

Auth4App: Streamlining Authentication for Integrated Cyber-Physical Environments

Vagner Ereno Quincozes^a, Rodrigo Brandão Mansilha^b, Diego Kreutz^{b,*}, Charles Christian Miers^c, Roger Immich^d

^a*Institute of Computing,
Federal Fluminense University - UFF,
Niterói, Brazil*

^b*Graduate Program in Software Engineering,
Federal University of Pampa - UNIPAMPA,
Alegrete, Brazil*

^c*Graduate Program in Applied Computing,
Santa Catarina State University - UDESC,
Joinville, Brazil*

^d*Digital Metropolis Institute
Federal University of Rio Grande do Norte (UFRN),
Natal, Brazil*

Abstract

The growing integration of mobile applications for user authentication has revolutionized user interactions with digital platforms, offering novel possibilities in user experience (UX). However, this paradigm shift poses significant security challenges. Leveraging smartphones for authentication purposes provides convenient and swift access to services, streamlining user interactions with various platforms through simple taps. Several institutions adopt static QR Codes generated from primary, unchanging user data (e.g., individual citizen national identification numbers) for physical authentication procedures like access turnstiles. However, relying on static data introduces critical security vulnerabilities as this data is susceptible to compromise. Implementing an One-Time Authentication Code (OTAC) approach appears promising in addressing these issues. Nevertheless, the absence of an integrated solution for developing a physical authentication process using OTAC leads to suboptimal API user experiences (UX APIs) and subsequent security vulnerabilities. In response to this challenge, we introduce Auth4App, a protocol set designed for identification and authentication using mobile applications. Auth4App comprises two core protocols: one dedicated to linking user credentials to mobile devices (i.e., identification), and the other for generating OTAC. We showcase the adaptability and practicality of Auth4App through three distinct case studies: a mobile-only scenario, integration of mobile devices with a turnstile, and integration of Auth4App with FIDO2. To ensure the robustness of the security protocols, Auth4App is evaluated using automated verification tools and argument proofs, solidifying the system's reliability.

Keywords: Authentication Protocol, One-Time Authorization Code, Binding Protocol, Mobile Application

1. Introduction

Nowadays, the necessity to establish the identity of individuals in a manner that is both user-friendly and universally accessible has gained paramount importance. This requirement stems from the important role identity verification plays in ensuring seamless interactions across different areas. One particularly compelling solution that addresses this challenge revolves around using smartphones in the process. These devices have transformed into multi-functional tools that extend beyond mere communication. Employing smartphones for identity verification offers a

level of convenience that was not possible before. This approach capitalizes on the widespread familiarity people have with their smartphones, making the process of identification inherently intuitive and accessible.

Mobile-based identification not only streamlines user experiences but also introduces opportunities for innovative services that can be seamlessly integrated into everyday routines. For example, in an increasingly hybrid connected world, services either real, including facilities such as gyms, libraries, swimming pools, sports courts, and water parks, or virtual, such as email and streaming services, require some authentication mechanism (i.e., identification and verification), and access control (permission or denial of entry) to a particular space (physical or logical/virtual). Figure 1 illustrates these types of services that are accessed by a user who needs to authenticate with the service provider. He/She uses a device (e.g., physical card, smartphone, smartwatch, personal computer) to commu-

*Av. Tiarajú, 810, Alegrete, Brazil, 97546-550

Email addresses: vequincozes@gmail.com (Vagner Ereno Quincozes), mansilha@unipampa.edu.br (Rodrigo Brandão Mansilha), kreutz.research@gmail.com (Diego Kreutz), charles.miers@udesc.br (Charles Christian Miers), roger@imd.ufrn.br (Roger Immich)

nicate with some access control mechanism (e.g., doorman, turnstile, lock door) using some communication channel.

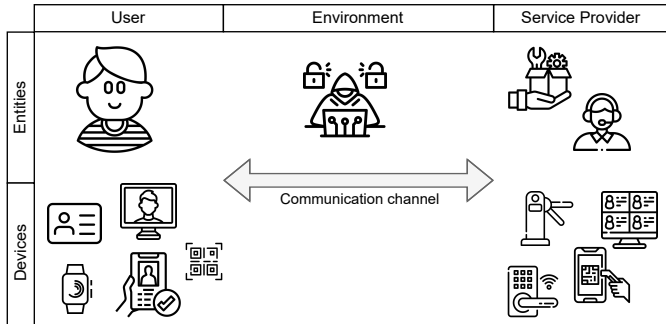


Figure 1: Overview of the target system.

This identification can be done in different ways, ranging from contactless codes (e.g., authentication code sent through Bluetooth/NFC) to virtual cards/credentials having a QR Code used for authentication purposes. However, existing virtual card-based solutions use an authentication code consisting of a static QR Code. This code is obtained only from this person’s registration number or some other identifier, such as some type of individual citizen national unique identification number, e.g., Social Security Number (SSN) in the USA.

Authentication mechanisms like this are known as static single-factor authentication. They are relatively simple to circumvent since it is enough for the malicious agent to gain access to the user’s credentials to compromise the authentication mechanism. For example, in systems using a static single-factor, the malicious agent only needs to know this person’s credentials or clone the QR Code. Moreover, a QR Code can be easily read/ copied/ cloned from a distance due to its nature. According to recent research [1], user credentials are still one of the main targets of hackers, and one of the root causes of data leakage. Furthermore, leaking user credentials is the leading cause of improper access to private data [2, 3].

It is worth noticing that high entropy unique codes (e.g., OTAC) can be considered more reliable than biometric data, which are static in nature [4]. In other words, biometric data is comparable, to some extent, to an unchangeable password. If immutable password leaks, all security on all systems based solely upon it will be compromised. Additionally, biometric readers should also be improved to provide anti-spoofing reading (e.g., silicone and Play-Doh fingerprint clones). For this reason, biometric authentication should be adopted cautiously and preferably together with other authentication methods.

In response to security vulnerabilities, numerous multi-factor and behavior/sensor-based authentication protocols have been proposed [5, 6, 7, 8, 9, 10]. Two-factor authentication typically involves a username/password as the initial factor, supplemented by an additional authentication code generated by specific applications like Google Authenticator or sent via Short Message Service (SMS)

as a second factor. However, SMS-based authentication lacks end-to-end security [11] and may pose usability challenges, similar to other multi-factor authentication mechanisms [12, 13, 14, 15, 16]. For instance, implementing an additional factor could introduce delays for users accessing facilities such as gyms or libraries, an undesirable overhead that may impede the widespread deployment of multi-factor mechanisms [6, 13]. Differently, most sensor-based authentication protocols primarily contribute to ensuring in-smartphone security, particularly by discerning whether the current user is the smartphone owner or an unauthorized attacker [10, 17, 18, 19].

Our solution, Auth4App, stands out from traditional multi-factor authentication methods. It offers a single-factor approach that balances both security and usability. Auth4App’s primary objective is to establish robust authentication using a single dynamic factor, a significant improvement over the complex and often cumbersome multi-factor methods. Auth4App is composed of two distinct protocols, each serving a specific purpose. The initial protocol is responsible for associating user credentials (i.e., identification) with the mobile device, while the subsequent protocol generates disposable one-time authentication codes (OTACs), enhancing the security of the authentication process. The fundamental concept underlying our approach is to confine the linkage between a user’s identity and a specific smartphone, thereby mitigating the potential misuse of credentials across multiple devices by different individuals. The binding protocol initiates the generation of a master key, which in turn serves as the foundation for deriving unique authentication codes through the second protocol.

It is worth emphasizing that this work is an extended version of our prior work [20], with the following improvements and main contributions:

1. developing and implementing a suite of protocols for associating users with unique devices and one-time authentication codes (OTACs).
2. enhancing usability through a comprehensive analysis of three distinct use cases for Auth4App: (i) Utilizing Auth4App for access control in electronic turnstiles (ii) Implementing Auth4App for user authentication in internet service providers (ISPs); and (iii) Integrating Auth4App with the FIDO2 authentication standard for enhanced security.
3. security analysis by performing formal verification of the protocols utilizing the Scyther tool and conceptual analysis of the system.

The subsequent sections of this paper are structured as follows: In Section 2, we delve into related works, while Section 3 covers preliminary concepts, encompassing the system model and underlying assumptions. Our proposed solution is detailed in Section 4. Section 5 presents an exploration of three case studies, and Section 6 shows an

evaluation of the solution through formal assessment and proof analysis of the system. Finally, Section 7 concludes with our final remarks.

2. Related Work

The related works are discussed following a top-down approach. In the first subsection, we summarize the works we found closest related to ours (see Table 1). In the following subsections, we support our claims by describing three systematic literature reviews to address, each one, an orthogonal aspect of the integrated solution we propose in this work.

2.1. Overview

We summarize the works closest to our solution in Table 1. In summary, none describes an integrated solution for mobile applications using OTAC and a binding mechanism to offer both mutual authentication and device revocation. Next, we detail each of the closest works. In the following subsection, describe how we came to them.

Hassan et al. [21] proposed a lightweight authentication protocol suitable for mobile and IoT applications in client-server environments. The protocol features user authentication, key agreement, and mutual authentication between the client and server, and it utilizes certificateless cryptography and elliptic curve cryptography to minimize computational resource usage. The protocol initiates a key-extraction process, where the server generates a partial private key for the client, and the client computes the complete private key and public key using its secret value. Despite its advantages, the protocol lacks a notation table, which makes it challenging to understand. Moreover, it does not employ one-time authentication codes or address user identification, meaning the user’s identity is not linked to a single mobile device.

A user authentication protocol for online mobile devices is proposed per Riaz et al. [22]. The proposal consists of two modules: client and server. The client must provide three pieces of information to the server: contact number, IMEI, and password. The server concatenates the received information, resulting in a PIN. The Least Significant Bit (LSB) technique hides the PIN before submitting it to the client. Upon receiving the PIN, the client decrypts the image and confirms the PIN to the server. The client’s IMEI is then retrieved and XORed with the PIN. The server receives the XORed value with the IMEI and verifies if the received PIN is equal to the original one generated by the server. If they match, the authentication process is successfully completed. One of the drawbacks of this work is the adoption of stenography instead of cryptography to protect the PIN. Such technique supports only confidentiality and authentication principles, while cryptography provides confidentiality, data integrity, authentication, and non-repudiation. On the other hand, the mechanism does not offer the option to revoke a user, making it vulnerable to impersonation attacks.

W. I. Khedr [23] proposes a two-factor visual authentication protocol that provides mutual authentication between the user and the server. According to the author, the mechanism resists keylogging and shoulder-surfing attacks. The protocol defines assumptions and an attack model for the proposed scenario. One of the positive points of this work is the adoption of OTACs, which are composed of a random password generated by the server, a current date and time stamp, and a PIN code. This information is encrypted with a session key generated earlier, along the user’s password and a current date and time stamp. OTAC reuse is prevented by storing the last login time and the timestamp mechanisms. Although the protocol is secure within the attack model and assumptions considered, the authentication process is complex for users as it (1) requires the user to scan a QR Code with a terminal’s webcam, (2) requires the user to verify that the PIN received on the smartphone matches the one displayed on the terminal, and (3) requires multiple entities, namely the user, smartphone, terminal, web server, and database server.

Neto et al. [24] propose a solution called AoT. This solution is based on a set of protocols that provide authentication and access control throughout the life cycle of IoT devices, i.e., during pre-deployment, ordering, development, operation, and retirement. AoT adopts Identity-Based Cryptography (IBC) to distribute keys and authenticate devices and Attribute-Based Cryptography (ABC) to apply the Attribute-Based Access Control (ABAC) cryptographically. The solution provides auxiliary protocols for: (1) key agreement to generate unique sessions; (2) key issuance, which issues a private key for device D in server S, domain Z, and cryptosystem Y; (3) binding, which binds a device D to a user U in the cloud domain; and (4) unbinding, which is analogous to the binding process, except that it unbinds the device D and the user U. In (3), device D employs digital signatures to authenticate itself to server C, while the communication from cloud server C to device D is authenticated using MACs. Despite the authors presenting a robust work, they use Message Authentication Code (MAC) instead of One-Time Authentication Codes (OTACs).

2.2. Systematic Literature Review

A key challenge of our research problem is to develop an integrated solution for orthogonal security aspects. We were able to find several solutions for either app-based authentication (e.g., [21, 22]), OTAC-based authentication (e.g., [23]), or binding protocol (e.g., [24]). We performed three independent systematic literature reviews, one for each aspect, to support this claim. To support this claim, we performed three independent systematic literature reviews, one for each aspect. Below, we describe the standard methodology we employed on them, and next, we show the summary of each of their most prominent results. The full results can be found on a public repository¹.

¹github.com/vagnerereno/systematic_review_jisa

Table 1: Comparison of the identified related works.

Ref.	Design for	App based Auth.	OTAC based Auth.	Binding protocol	Mutual Auth.	Device Revogation	Systematic Review
[21]	Mobile Environment	✓	×	×	×	×	Sec. 2.2.1
[22]	Online Smartphone's	✓	×	×	×	×	Sec. 2.2.1
[23]	Keylogging and Shoulder-surfing	×	✓	×	✓	×	Sec. 2.2.2
[24]	IoT Device Life-Cycle	×	×	✓	×	✓	Sec. 2.2.3
Auth4App	Mobile Devices	✓	✓	✓	✓	✓	*

Our systematic review follows the methodology presented in [25]. Briefly, the revision comprises four phases: (i) Planning Review, (ii) Planning Evaluation, (iii) Execution Review, and (iv) Results in Analysis. In the first stage, we defined the objectives of the systematic review. We also formulated the research questions and search strings by considering keyword synonyms. Next, we choose the location (i.e., data source) to perform the searches. We also defined the criteria for selecting the studies found. In the second phase, we evaluate the desired planning. Therefore, we ran the protocol to verify whether the results obtained were adequate in quality and viability. Soon after verifying that the study was adequate, we went to the third stage and executed the criteria defined in the previous phases. Finally, we summarize and analyze the results in the last step.

We use Figure 2 to exemplify the results of the review. The first two columns illustrate the chosen data source and search string. The remaining columns show the defined exclusion criteria (EC) (in orange) and the number of articles selected (in light green) after applying the EC. It is important to note that each objective (e.g., OTAC-based authentication, binding protocols) has its criteria (e.g., search string).

2.2.1. App based Authentication

At first, we conducted a literature review aiming to find works that propose authentication protocols based on smartphone applications. To do so, we developed a search string containing the following keywords: ((“Authetication Protocol”) AND (“Smartphone Apps” OR “Smartphone Applications”)). We chose to investigate works included in the Google Scholar² database, considered the world’s largest search engine in 2018 [26].

We defined inclusion and exclusion criteria to select the most relevant works and illustrated the work selection process in Figure 2. We thoroughly studied the 16 works in the final stage. As a result, we selected works [21] and [22] as the most related.

2.2.2. OTAC based Authentication

In a second stage, we reviewed the literature to search for works that propose authentication mechanisms based

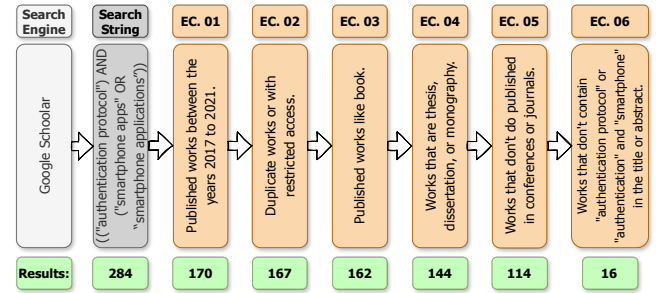


Figure 2: Systematic Review about App based Authentication

on OTAC. To achieve our goal, we defined the following search string: ((“Authentication Protocol”) AND (“OTAC” OR “One Time Authentication Code”)).

The search process, libraries, and selection criteria adopted are illustrated in Figure 3. We thoroughly studied the 15 works in the final stage. As a result, we selected work [23] as the most related.

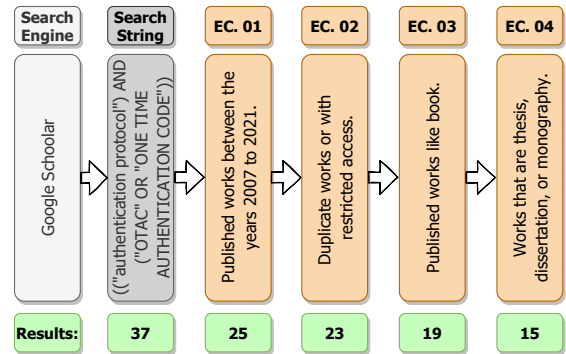


Figure 3: Systematic Review about OTAC based Authentication

2.2.3. Systematic Literature Review on binding protocol

In the third and last step, we searched for works proposing binding and authentication protocols for smartphones. To this end, we defined the following search string: ((“binding protocol”) AND (“smartphone”) AND (“authentication”)). We thoroughly studied the 22 works in the final stage. As a result, we selected the work [24] as the most related.

²<https://scholar.google.com.br/>

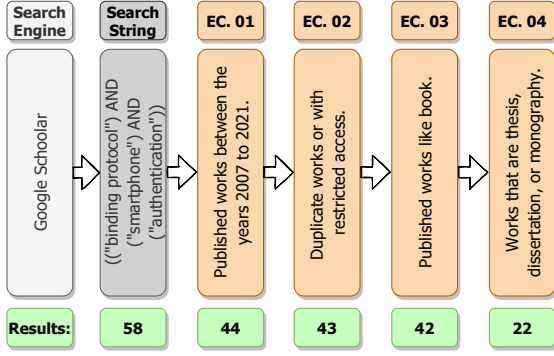


Figure 4: Systematic Review about Binding Protocol

3. Preliminaries

In this section we introduce the system model, requirements, assumptions, threat model and notations.

3.1. System model

We list entities of our system below and their relations in Figure 5. In summary, a user binds one of his personal devices (e.g., smartphone) with a service (e.g., library) offered by some provider (e.g., university). Then, the user can use that device to gain access to that service through an organization’s device (e.g., turnstile).

- **User.** An instance of entities that need to authenticate themselves, such as customers, employees, members, and third partners. Each user has associated a device, which is electronic equipment that users use to perform authentication, such as smartphones or smartwatches.
- **Service Provider.** Companies, institutions (government or not), clubs, etc., that authenticate users before providing access to a physical or digital service, such as libraries, gyms, and parking.
- **Access Control Device.** Service providers use electronic equipment, such as smartphones or turnstiles, to authenticate a user.
- **Communication Channel.** One physical transmission medium, or a logical connection running on top of multiple transmission mediums. Each communication channel offers additional verification capabilities to increase security.
- **Negotiation/Model Channel.** We consider that several users can trade with different organizations (n-n), without limitations.

3.2. Requirements

Entities must address the following requirements.

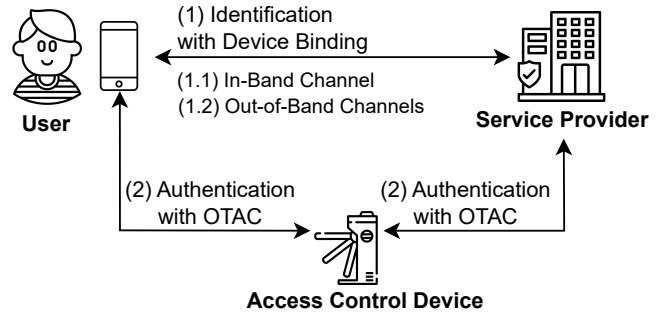


Figure 5: System Model.

- **Unique bond.** At any given time ‘t’, a user is associated with only one device. This singular association significantly mitigates risks stemming from the potential misuse of leaked user credentials.
- **General-purpose solution.** The proposed solution should be versatile and applicable across various use cases and scenarios, ensuring its general-purpose nature.
- **Multiple communication channels.** This work considers in-band (i.e., primary) and at least one out-of-band (i.e., secondary) communication channel. The former means the default channel, which is also used by the other steps of the protocol, and the latter is an additional communication channel, acting like an “safety net”. To guarantee a single and secure link between a user and a device, out-of-band channels must be used to send additional security codes.
- **Revocation of compromised identities.** The user’s device is considered nontransferable. For instance, if the user’s device is sold, lost, or stolen, mechanisms should be established to identify and revoke the linkage of the device’s application.

3.3. Assumptions

We build our system model on top of the following assumptions.

- **Assumption 1.** Both the primary and secondary channels can be compromised. However, the probability of these two channels ($P1$ and $P2$) being compromised by an attacker at the same time is significantly lower ($P1 \times P2$) than compromising only one.
- **Assumption 2.** Cryptographic primitives are ideal – that is, flawless – and can be treated as black boxes. For example, assuming that Q is an ideal and flawless cryptographic primitive, it will always produce the correct output given a correct input. Therefore, it is not vulnerable to any known attacks. In other words, Q behaves like a black box, receiving an input and producing an output without any visible internal workings.

- **Assumption 3.** Cryptographic algorithms are secure, which means that without knowing the correct keys, an adversary will never be able to forge signatures or decrypt messages.
- **Assumption 4.** An attacker can eavesdrop on, intercept or manipulate communications over an established channel between legitimate entities.
- **Assumption 5.** Devices are trusted.
- **Assumption 6.** The organization’s identification service is trusted.

3.4. Threat model

We model the capabilities of an attacker following the Dolev-Yao model, where participating entities communicate with each other through an insecure channel. During communication, the adversary has full control over the network and can delete, modify, delay, and insert communication in intercepted messages between users.

Also, we model the capabilities of an attacker following the CK-adversary model, where A can transmit the information as in the Dolev-Yao model. In addition, it can also compromise secret credentials such as session keys, private keys, and session state.

- **Threat model 1.** An adversary can subtract a user device to perform impersonation attacks.
- **Threat model 2.** An adversary can gain complete control of n-1 communication channels at any given time.
- **Threat model 3.** An attacker can attempt relay-type attacks against user A, impersonating B.
- **Threat model 4.** An attacker can attempt to replay attacks against user A, using a legitimate identification of B obtained in a previous authentication.
- **Threat model 5.** The adversary can forge a user’s identity.

3.5. Notations

For the sake of readability, we show in Table 2 symbols used to describe our solution.

4. Proposed scheme

In this section we introduce the binding and authentication protocols.

Table 2: Notations.

Symbol	Meaning
$CODE_TLS$	Transport Layer Security Code
$CODE_SMS$	Short Message Curt Code
$CODE_EMAIL$	Eletronic Mail Code
$IMEI$	International Mobile Equipment Identity
$code$	Code
$nonce$	Nonce
H	Hash Function
$HMAC$	Hash-based Message Authentication Code
E	Encrypt
K	Key
app_rnd	Pseudo-random Number
srv_rnd	Pseudo-random Number
mk_rnd	Pseudo-random Number
$KT1$	First Temporary Key
$KT2$	Second Temporary Key
KM	Master Key
V_M	Verify Master Key
$OTAC$	One-time Authentication Code
iA	Application Indice
iS	Server Indice

4.1. Overview

Auth4App comprises two primary protocols: (i) Identification: This protocol binds the user’s application to a single mobile device; (ii) Authentication: This protocol manages disposable verification codes and their generation.

Figure 6 depicts a sequence diagram illustrating how a user gains access to a service using Auth4App. The scenario involves three actors: the user, the turnstile (exemplifying an access control mechanism), and the authentication service.

Initially, the user requests registration within the corporate authentication service. Subsequently, the user can seek authentication from the turnstile. This authentication process can be offline if the OTAC mechanism is integrated into the turnstile or online by utilizing the authentication service. Detailed explanations of both registration and authentication protocols are provided in subsequent sections.

4.2. Identification Protocol: Binding a User’s Device to a Organizations’ Service

By running the identification protocol, a unique device is associated with the user’s knowledge factor, such as a login/password combination. After installing the mobile application on a selected device, the user is prompted to

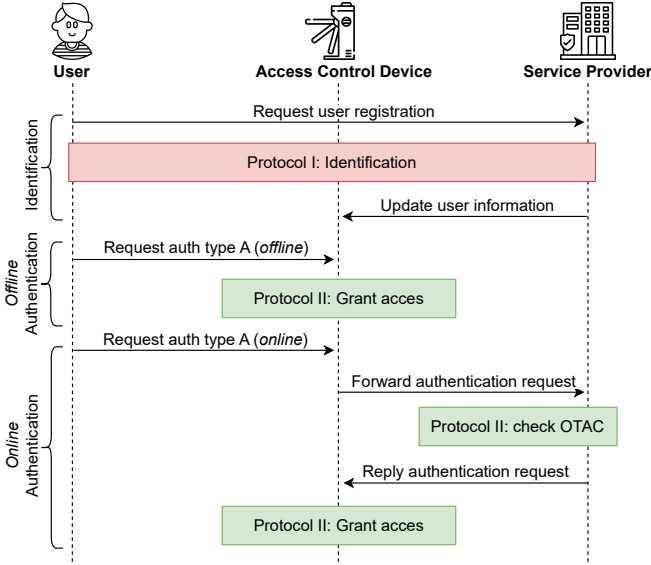


Figure 6: Auth4App: Sequence Diagram.

provide their credentials for identification and authentication (e.g., login/password) during the initial use. Subsequently, during this first access, the registration and linking protocols are initiated.

Protocol 3 specifies the identification process using SMS and email as out-of-band channels. It initiates with a TLS connection established between the application and the server (line 1). Subsequently, the server transmits three distinct codes to the client using the TLS channel (line 2) and through two out-of-band channels, namely an SMS and an email (lines 3 and 4, respectively).

One temporary key, $KT1$, is generated from the TLS session key combined with the three codes previously received from the server (line 5). This is done using a strong cryptographic hash function, as proposed and proven to be a robust alternative for generating high entropy authentication codes [27] and secret keys [28]. Subsequently, $KT1$ is used to encrypt (denoted by E) the International Mobile Equipment Identity (IMEI) and a pseudo-random number, mk_rnd (line 6). Additionally, $KT1$ is applied to sign the message (by computing the HMAC) transmitted from the client to the server.

A second temporary key, $KT2$, is computed from the values of IMEI, mk_rnd , and the key $KT1$ (line 7). The second key is presumed to be more robust due to its incorporation of a unique global identifier (the IMEI of the device) and an application-specific pseudo-random number (mk_rnd). This augmented uniqueness significantly elevates the entropy of the $KT2$ key.

After the transmission from the server to the client of a pseudo-random number srv_rnd (line 8), they finally generate a high-entropy master key KM (line 9). The master key shared by the client and the server is the main secret artifact for enabling the Authentication protocol, as discussed in the next subsection.

The last steps are dedicated to validating the master key KM . The user dispatches an encrypted pseudo-random number mk_rnd to the server (line 10). The server decrypts the received pseudo-random number using its KM key, increments it by one (+1), encrypts the incremented value, and transmits it back to the client (line 11). Successful validation of the received value by the client confirms the equality of keys, signifying the successful execution and finalization of the protocol.

It is important to recall that the user can only utilize one device at a given time. This means it is prohibited to register additional devices using the credentials (i.e., the International Mobile Equipment Identity (IMEI) and the master key) that remain valid or in use by the user. To employ a new device, the user must first revoke the current application/registration associated with the former device. This security protocol mirrors practices adopted by various fintechs and digital banks like Revolut³, N26⁴, and NuBank⁵.

4.3. Authentication Protocol: Utilizing a User's Device to gain access to a Organizations' Service

The proposed authentication protocol was designed to generate unique codes suitable for authentication purposes. As previously mentioned, the identification protocol produces a high-entropy master key, KM , which can be used to derive the key for the unique code generator. For this derivation, it is recommended to utilize robust cryptographic hash functions from reputable families such as SHA2 and SHA3⁶. The initial key for generating unique authentication codes could be as straightforward as $K_c = H(K_m || K_c)$. Initially, given that the key K_c starts empty, the first iteration of K_c is equivalent to $H(KM)$.

The key K_c is used to generate OTACs, which remain valid for a single transaction or login session. To synchronize the OTAC generation, the solution employs indexes, iA within the application and iS on the server. Initially, the OTAC equals K_c . Once generated, K_c transitions to the next value, determined by $K_c = H(K_m || K_c)$. Subsequently, OTACs are produced via the formula: $OTAC = H^N(OTAC)$, wherein the cryptographic hash function is iteratively applied N times, generating N distinct and unique authentication codes. These codes exhibit the Property of Forward Confidentiality (PFC), ensuring that previous OTAC codes cannot be deduced from the current ones. This assurance is established through the irreversibility property inherent in cryptographic hash functions.

For instance, let us assume the iA and iS indexes are temporarily set to 1 and 0, respectively. The client sends a single message to the server for authentication, including the current index of the local OTAC. When the client dispatches the message "[GET, file_name, nonce, iA],

³<https://www.revolut.com>

⁴<https://n26.com>

⁵<https://nubank.com.br/>

⁶<https://csrc.nist.gov/projects/hash-functions>

Protocol 3: Device linking and master key generation between a user (client) and a service provider (server using SMS and email as out-of-band channels).

1. Client — Server	Secure connection to the Server
2. Server → Client	[CODE_TLS, $code1$]
3. Server → Client	[CODE_SMS, $code2$]
4. Server → Client	[CODE_EMAIL, $code3$]
5. Client, Server	$KT1 \leftarrow H(K code1 code2 code3)$
6. Client → Server	[Client, $nonce$, $E_{KT1}(IMEI, app_rnd)$], $HMAC_{KT1}$
7. Client, Server	$KT2 \leftarrow H(IMEI app_rnd KT1)$
8. Server → Client	[Server, $nonce$, $E_{KT2}(srv_rnd)$], $HMAC_{KT2}$
9. Client, Server	$KM \leftarrow H(KT1 KT2 IMEI app_rnd srv_rnd)$
10. Client → Server	[Client, V_M , $nonce$, $E_{KM}(mk_rnd)$], $HMAC_{KM}$
11. Server → Client	[Server, V_M , $nonce$, $E_{KM}(mk_rnd + 1)$], $HMAC_{KM}$

HMAC”, the HMAC (message signature) is generated utilizing the application’s OTAC as the key. Upon receipt of this message, the server updates its OTAC to $OTAC = H^{iA-iS}(OTAC)$ using the index value received from the application. Subsequently, employing the new OTAC value, the server verifies the HMAC signature of the message. Finally, the server confirms successful authentication if the signatures match and denies access otherwise.

5. Study Cases

This section presents three subsections with distinct case studies, all using the Auth4App solution for authentication. In the first subsection (Subsection 5.1), a case study that uses Auth4App for access control in electronic turnstiles is presented, demonstrating its application in environments requiring security and restricted access. In the second subsection (Subsection 5.2), BKE4ISP is used as an example of how Auth4App can be applied in Internet Service Providers (ISP) for user authentication to make the authentication process more secure. In the third subsection (Subsection 5.3), FIDO2 is presented as an example of integrating other solutions with Auth4App for increased security, highlighting the solution’s versatility in dealing with different forms of authentication. As a result, these three case studies emphasize the effectiveness of Auth4App as an authentication solution for various use cases, making the authentication process more secure.

5.1. Auth4Turnstile

The use of electronic turnstiles for access control is on the rise across various facilities like gyms and universities. However, these establishments commonly rely on a static data model, such as Social Security Numbers (SSN) or a static QR Code with a pseudo-random token, for access control. This approach poses a significant security risk if a user’s credentials are compromised or leaked.

In our solution, the One-Time Authentication Code (OTAC) can facilitate access control, even in basic turnstiles capable of scanning QR Codes, as depicted in Protocol 4. Each individual within the system possesses a

digital identification card (line 1). Subsequently, the user initiates the application, automatically generating a QR Code exclusively meant for one-time use during authentication (line 2). The user then presents the QR Code to the reader (line 3), which the turnstile proceeds to scan (line 4). Following this, the turnstile updates the OTAC (line 5) and verifies it using HMAC (line 6), ultimately granting or denying user access based on the verification outcome.

Protocol 4: Electronic turnstile OTAC.

1. User	Opens the identification application
2.	QR Code = [id, iA], $HMAC_{OTAC}$
3.	Brings the QR Code closer to the Turnstile
4. Turnstile	Reads the QR Code
5.	Updates the OTAC $\leftarrow H^{iA-iS}(OTAC)$
6.	Checks HMAC using the OTAC as key

We contend that the process is efficient and straightforward due to its complete offline operation. There is no reliance on network communication or the need for time and clock synchronization between devices to generate, read, and validate QR Codes. The turnstile only necessitates a network connection for updating the user database, a task that can be scheduled. Nonetheless, in this scenario, it becomes imperative to trust the turnstile as it can potentially impersonate a user.

An alternative to the offline process involves shifting the authentication online, thereby eliminating the OTAC generator from the turnstile. In this scenario, authentication can be managed by a Trusted Third Party (TTP) or outsourced to a specialized third-party service provider, a prevalent approach utilized across various domains and scenarios [29, 30, 28]. It is noteworthy to highlight previous studies that have demonstrated the technical viability of resilient security services [31, 32], indicating that employing a TTP could be a compelling choice.

5.2. BKE4ISP

As a practical application, we consider the case of a customer requesting remote or on-site service from a third-party team through their ISP. This case illustrates several current and future challenges in handling independent providers contracted through sharing economy applications such as Uber and Airbnb.

The solution comprises three entities: Client, Technician, and Manager. These entities interact with the system primarily through a mobile application. The devices implement instances of identification and authentication protocols. Depending on the context, we adopted QR Code or the TCP/IP stack as communication channels.

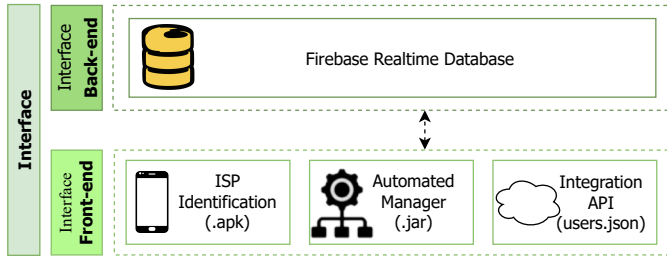


Figure 7: Implemented Interface.

Figure 7 shows the software technologies used in implementing the interfaces. These technologies can be organized into two layers: Interface Front-end and Interface Back-end. The Front-end layer includes a mobile application named ISP Identification (.apk), an Automated Manager responsible for authenticating human entities, developed in Java (.jar), and an Integration API to obtain registration data and other systems (e.g., CRM, ERP) and facilitate the user registration process. Finally, the Interface Back-end layer is responsible for maintaining user information and providing persistence for sessions. Therefore, if an entity temporarily loses its connectivity, the process will continue where it left off when connectivity is restored. We opted to use Firebase Realtime Database as the Back-end technology because it is a popular solution for storing and synchronizing data.

The three main screens of the implemented mobile application are exemplified in Figure 8. The first screen allows the user to choose between authentication functions: (i) generate authentication code and (ii) read authentication code. The second screen displays a dynamically generated QR Code to be presented. The third screen displays the identification of the other entity/entities and allows confirmation.

Authentication begins when an entity (e.g., client) clicks the generate authentication code button. The QR Code with the client’s information is generated from that moment. If any problem occurs during the protocol execution, the application displays an error message. In the normal flow, another entity (e.g., technician) must click on read authentication code and use the reader to read the QR Code presented by the client. Once the QR Code



Figure 8: Examples of the main screens of the application.

is read, the information is sent to the automated manager, which verifies both identities. If the identities are confirmed, the identification screen displayed in Figure 8(c) is presented to the client with the technician’s information (i.e., photo, name, rating, etc.). Similarly, an identification screen with the client’s information is displayed to the technician. Both entities must verify if the presented identities are correct.

In the current implementation, the Manager interface is simplified. Through this interface, the manager, in addition to authenticating users, can manage (i.e., list, edit, add, and revoke) clients and technicians. In practice, this process should be streamlined by taking advantage of some integration with third-party systems. To illustrate this scenario, the “add user” functionality implemented only asks for the user’s CPF (a Brazilian identification number) – the remaining data of interest is obtained through a query to the Firebase Realtime Database. We assume that third-party applications offer some integration API that generates, for example, a JSON file. To make the demonstration self-contained, we abstracted the third-party API by creating a JSON file with synthetic data and loading it directly into the Firebase Realtime Database. Additionally, an interface representing an automated manager was implemented. In summary, this interface runs a process that automatically verifies and authorizes all requests.

In future work, we aim to incorporate additional configurations for the manager, including customized operational protocols (such as authorization, denial, or confirmation) tailored to specific circumstances. For instance, default authorization might be set for routine technician visits to clients during regular business hours. Conversely, the initial service visit by a new technician to a client on a holiday might necessitate confirmation before access is granted.

In the implementation of Auth4ISP, all messages exchanged by the protocol are encrypted before sending and decrypted upon arrival. To do this, we adopted the Advanced Encryption Standard (AES) symmetric key algorithm, whose secret key is an OTAC. AES operates with

key sizes of 128, 192, and 256 bits and block encryption of 128 bits [33]. Both the cipher algorithm and the HMAC use the Message Digest library⁷ with the SHA-256 instance to generate cryptographic hash functions.

5.3. Integration with FIDO2

In this case study, we demonstrate how it is possible to integrate Auth4App with the Fast IDentity Online 2 (FIDO2) authentication standard⁸. FIDO2 is an open and universal standard for strong authentication, which enables user authentication without passwords through devices such as smartphones and smartcards.

To illustrate, let us imagine a scenario in which an electronic turnstile uses FIDO2 to register users and Auth4App to authenticate users, as described in Protocol 5. At the beginning of the process, the user registers their device using the FIDO2 standard (line 1). This ensures authentication security, as the device is registered uniquely. Furthermore, upon first use, the user authenticates the device using FIDO2 (line 2), which generates a credential to produce the first OTAC key (line 3).

Protocol 5: Integration Auth4App with FIDO2.

1. User	Register device with FIDO2
2.	Authenticates device with FIDO2
3.	$OTAC = ClientDataJSON$
4.	QR Code = $[id, iA]$, $HMAC_{OTAC}$
5.	Brings the QR Code closer to the Turnstile
6. Turnstile	Reads the QR Code
7.	Updates the OTAC $\leftarrow H^{iA-iS}(OTAC)$
8.	Checks HMAC using the OTAC as key

Next, upon approaching the electronic turnstile, the user presents a QR Code generated to authenticate with Auth4App (lines 4 and 5). The turnstile reads the QR Code (line 6). The system verifies if the registered device is the same one being used at the moment and if the HMACs match. If positive, the system grants access and updates the OTAC key (lines 7 and 8).

After registration and authentication with FIDO2, we use one of the available credentials to generate the first OTAC and initiate the authentication process with Auth4App, as illustrated in Figure 9. This credential is called `ClientDataJSON` and contains three properties: type, challenge, and origin. The type is a registration or authentication response. The challenge is the same as was sent by the Relying Party (RP) during the creation or acquisition ceremony. The origin contains the domain name of the terminal to which the client is connecting during registration or authentication.

⁷<https://docs.oracle.com/javase/7/docs/api/java/security/MessageDigest.html>

⁸Source code available: <https://github.com/vagnerereno/Auth4App-FIDO2>

Next, the user creates a QR Code composed of the server index (*iS*), the application index (*iA*), and an HMAC of these two concatenated pieces of information, for example, $QR\ Code = [id, iA], HMAC(OTAC)$. To generate the HMAC, we use the previously generated OTAC as the key, which is valid for a single authentication. Then, the electronic turnstile reads the QR Code to extract the received HMAC, generates a new HMAC using the OTAC as the key, and compares the two. If the HMACs match, the turnstile updates the OTAC to a hash of $iA-iS(OTAC)$.

Using this integration, we can combine the security offered by FIDO2 with the security, flexibility, resource savings, and ease of use of Auth4App. Furthermore, the FIDO2 standard is widely adopted, ensuring interoperability with other applications.

6. Evaluation

In this section, we evaluate the security of the proposed system by means of automatic verification of the proposed protocols using Scyther [34], and arguments proof regarding the overall system.

6.1. Protocols

Automated verification of a protocol’s security properties is essential to showcase its effectiveness and accuracy. For this purpose, we employ the Scyther tool [34], which takes a file containing a protocol description and security claims as input, along with a set of specified options. Notably, we rely on the “-all-attacks” option within Scyther, enabling a thorough assessment of our protocol. This includes an examination of designated roles, such as `KGC` and `MKG`, in addition to our specified security claims.

Scyther generates comprehensive reports detailing the outcomes of all executed tests and potential attacks. When the analysis uncovers vulnerabilities or weaknesses in the protocol, Scyther provides detailed flowcharts outlining possible attack vectors. Conversely, when the protocol is deemed secure, Scyther confirms this status by displaying "Ok verified" alongside "No attack" in the report comments.

Protocol 3 is presented in the semantics of the Scyther tool in the Algorithm 1. Line 1 declares the predefined types `Code`, `TemporaryKey`, and `MasterKey`. Line 2 defines a cryptographic hash function H , while line 3 introduces an HMAC function. The next three lines declare globally the keys K , $KT1$, $KT2$, and KM , respectively.

The initiation of the specification for the `Auth4App` protocol begins with the call to the `protocol` function on line 7. This specification includes four agents with explicitly defined roles: `KGC` (line 8), `MKG` (line 11), `Client` (line 22), and `Server` (line 36).

For each sending event (e.g., `send_1`, line 9), there is a corresponding receiving event (e.g., `recv_1`, line 14). The syntax of the `send_1` event signifies transmission from the Key Generation Center (KGC) to the Master Key Generation (MKG), simulating the creation of the session key

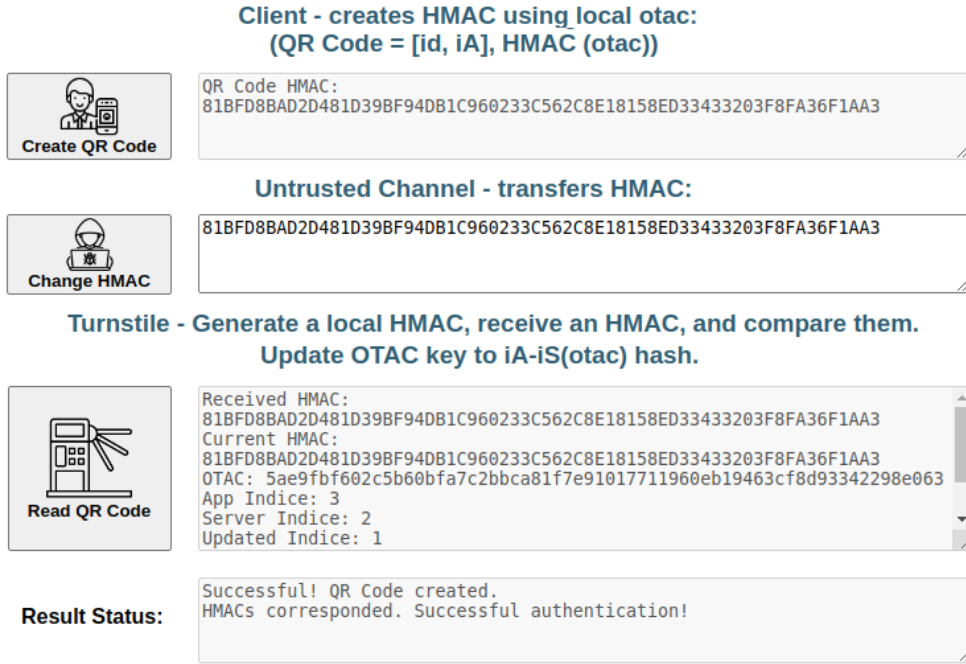


Figure 9: Auth4App integration FIDO2.

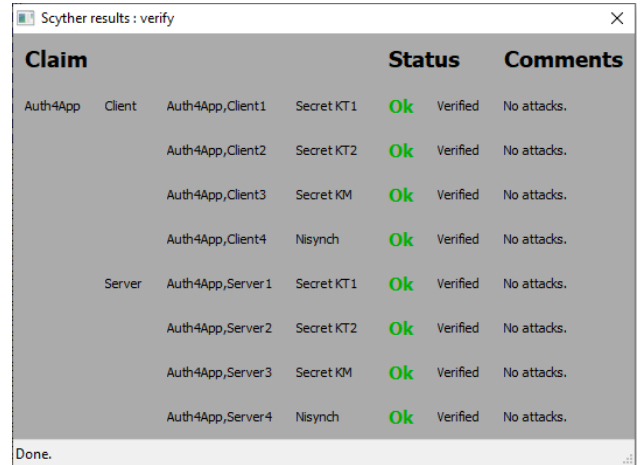
K . Subsequently, the first temporary key $KT1$ is generated using the cryptographic hash function H , utilizing as parameters the session key K alongside the codes $code1$, $code2$, and $code3$. The key $KT1$ is generated on **Client** (line 15) and **Server** (line 16), ensuring their consistency across both agents.

Finally, asserting specific security properties within the protocol specification is necessary, achieved in the Scyther tool using the `claim` function. This is used with two specific security requirements (`Secret` and `Nisynch`) to ascertain a term is secret and authentic, as can be noted from lines 31 to 34, and 46 to 49. When a term, such as the generated keys $KT1$, $KT2$, or KM , is claimed to be secret, it implies that these keys are unknown to any potential adversary. This verification ensures that these keys maintain their confidentiality and authenticity throughout communications. Moreover, the `Nisynch` requirement, specified on lines 34 and 49 for the client and server, respectively, asserts the integrity of all messages sent and received by the rightful communication partner. This claim guarantees the validity and correctness of message exchanges between the client and server.

Fig.10 showcases the output of the Scyther tool when provided with Algorithm 1 as input and utilizing the “–all-attacks” option. Notably, this status report does not exhibit any indications of failure. Based on this evidence, we confirm that the Identification protocol can be considered secure for its intended purposes.

6.1.1. Authentication Protocol

Algorithm 2 outlines the Scyther semantics for Protocol 4 (authentication employing a turnstile case study). The initial lines (1 to 4) introduce the variables used, and



Claim	Status	Comments
Auth4App Client Auth4App_Client1 Secret KT1	Ok	Verified No attacks.
Auth4App_Client2 Secret KT2	Ok	Verified No attacks.
Auth4App_Client3 Secret KM	Ok	Verified No attacks.
Auth4App_Client4 Nisynch	Ok	Verified No attacks.
Server Auth4App_Server1 Secret KT1	Ok	Verified No attacks.
Auth4App_Server2 Secret KT2	Ok	Verified No attacks.
Auth4App_Server3 Secret KM	Ok	Verified No attacks.
Auth4App_Server4 Nisynch	Ok	Verified No attacks.

Done.

Figure 10: Report: Identification Protocol Security Analysis.

the definitions for the cryptographic hash function H and the HMAC function.

The authentication process commences with the update of the user’s OTAC, which is designed for authentication purposes at the turnstile. The KGC agent takes the lead, generating a fresh code and sending it to the user (line 7). Subsequently, the user transmits their id , iA , and the HMAC message to the turnstile (line 14).

It’s essential to highlight that, owing to limitations in Scyther’s functionality, illustrating the disparity in algorithm indexes, as exemplified in Section 5.1, is not achievable. To address this limitation, an abstraction method was devised, enabling the turnstile to receive and update its OTAC code through the intervention of the KGC agent (as observed in lines 19 and 20). The claim events, as

Algorithm 1 Identification Protocol Analysis.

```

1: usertype Code, TemporaryKey, MasterKey;
2: hashfunction H;
3: const HMAC: Function;
4: secret K: SessionKey;
5: secret KT1, KT2: TemporaryKey;
6: secret KM: MasterKey;
7:
8: protocol Auth4App(KGC, MKG, Client, Server) {
9:
10:  role KGC{                                ▷ Key Generation Center
11:    send_1(KGC, MKG, H(K));
12:  }
13:
14:  role MKG{                                  ▷ Master Key Generation
15:    fresh code1, code2, code3: Code;
16:    fresh appRand1, serverRand, imei: Nonce;
17:    recv_1(KGC, MKG, H(K));
18:    send_2(MKG, Client, KT1(H(K,code1, code2,
19:    code3)));
20:    send_3(MKG, Server, KT1(H(K,code1, code2,
21:    code3)));
22:    send_5(MKG, Client, KT2(H(imei, appRand1, KT1)));
23:    send_6(MKG, Server, KT2(H(imei, appRand1, KT1)));
24:    send_8(MKG, Client, KM(H(KT1, KT2, imei,
25:    appRand1, serverRand)));
26:    send_9(MKG, Server, KM(H(KT1, KT2, imei,
27:    appRand1, serverRand)));
28:  }
29:
30:  role Client{
31:    fresh nonce: Nonce;
32:    fresh imei, appRand1, serverRand: Nonce;
33:    var code1, code2, code3: Code;
34:    recv_2(MKG, Client, KT1(H(K, code1, code2, code3)));
35:
36:    send_4(Client, Server, HMAC(nonce, {imei,
37:    appRand1}KT1(Client, Server)));
38:    recv_5(MKG, Client, KT2(H(imei, appRand1,
39:    KT1)));
40:    recv_7(Server, Client, HMAC(nonce,
41:    {serverRand}KT2(Server, Client)));
42:    recv_8(MKG, Client, KM(H(KT1, KT2, imei,
43:    appRand1, serverRand)));
44:    claim(Client, Secret, KT1);
45:    claim(Client, Secret, KT2);
46:    claim(Client, Secret, KM);
47:  }
48:
49:  role Server{
50:    var nonce:Nonce;
51:    var imei, appRand1:Nonce;
52:    var code1, code2, code3:Code;
53:    fresh serverRand: Nonce;
54:    recv_3(MKG, Server, KT1(H(K,code1, code2,
55:    code3)));
56:    recv_4(Client, Server, HMAC(nonce, {imei,
57:    appRand1}KT1(Client, Server)));
58:    recv_6(MKG, Server, KT2(H(imei, appRand1,
59:    KT1)));
60:    send_7(Server, Client, HMAC(nonce,
61:    {serverRand}KT2(Server, Client)));
62:    recv_9(MKG, Server, KM(H(KT1, KT2, imei,
63:    appRand1, serverRand)));
64:    claim(Server, Secret, KT1);
65:    claim(Server, Secret, KT2);
66:    claim(Server, Secret, KM);
67:    claim(Server, Nisynch);
68:  }

```

depicted in lines 15 and 21, are executed to ascertain the maintenance of OTAC security for both agents during Scyther's automated security analysis.

Fig. 11 illustrates the outcome of Scyther's analysis, indicating a successful evaluation. The analysis reports "Ok" for both the user and turnstile statements, confirming the security of OTAC.

Algorithm 2 Authentication Protocol Analysis.

```

1: usertype UniqueCode;
2: hashfunction H;
3: secret OTAC: UniqueCode;
4: const HMAC: Function;
5:
6: protocol OTACG(KGC, User, Turnstile){
7:
8:  role KGC{
9:    send_1(KGC, User, H(OTAC));
10:   send_3(KGC, Turnstile, H(OTAC));
11:  }
12:
13:  role User{
14:    fresh id, iA: Nonce;
15:    var nr: Nonce;
16:    recv_1(KGC, User, H(OTAC));
17:    send_2(User, Turnstile, id, iA, HMAC(id, iA));
18:    claim(User, Secret, OTAC);
19:  }
20:
21:  role Turnstile{
22:    var iA, id: Nonce;
23:    recv_2(User, Turnstile, id, iA, HMAC(id, iA));
24:    recv_3(KGC, Turnstile, H(OTAC));
25:    claim(Turnstile, Secret, OTAC);
26:  }
27: }

```

Claim	Status	Comments
OTACP User OTACP,User1 Secret OTAC	Ok	Verified No attacks.
Turnstile OTACP,Turnstile1 Secret OTAC	Ok	Verified No attacks.

Done.

Figure 11: Report: Authentication Protocol Security Analysis.

6.2. Theorem and Proofs

Next, we present theorems and proofs that illustrate how Auth4App resists impersonation attacks (Theorem 1), leverages multiple communication channels to enhance security (Theorem 2), and is robust against both relay (Theorem 3) and replay (Theorem 4) attacks. Additionally, we will demonstrate how it ensures a robust registration process to combat identity forgery (Theorem 5).

Theorem 1. With the use of Auth4App, the proposed scheme is secure against impersonation attacks. An adversary may even steal a user's device to carry out attacks.

Proof. Suppose that an adversary A has taken or stolen the device belonging to user U and is attempting to carry out an impersonation attack on some system or service. However, the act of taking the device does not guarantee that the adversary has all the necessary elements to successfully carry out the impersonation attack. Furthermore, if user U immediately reports the loss of the device, the system can revoke the device and the associated credentials. This means that the device will no longer be able to authenticate itself, preventing adversary A from using it to carry out a successful impersonation attack.

Theorem 2. AuthApp is safe against the threat model 2, as our approach incorporates multiple communication channels. Thus, if an attacker compromises $n-1$ channels, at least one channel will remain uncompromised, bolstering the overall security of the system.

Proof. For simplicity, considering that the system uses two independent channels, we can state that the probability of an attacker compromising both channels simultaneously is lower than the probability of compromising only one of the channels. For example, let’s assume that the probability of compromising the $P1$ and $P2$ channels is 50% ($P1 = 0.5$ and $P2 = 0.5$). The probability of compromising both channels simultaneously is only 25% ($P1 * P2 = 0.5 * 0.5 = 0.25$), while the probability of compromising only one of the channels is 75% ($P1 + P2 - P1 * P2 = 0.5 + 0.5 - 0.5 * 0.5 = 0.75$). Therefore, by increasing the number of channels employed, the probability of an attacker successfully compromising all of them simultaneously diminishes.

Theorem 3. Auth4App is secure against relay-type attacks, where an attacker attempts to relay messages between two parties to forge their identities.

Proof. Our proposed solution uses HMACs to ensure message integrity. HMACs rely on a secret key and a cryptographic hash function to generate summary messages that can be compared before and after transmission. This prevents tampering by an attacker attempting to relay messages between two parties. For example, if an attacker attempts to impersonate a legitimate user to access user U ’s account, they will not be successful since they cannot access the encrypted messages exchanged between the legitimate users without knowledge of the secret key. Furthermore, Auth4App implements a registration process that binds a user’s identity to a specific device, making it difficult for an attacker to impersonate another user on a different device.

Theorem 4. By using unique and disposable codes for each authentication round, the system becomes secure against replay attacks.

Proof. Auth4App utilizes the One-Time Authentication Code (OTAC) mechanism to generate unique codes for each new authentication round, rendering the system secure against replay attacks. Even if an adversary intercepts a legitimate authentication session between the user and the system, the generated code remains valid only for that specific authentication session. It cannot be reused

for a subsequent authentication. Therefore, the system can resist replay attacks, making it more resilient to unauthorized access attempts.

Theorem 5. The system is secure against attack model 5, as we have adopted a rigorous device registration process, linked the user’s identity to the device, and provided the ability to revoke the device at any time.

Proof. We have adopted a rigorous identification/binding process (see Protocol 3) that links the user’s identity to a device and enables revocation of a device at any time. Thus, the system becomes secure against impersonation attacks. These measures ensure that an adversary cannot impersonate a legitimate user by using false information to authenticate with the system. The device registration process allows the system to verify if the device attempting authentication is registered and linked to the user’s identity. In contrast, the ability to revoke a device prevents compromised devices from being used for fraudulent authentication attempts.

6.3. Security Comparison

In Table 6, we compare Auth4App with similar proposals from related work. This comparison highlights the resilience of each system against relay, replay, and impersonation attacks, as well as their use of multiple communication channels and specific resources to prevent identity forgery.

To prevent identity forgery, AoT [24] and Hassan [21] employ Identity-Based Encryption (IBE). IBE links cryptographic operations to user-specific identifiers (e.g., email, phone number), significantly reducing the risk of identity compromise. Taking a different approach, SVOSK [23] incorporates a security mechanism in which the smartphone uses device encryption to safeguard authentication information. Specifically, it stores an encrypted password (i.e., RP ID), which is essential for generating the OTAC. This encrypted password can only be decrypted using a key derived from the user’s password, which is not stored on the device. Therefore, even if the smartphone is stolen, the thief cannot access the RP ID without knowing the user’s password, thereby enhancing security even in the case of a device compromise.

It’s worth emphasizing that Auth4App stands out as the only system utilizing multiple communication channels, avoiding reliance on a single channel for its security. Indeed, past research has demonstrated the significant security enhancement achievable through multi-channel based systems [15, 35, 36]. This approach adds complexity for attackers since compromising multiple independent channels simultaneously is notably more challenging than breaching a single one. By incorporating this method into its security framework, Auth4App diversifies risk, ensuring that even if one channel is compromised, the overall integrity of the system remains protected.

In Table 6, we can observe that both AoT and SVOSK demonstrate robustness against Replay and Impersonation

Table 6: Security Resources and Attacks.

	Type	AoT [24]	OSAP [22]	SVOSK [23]	Hassan [21]	Auth4App
Attacks	Relay	×	×	×	×	✓
	Replay	✓	×	✓	×	✓
	Impersonation	✓	×	✓	×	✓
Resources	Multi-Channel	×	×	×	×	✓
	Identity Forgery	✓	×	✓	✓	✓

Table 7: Performance Evaluation.

Proposal	Registration	Authentication	Mechanism	Application	Year
AoT [24]	Not specified	4,600 ms	Based on identity and attributes for access control	IoT devices	2016
OSAP [22]	Not specified	171,000 ms	Two-step authentication with device verification and PIN	Mobile devices	2017
SVOSK [23]	21,450 ms	42,150 ms	Visual two-factor authentication protocol that ensures mutual authentication and prevents unauthorized reuse through user interaction	Mobile devices	2018
Hassan [21]	Not specified	964.29 ms	Certificateless authentication protocol based on complex elliptic curve cryptography mathematical problems	IoT devices	2019
Auth4App	88.33 ms	5.43 - 12.55 ms	Implements dynamic authentication through unique and temporal factors	Mobile devices	2024

attacks, while OSAP does not. However, OSAP offers resistance against a diverse range of other threats such as phishing, dictionary attacks, brute force, cryptanalysis, shoulder surfing, content injection attacks, and guessing strategies [22]. Similarly, SVOSK covers a variety of threats such as shoulder surfing, camera-based recordings, keylogging, Man-In-The-Middle (MITM), phishing, session hijacking, smartphone theft, and server-side attacks [23].

6.4. Performance Evaluation

In Table 7, we emphasize significant disparities in registration and authentication times across different security protocols, pivotal for evaluating the usability and efficiency of such systems. Notably, our proposal surpasses similar existing ones, including AoT [24], OSAP [22], SVOSK [23], and Hassan [21], in both registration and authentication performance.

In the case of AoT, while the total times for registration and authentication aren’t clearly defined, the focus is on the performance of cryptographic operations. For example, the generation of Attribute-Based Signatures (ABS) takes less than 1,600 milliseconds (ms), and verification takes less than 3,000 ms for predicates in the form of $A \wedge B$. Although we lack specific registration and authentication times, it’s evident that our proposal significantly outperforms AoT solely based on the expensive signature generation and verification operations it employs.

On the other hand, the SVOSK proposal specifies a registration time of approximately 21,450 milliseconds and an average authentication time of 42,150 milliseconds. Although it utilizes a secure virtual keyboard to enhance security, this introduces a significant delay, accounting for approximately 76.87% of the total time, which can potentially impact the user experience.

While OSAP is faster than traditional systems like WhatsApp, it still requires approximately three minutes

for authentication—a significant duration compared to the Auth4App proposal. Auth4App stands out for its remarkable speed, requiring only 88.33 milliseconds for registration and between 5.43 and 12.55 milliseconds for authentication. The evolution of Auth4App’s performance is significant, demonstrating an authentication system that is robust, as confirmed by security evaluations, and highly efficient. This makes it ideal for environments that necessitate rapid and dependable access processing, without compromising user experience.

The significant disparities among the proposals seem to be from the cryptographic protocols and primitives employed in their design and implementation. For instance, Attribute-Based Signatures and Public Key Cryptography are notably more resource-intensive compared to symmetric cryptography and HMAC primitives.

7. Conclusion

The demand for resilient identification and authentication solutions is rapidly growing in contemporary times. Auth4App offers a protocol suite to facilitate identification and authentication through smartphone applications. Central to this system are two fundamental protocols: one dedicated to associating user credentials with the device and another focused on generating distinctive codes.

At the core of Auth4App lies the principle of One-Time Authentication Codes (OTAC), representing robust and disposable codes generated for authentication purposes. Each OTAC is exclusive to a single authentication instance and is subsequently discarded, rendering it considerably challenging for malicious users to duplicate the generated codes. Auth4App showcases its usability through three case studies, demonstrating its potential as a comprehensive standalone solution or as an integrated component within other authentication systems. Moreover, the security of Auth4App protocols can be deemed robust, as

essential security properties have undergone formal verification using the Scyther tool and theorem-proof analysis.

Potential future directions for this research may encompass exploring hardware-assisted solutions like Trusted Execution Environments (TEEs) [37] to bolster secure key storage. Additionally, the utilization of symbolic analysis tools could be considered to formally validate the protocols.

Acknowledgements

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

Conflict of interest

The authors declare that they have no conflict of interest.

References

- [1] G. Belani, 5 Cybersecurity Threats to Be Aware of in 2020 | IEEE Computer Society, library Catalog: www.computer.org (2020).
URL <https://www.computer.org/publications/tech-news/trends/5-cybersecurity-threats-to-be-aware-of-in-2020/>
- [2] R. S. Verma, B. R. Chandavarkar, P. Nazareth, Mitigation of hard-coded credentials related attacks using QR code and secured web service for IoT, in: 10th International Conference on Computing, Communication and Networking Technologies, 2019, pp. 1–5. doi:10.1109/ICCCNT45670.2019.8944592.
- [3] InfoArmor, Understanding the impact of compromised credentials (2017).
URL https://www.infosecurityeurope.com/___novadocuments/384964?v=636398853907630000
- [4] Z. Rui, Z. Yan, A survey on biometric authentication: Toward secure and privacy-preserving identification, IEEE Access 7 (2019) 5994–6009.
- [5] A. Mahfouz, T. M. Mahmoud, A. S. Eldin, A survey on behavioral biometric authentication on smartphones, Journal of information security and applications 37 (2017) 28–37.
- [6] M. A. Ferrag, L. A. Maglaras, A. Derhab, A. V. Vasilakos, S. Rallis, H. Janicke, Authentication schemes for smart mobile devices: Threat models, countermeasures, and open research issues, CoRR abs/1803.10281 (2018). arXiv:1803.10281.
- [7] M. Abuhamad, A. Abusnaina, D. Nyang, D. Mohaisen, Sensor-based continuous authentication of smartphones' users using behavioral biometrics: A contemporary survey, IEEE Internet of Things Journal 8 (1) (2020) 65–84.
- [8] A. H. Y. Mohammed, R. A. Dziyauddin, L. A. Latiff, Current multi-factor of authentication: Approaches, requirements, attacks and challenges, International Journal of Advanced Computer Science and Applications 14 (1) (2023).
- [9] H. R. M. H. Hamid, N. W. Nordin, N. Y. Abdullah, W. H. W. Ismail, D. Abdullah, Two Factor Authentication: Voice Biometric and Token-Based Authentication, Springer Nature Switzerland, 2024, Ch. 1, pp. 27–35. doi:10.1007/978-3-031-47727-0_4.
URL https://doi.org/10.1007/978-3-031-47727-0_4
- [10] M. L. Shuwandy, A. Jouda, M. Ahmed, M. M. Salih, Z. Al-qaysi, A. Alamoodi, S. Garfan, O. Albahri, B. Zaidan, A. Albahri, Sensor-based authentication in smartphone: A systematic review, Journal of Engineering Research (2024). doi:https://doi.org/10.1016/j.jer.2024.02.003.
URL <https://www.sciencedirect.com/science/article/pii/S2307187724000300>
- [11] I. I. Androulidakis, SMS Security Issues, Springer International Publishing, Cham, 2016, Ch. 1, pp. 71–86. doi:10.1007/978-3-319-29742-2_5.
- [12] E. D. Cristofaro, H. Du, J. Freudiger, G. Norcie, Two-factor or not two-factor? A comparative usability study of two-factor authentication, CoRR abs/1309.5344 (2013). arXiv:1309.5344.
- [13] K. Marky, K. Ragozin, G. Chernyshov, A. Matvienko, M. Schmitz, M. Mühlhäuser, C. Eghtebas, K. Kunze, “nah, it's just annoying!” a deep dive into user perceptions of two-factor authentication, ACM Trans. Comput.-Hum. Interact. 29 (5) (oct 2022). doi:10.1145/3503514.
URL <https://doi.org/10.1145/3503514>
- [14] S. Das, B. Wang, Z. Tingle, L. J. Camp, Evaluating user perception of multi-factor authentication: A systematic review, CoRR abs/1908.05901 (2019). arXiv:1908.05901.
URL <http://arxiv.org/abs/1908.05901>
- [15] A. Ometov, S. Bezzateev, N. Mäkitalo, S. Andreev, T. Mikkonen, Y. Koucheryavy, Multi-Factor Authentication: A Survey, Cryptography 2 (1) (2018). doi:10.3390/cryptography2010001.
URL <https://www.mdpi.com/2410-387X/2/1/1>
- [16] A. G. D. Benedetto, WhatsApp turns on passwordless logins with passkeys for android users, <https://www.theverge.com/2023/10/16/23919609/whatsapp-passkey-android-update-passwordless-security> (2023).
- [17] A. Mahfouz, A. Hamdy, M. A. Eldin, T. M. Mahmoud, B2auth: A contextual fine-grained behavioral biometric authentication framework for real-world deployment, Pervasive and Mobile Computing 99 (2024) 101888. doi:https://doi.org/10.1016/j.pmcj.2024.101888.
URL <https://www.sciencedirect.com/science/article/pii/S1574119224000142>
- [18] N. Cariello, S. Levine, G. Zhou, B. Hoplight, P. Gasti, K. S. Balagani, SMARTCOPE: Smartphone change of possession evaluation for continuous authentication, Pervasive and Mobile Computing 97 (2024) 101873. doi:https://doi.org/10.1016/j.pmcj.2023.101873.
URL <https://www.sciencedirect.com/science/article/pii/S1574119223001311>
- [19] S. Vhaduri, W. Cheung, S. V. Dibbo, Bag of on-phone ANNs to secure iot objects using wearable and smartphone biometrics, IEEE Transactions on Dependable and Secure Computing (2023) 1–12doi:10.1109/TDSC.2023.3269037.
- [20] D. Kreutz, R. Fernandes, G. Paz, T. Jenuario, R. Mansilha, R. Immich, C. Miers, Auth4app: Protocols for identification and authentication using mobile applications, in: XX Brazilian Symposium on Information and Computational Systems Security, SBC, Porto Alegre, RS, Brasil, 2020, pp. 422–435. doi:10.5753/sbseg.2020.19254.
URL <https://sol.sbc.org.br/index.php/sbseg/article/view/19254>
- [21] A. Hassan, R. Hamza, V. G. Mawutor, A. S. Patil, F. Li, A lightweight certificateless user authentication scheme for mobile environment, in: International Conference on Machine Learning for Cyber Security, Springer, 2019, pp. 112–122.
- [22] R. Riaz, S. S. Rizvi, E. Mushtaq, S. Shokat, Osap: Online smartphone's user authentication protocol, International Journal of Computer Science and Network Security (IJCSNS) 17 (3) (2017) 7.
- [23] W. I. Khedr, Improved keylogging and shoulder-surfing resistant visual two-factor authentication protocol, Journal of Information Security and Applications 39 (2018) 41–57.
- [24] A. L. M. Neto, A. L. Souza, I. Cunha, M. Nogueira, I. O. Nunes, L. Cotta, N. Gentile, A. A. Loureiro, D. F. Aranha, H. K. Patil, et al., Aot: Authentication and access control for the entire iot device life-cycle, in: Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems CD-ROM, 2016, pp. 1–15.
- [25] J. Bionchini, P. G. Mian, A. C. C. Natali, G. H. Travassos, Systematic Review in Software Engineering, System engineering and computer science department COPPE/UFRJ, Technical

- Report ES 679 (05) (2005) 45.
- [26] M. Gusenbauer, Google scholar to overshadow them all? comparing the sizes of 12 academic search engines and bibliographic databases, *Scientometrics* 118 (1) (2019) 177–214.
- [27] D. Kreutz, J. Yu, P. Esteves-Veríssimo, C. Magalhães, F. M. V. Ramos, The KISS principle in software-defined networking: A framework for secure communications, *IEEE Security & Privacy* 16 (5) (2018) 60–70.
- [28] D. Kreutz, J. Yu, F. M. V. Ramos, P. Esteves-Verissimo, ANCHOR: Logically centralized security for software-defined networks, *ACM Transactions on Privacy and Security* 22 (2) (2019) 8:1–8:36. doi:10.1145/3301305. URL <http://doi.acm.org/10.1145/3301305>
- [29] M. Aloqaily, B. Kantarci, H. T. Mouftah, Trusted third party for service management in vehicular clouds, in: 13th International Wireless Comm. and Mobile Computing Conference, 2017, pp. 928–933.
- [30] J. Zhan, X. Fan, L. Cai, Y. Gao, J. Zhuang, TPTVer: A trusted third party based trusted verifier for multi-layered outsourced big data system in cloud environment, *China Communications* 15 (2) (2018) 122–137.
- [31] D. Kreutz, A. Bessani, E. Feitosa, H. Cunha, Towards secure and dependable authentication and authorization infrastructures, in: IEEE 20th Pacific Rim International Symposium on Dependable Computing, 2014, pp. 43–52. doi:10.1109/PRDC.2014.14. URL <https://ieeexplore.ieee.org/abstract/document/6974750>
- [32] D. Kreutz, O. Malichevskyy, E. Feitosa, H. Cunha, R. [da Rosa Righi], D. D. [de Macedo], A cyber-resilient architecture for critical security services, *Journal of Network and Computer Applications* 63 (2016) 173 – 189. doi:<https://doi.org/10.1016/j.jnca.2015.09.014>. URL <http://www.sciencedirect.com/science/article/pii/S1084804516000539>
- [33] J. Daemen, V. Rijmen, *The design of Rijndael*, Vol. 2, Springer, 2002.
- [34] C. J. F. Cremers, *Scyther: Semantics and verification of security protocols*, Eindhoven University of Technology Eindhoven, 2006.
- [35] F. Sinigaglia, R. Carbone, G. Costa, N. Zannone, A survey on multi-factor authentication for online banking in the wild, *Computers & Security* 95 (2020) 101745.
- [36] S. P. Otta, S. Panda, M. Gupta, C. Hota, A systematic survey of multi-factor authentication for cloud infrastructure, *Future Internet* 15 (4) (2023) 146.
- [37] M. Schneider, R. J. Masti, S. Shinde, S. Capkun, R. Perez, SoK: Hardware-supported Trusted Execution Environments, arXiv preprint arXiv:2205.12742 (2022).