general, it is important to use a secure, slow hash function for hashing passwords rather than a faster but less secure one. This helps to protect against brute-force attacks and other types of password cracking.

Bcrypt is a password hashing function designed to be computationally expensive to compute in order to increase the amount of time and resources required to successfully attack a password. As such, it is generally slower than non-cryptographic hash functions, such as MurmurHash, FNV, CityHash, SpookyHash, and xxHash.

The exact speed of Bcrypt depends on a number of factors, including the hardware and software environment in which it is being run, the size of the input data, and the specific parameters being used (such as the cost factor). In general, Bcrypt is considered to be a

relatively slow hashing function, with a speed of around 100–1000 nanoseconds per byte on modern hardware. This is significantly slower than non-cryptographic hash functions, which can operate at speeds of 1–5 nanoseconds per byte or faster on modern hardware. However, the slower speed of Bcrypt is an intentional design feature, as it helps to make it more resistant to attacks by increasing the amount of time and resources required to compute the hash.

We calculate the time that the function takes from hashing the password and the OTP number to storing them in a database (See Table 4). We tested some well-known hashing methods. For a five-letter password and a hashed OTP number using Bcrypt, the time was 0.5097146034240723 ms. For pbkdf2 sha256, the time was 0.2055501937866211 ms. For Bcrypt sha256, the time was 0.5090336799621582 ms. For argon2, the time was 0.02352142333984375 ms. As can be observed, Bcrypt is the most time-demanding, but in terms of a real-time environment, the time is relatively small, and we cannot observe a difference.

Table 4. Hashing methods and time calculation for hashing and storing passwords (five-letter password and OTP number.

Hashing Methods	Time (ms)
Bcrypt	0.0.5097146034240723 ms
pbkdf2 sha256	0.02352142333984375 ms
Bcrypt sha256	0.5090336799621582 ms
Argon2	0.02352142333984375 ms

In Django, the recommended way to store passwords is to use the built-in password hashers provided by the Django authentication system. Django provides a number of different password hashers to choose from, and the best one for your project will depend on your specific needs and requirements.

Some of the most commonly used password hashers in Django include the following:

PBKDF2PasswordHasher is the default password hasher in Django, and it is considered to be a secure and reliable choice for most applications. It uses the Password-Based Key Derivation Function 2 (PBKDF2) [32] algorithm to hash passwords.

Argon2PasswordHasher uses the Argon2 algorithm [33], which is considered to be one of the most secure and efficient password-hashing algorithms available. It is a good choice for applications that require the highest level of security.

BCryptSHA256PasswordHasher uses the BCrypt algorithm to hash passwords, and it is considered to be a secure and reliable choice for most applications. It is a good alternative if you want to use the BCrypt algorithm.

SHA1PasswordHasher uses Secure Hash Algorithm 1 (SHA-1) to hash passwords; while SHA-1 is no longer considered to be a secure choice for password hashing, it is included in Django for backwards compatibility with older Django projects that may be using it.

9. Future Improvements

A future enhancement would be to combine our method with passwordless authentication. Passwordless authentication is a measure of verifying a user's identity without using a password. Instead, passwordless authentication uses more secure alternatives, such as possessive factors (one-time passwords (OTPs) and enrolled smartphones) and biometrics (fingerprints and retinal scans). It can be accomplished in various ways (biometrics, possession factors, and magic links). Passwordless authentication works by substituting passwords with other inherently more secure authentication factors. Password-based authentication checks the password supplied by the user against one saved in the database.

Some passwordless systems, such as biometrics, perform comparisons in a similar fashion, but instead of comparing passwords, they compare users' identifying characteris-

tics. For example, the system captures a user's face, extracts numeric information from it, and compares it with verified data held in a database. Other passwordless implementations may compare information differently. For example, a system will send a one-time passcode to a user's mobile phone. The user retrieves it and enters it into the login field. The system then compares the passcode entered by the user with the passcode sent. Passwordless authentication is based on the same principles as digital certificates.

Blockchain-based authentication methods have recently gained attention because they combine decentralized solutions, achieve mutual trust, and help avoid additional overheads [34]. Moreover, blockchain can provide authentication logs for auditing purposes, verify one-time passwords (OTPs) that are produced by authentication mechanisms, and ensure the integrity of data stored in it. In this sense, we are exploring the integration of blockchain technology to further enhance security, as well as integration with the SNE2EE method, for the end-to-end encryption of communication between websites and users [35].

Finally, in the future, we aim to improve our defense system and make it platform-agnostic. We plan to demonstrate proof of concept by integrating the 2FHA mechanism into banking and healthcare environments, which have been primary targets of cyber attacks and currently rely on simple OTP systems for online transactions.

10. Discussion and Conclusions

In this paper, we proposed a novel approach that combines honeytokens with a two-factor authentication system. The method we suggest is effective in defending against various types of attacks, including SIM swapping, and adds an extra layer of security for the user without compromising ease of use.

Our proposed approach combines honeytokens with a two-factor authentication system, making it more difficult for attackers to successfully breach the system. It can be easily integrated into existing security solutions and offers the option of sending simple SMS messages for those less familiar with using smartphone apps. We examined the operation of the proposed method for a number of forty users. By evaluating the obtained results, we further examined whether an expansion to a larger number of users would be feasible, as a test including thousands of users would be too costly. Each SMS costs because it is derived from a third-party tool, Twilio, and it is difficult to calculate the precise costs and requirements at this stage of the work. Moreover, each company has different technologies and perspectives. Scaling, however, for a large number of users does not require any change to the proposed system, which would not have any problems in servicing them. Other limitations related to the implementation platform will be examined in future work, e.g., in the health care environment or banking systems.

Ensuring the security of one aspect of a system (online user authentication) can protect a system against various types of attacks. However, to make the system secure against all attacks, a comprehensive approach must be taken. There are various maturity models and risk management processes that can be utilized to evaluate an organization's security posture, but these must be tailored to fit a specific system. For systems connected to critical infrastructures, it is especially important to have a high level of security and to use multiple advanced security measures.

Author Contributions: Conceptualization, V.P. and L.M.; methodology, L.M. and M.P.; software, V.P. and I.K.; validation, C.D. and M.A.F.; formal analysis, M.A.F. and L.M.; investigation, V.P. and H.J.; resources, V.P. and M.P.; data curation, H.J.; writing—original draft preparation, V.P. and M.P.; writing—review and editing, L.M., I.K. and H.J.; visualization, V.P. and C.D.; supervision, L.M., C.D. and I.K.; project administration, C.D. and H.J.; funding acquisition, C.D. All authors have read and agreed to the published version of the manuscript.

Funding: This work received funding from the European Union's Horizon 2020 research and innovation program: project CyberSec4Europe (grant agreement no.: 830929).

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

References

1. Florêncio, D.; Herley, C.; Coskun, B. Do strong web passwords accomplish anything? *HotSec* **2007**, *7*, 159.
2. Leandros, L.; Kantzavelou, I. *Cybersecurity Issues in Emerging Technologies*; CRC Press: Boca Raton, FL, USA, 2021.
3. Wang, D.; Wang, P. Two birds with one stone: Two-factor authentication with security beyond conventional bound. *IEEE Trans. Dependable Secur. Comput.* **2016**, *15*, 708–722. [CrossRef]
4. Lai, R.W.; Egger, C.; Schröder, D.; Chow, S.S. Phoenix: Rebirth of a cryptographic password-hardening service. In Proceedings of the 26th USENIX Security Symposium (USENIX Security 17), Vancouver, BC, USA, 16–18 August 2017; pp. 899–916.
5. Juels, A.; Rivest, R.L. Honeywords: Making password-cracking detectable. In Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security, Berlin, Germany, 4–8 November 2013; pp. 145–160.
6. Genç, Z.A.; Kardaş, S.; Kiraz, M.S. Examination of a New Defense Mechanism: Honeywords. In Proceedings of the Information Security Theory and Practice, Crete, Greece, 28–29 September 2017; Hancke, G.P., Damiani, E., Eds.; Springer International Publishing: Cham, Switzerland, 2018; pp. 130–139.
7. Palaniappan, S.; Parthipan, V.; Stewart kirubakaran, S.; Johnson, R. Secure User Authentication Using Honeywords. In Proceedings of the International Conference on Computer Networks, Big Data and IoT (ICCBI-2018), Madurai, India, 19–20 December 2018; Pandian, A., Senjyu, T., Islam, S.M.S., Wang, H., Eds.; Springer International Publishing: Cham, Switzerland, 2020; pp. 896–903.
8. Li, W.; Wang, P. Two-factor authentication in industrial Internet-of-Things: Attacks, evaluation and new construction. *Future Gener. Comput. Syst.* **2019**, *101*, 694–708. [CrossRef]
9. Papaspirou, V.; Maglaras, L.; Ferrag, M.A.; Kantzavelou, I.; Janicke, H.; Douligeris, C. A novel Two-Factor HoneyToken Authentication Mechanism. In Proceedings of the 2021 International Conference on Computer Communications and Networks (ICCCN), Athens, Greece, 19–22 July 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 1–7.
10. Papaspirou, V.; Papathanasaki, M.; Maglaras, L.; Kantzavelou, I.; Douligeris, C.; Ferrag, M.A.; Janicke, H. Security Revisited: Honeytokens meet Google Authenticator. In Proceedings of the 2022 7th South-East Europe Design Automation, Computer Engineering, Computer Networks and Social Media Conference (SEEDA-CECNSM), Ioannina, Greece, 23–25 September 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 1–8.
11. Dasgupta, D.; Roy, A.; Nag, A. *Advances in User Authentication*; Springer: Cham, Switzerland, 2017.
12. Papathanasaki, M.; Maglaras, L.; Ayres, N. Modern Authentication Methods: A Comprehensive Survey. *AI Comput. Sci. Robot. Technol.* **2022**. [CrossRef]
13. Ometov, A.; Petrov, V.; Bezzateev, S.; Andreev, S.; Koucheryavy, Y.; Gerla, M. Challenges of multi-factor authentication for securing advanced IoT applications. *IEEE Netw.* **2019**, *33*, 82–88. [CrossRef]
14. Aggrawal, N. Authentication methods: A review. *Productivity* **2012**, *52*, 243.
15. Polleit, P.; Spreitzenbarth, M. Defeating the Secrets of OTP Apps. In Proceedings of the 2018 11th International Conference on IT Security Incident Management & IT Forensics (IMF), Hamburg, Germany, 7–9 May 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 76–88.
16. Kantzavelou, I.; Tzikopoulos, P.F.; Katsikas, S.K. Detecting Intrusive Activities from Insiders in a Wireless Sensor Network using Game Theory. In Proceedings of the 6th International Conference on PErvasive Technologies Related to Assistive Environments, Rhodes, Greece, 29–31 May 2013; pp. 1–8.
17. Haller, N.; Metz, C.; Nesser, P.; Straw, M. *A One-Time Password System*; RFC Editor RFC 2289, IETF Request for Comments 2289; IETF: Fremont, CA, USA, 1998. [CrossRef]
18. Chowhan, R.S.; Rohit, T. Password-Less Authentication: Methods for User Verification and Identification to Login Securely Over Remote Sites. In *Machine Learning and Cognitive Science Applications in Cyber Security*; Khan, M.S., Ed.; IGI Global: Hershey, PA, USA, 2019; Chapter 8, pp. 190–212.
19. M'Raihi, D.; Machani, S.; Pei, M.; Rydell, J. *TOTP: Time-Based One-Time Password Algorithm*; RFC 6238, Portwise, RFC 6238; IETF: Fremont, CA, USA, 2011; ISSN 2070-1721. [CrossRef]
20. Srinivas, S.; Balfanz, D.; Tiffany, E.; Czeskis, A.; Alliance, F. Universal 2nd factor (U2F) overview. In *FIDO Alliance Proposed Standard*; FIDO Alliance: Wakefield, MA, USA, 2015; Volume 15, pp. 1–5.
21. M'Raihi, D.; M'Raihi, D.; Hoornaert, F.; Naccache, D.; Bellare, M.; Ranen, O. *HOTP: An HMAC-Based One-Time Password Algorithm*; RFC 4226; IETF: Fremont, CA, USA, 2005. [CrossRef]
22. Ferrag, M.A.; Maglaras, L.; Derhab, A.; Janicke, H. Authentication schemes for smart mobile devices: Threat models, countermeasures, and open research issues. *Telecommun. Syst.* **2020**, *73*, 317–348. [CrossRef]
23. Dolezel, D.; Alexander, M. Cyber-Analytics: Identifying Discriminants of Data Breaches. *Perspect. Health Inf. Manag.* **2019**, *16*, 17.
24. Han, Y.; Roundy, K.A.; Tamersoy, A. Towards Stalkerware Detection with Precise Warnings. In Proceedings of the Annual Computer Security Applications Conference (ACSAC '21), Virtual, 6–10 December 2021; Association for Computing Machinery: New York, NY, USA, 2021; pp. 957–969. [CrossRef]
25. Jover, R.P. Security Analysis of SMS as a Second Factor of Authentication: The Challenges of Multifactor Authentication Based on SMS, Including Cellular Security Deficiencies, SS7 Exploits, and SIM Swapping. *Queue* **2020**, *18*, 37–60. [CrossRef]
26. Javed, A.R.; Jalil, Z.; Zehra, W.; Gadekallu, T.R.; Suh, D.Y.; Piran, M.J. A comprehensive survey on digital video forensics: Taxonomy, challenges, and future directions. *Eng. Appl. Artif. Intell.* **2021**, *106*, 104456. [CrossRef]

27. Cha, S.; Kwag, S.; Kim, H.; Huh, J.H. Boosting the Guessing Attack Performance on Android Lock Patterns with Smudge Attacks. In Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security (ASIA CCS '17), Abu Dhabi, United Arab Emirates, 2–6 April 2017; Association for Computing Machinery: New York, NY, USA, 2017; pp. 313–326. [CrossRef]
28. Gattu, N.; Khan, M.N.I.; De, A.; Ghosh, S. Power Side Channel Attack Analysis and Detection. In Proceedings of the 39th International Conference on Computer-Aided Design, Paris, France, 7–9 October 2020; Association for Computing Machinery: New York, NY, USA, 2020. [CrossRef]
29. MITRE. Password Spraying. 2020. Available online: https://attack.mitre.org/techniques/T1110/003/ (accessed on 11 February 2020).
30. Khan, R.; McLaughlin, K.; Laverty, D.; Sezer, S. STRIDE-based threat modeling for cyber-physical systems. In Proceedings of the 2017 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe), Turin, Italy, 26–29 September 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 1–6.
31. Sriramya, P.; Karthika, R. Providing password security by salted password hashing using bcrypt algorithm. *ARPN J. Eng. Appl. Sci.* **2015**, *10*, 5551–5556.
32. Ertaul, L.; Kaur, M.; Gudise, V.A.K.R. Implementation and performance analysis of pbkdf2, bcrypt, scrypt algorithms. In Proceedings of the International Conference on Wireless Networks (ICWN), Las Vegas, NV, USA, 25–28 July 2016; p. 66.
33. Biryukov, A.; Dinu, D.; Khovratovich, D. Argon2: New generation of memory-hard functions for password hashing and other applications. In Proceedings of the 2016 IEEE European Symposium on Security and Privacy (EuroS&P), Saarbruecken, Germany, 21–24 March 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 292–302.
34. Catalfamo, A.; Ruggeri, A.; Celesti, A.; Fazio, M.; Villari, M. A Microservices and Blockchain Based One Time Password (MBB-OTP) Protocol for Security-Enhanced Authentication. In Proceedings of the 2021 IEEE Symposium on Computers and Communications (ISCC), Athens, Greece, 5–8 September 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 1–6.
35. Maglaras, L.; Ayres, N.; Moschoyiannis, S.; Tassiulas, L. The end of eavesdropping attacks through the use of advanced end to end encryption mechanisms. In Proceedings of the IEEE INFOCOM 2022-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), New York, NY, USA, 2–5 May 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 1–2.