Software and Systems Modeling
https://doi.org/10.1007/s10270-026-01375-9

SPECIAL SECTION PAPER

Formal modelling and verifying eIDAS multi-factor authentication with interface-based threat analysis

Matteo Paier[1,2] · **Roberto L.G. Van Eeden**[1] · **Marino Miculan**[1]

Received: 28 May 2025 / Revised: 30 December 2025 / Accepted: 6 March 2026

Abstract

This paper introduces a methodology for the formal modelling and verification of multi-factor authentication (MFA) schemes utilized in eIDAS digital identity cards. Our approach employs an *interface-based threat model* to systematically analyse potential vulnerabilities and enumerate a range of threat scenarios based on varying attacker capabilities. We demonstrate the automated generation of ProVerif models for these scenarios using the Italian *Carta di Identità Elettronica* (CIE), an eIDAS-compliant digital identity card, as a practical case study. Our analysis reveals several security weaknesses; notably, an attacker possessing only Level 1 (i.e. single-factor) credentials can, in certain circumstances, achieve Level 2 multi-factor authentication without needing to compromise any communication interface. To mitigate these vulnerabilities, we propose minor modifications to the protocols. Furthermore, at Level 3, our analysis shows that the authentication scheme relying on the CieID smartphone application presents a broader attack surface compared to the method employing a PC with a smart card reader. The interface-based modelling and analysis methodology presented in this work offers a valuable framework that can be adapted for the security assessment of other eIDAS digital identity cards.

Keywords Formal modelling, specification, and design · Multi-factor authentication · Security, privacy, and trust · Threat models

1 Introduction

A robust digital identity (eID) infrastructure forms a cornerstone of successful digital transformation in today's interconnected world. Recognizing this imperative, the European Commission introduced the *eIDAS* regulation in June 2014 [12], setting forth unified standards for eIDs, digital certificates, single sign-on (SSO) and trust services throughout the European Union. Despite this foundational framework, the widespread adoption of eIDAS-compliant national digital identity cards remains limited, primarily due to the inherent complexities of implementing such a nationwide service [15, 23, 33]. A significant difficulty lies in the design and implementation of secure multi-factor authentication (MFA) schemes, which are often susceptible to errors. Furthermore, past vulnerabilities discovered in SAML-based SSO systems underscore the critical need for rigorous security measures [3, 24, 27, 35]. Consequently, formal modelling and verification emerge as powerful and invaluable methodologies for the design and thorough analysis of these sensitive schemes, to improve their security and trust.

In this paper, we propose a methodology for the formal modelling and verification of multi-factor authentication schemes utilized within eIDAS digital identity cards. The analysis of these schemes is challenging due to the multitude of potential vulnerabilities, ranging from compromised passwords (the reason for MFA) to disruptions in network connectivity, human fallibility leading to phishing attacks and the compromise of user devices by malware.

To navigate this complexity, following the approach introduced in [18], we adopt an abstract perspective of an eID architecture as a collection of interacting components with clearly defined *input and output interfaces*. Subsequently, a specific *threat scenario* can be defined based on the level of access (read or write) a malicious actor can achieve on these interfaces, also simultaneously.

Communicated by Antonio Cerone, Alexander Knapp, and Alexandre Madeira.

✉ Marino Miculan
marino.miculan@uniud.it

[1] DMIF, University of Udine, Udine, Italy

[2] IMT School for Advanced Studies, Lucca, Italy

Published online: 18 May 2026

This interface-based methodology focuses on the consequences of compromise—that is, *what happens if* interfaces have been compromised—while abstracting away the specific mechanism of the attack (the *how* the control was acquired by the attacker). This abstraction allows for broad threat modelling. For instance, a malware that grants an attacker root access to an actor (e.g. the identity provider) can be seen as a scenario where the attacker fully controls all interfaces of that actor; a side-channel attack which allows to capture plaintext data flowing from the PC to the smart card is modelled as the attacker acquiring read-only access to the PC's output interface connected to the smart card.

This approach naturally leads to a combinatorial explosion of potential threat scenarios, depending on the attacker's capabilities. We demonstrate how to systematically reduce these scenarios by establishing a "strength ordering" between access levels. The resulting formal models can be then automatically generated through the use of dedicated scripts and rigorously analysed using the ProVerif security protocol verifier [8]. Overall, this approach enables a thorough assessment of the authentication schemes' resilience against various attack vectors, under different combinations of interface access acquired by the attacker.

As a practical example, we consider the Italian *Carta di Identità Elettronica* (CIE), the electronic identity card issued to all Italian citizens since July 2016. To accommodate varying security requirements, the CIE implements a multi-tiered authentication system. Level 1 relies solely on user credentials, i.e. username and password. Level 2 MFA presents three distinct methods: *L2SMS*, which delivers a one-time passcode (OTP) via SMS (this methods suits users without smartphones); *L2QRC*, which involves scanning a QR code using a dedicated CieID app; and *L2PSH*, which prompts direct authorization of the request within the CieID app. Level 3 MFA introduces the additional requirement of accessing the physical CIE smart card. This level offers two methods: *L3APP*, which requires the use of the CieID app on an NFC-enabled smartphone to interact with the smart card, and *L3DSK*, designed for computers equipped with a smart card reader, enabling smart card access without a mobile device.

This paper's analysis focuses on the security vulnerabilities of the CIE's Level 2 and 3 MFA mechanisms, the levels most commonly utilized in practice. Our investigation reveals several weaknesses across various attack scenarios. For instance, the L2SMS method is susceptible to compromise if an attacker can read the target's computer screen. More alarmingly, L2PSH exhibits a critical inherent flaw: an attacker can bypass Level 2 authentication using only Level 1 credentials by executing a precisely timed attack. Regarding Level 3, our findings indicate that L3APP presents a broader attack surface compared to L3DSK, rendering it vulnerable under a wider range of threat scenarios.

By analysing attack traces identified by ProVerif, we propose minor modifications to L2SMS, L2PSH and L3APP that significantly enhance their security. Further formal analysis confirms the effectiveness of these adjustments.

Summarizing, this work provides several contributions:

1. We propose a methodology for formal modelling and analysis of MFA schemes of eIDAS digital identity cards, adapting previous work [18] and using Italian CIE as a case study (Sect. 2).
2. We provide a threat model for CIE that covers vulnerabilities based on an attacker's access to interfaces of the system architecture. From this threat model, we can automatically enumerate a reduced yet comprehensive family of potential attack scenarios (Sect. 3).
3. We provide a complete formalization of CIE's Level 2 and 3 authentication schemes using both sequence diagrams (Sect. 2) and ProVerif code (Sect. 4).
4. We conduct an extensive analysis of these schemes in ProVerif, under the attack scenarios identified by the threat model. This analysis exposes specific security vulnerabilities within L2SMS, L2PSH and L3APP protocols (Sect. 5).
5. We propose targeted modifications to L2SMS, L3APP and L2PSH, effectively bringing the latter's security posture on par with L2QRC (Sect. 6).

In Sect. 7, we recall some related work, and finally, Sect. 8 summarizes our findings and proposes directions for future research.

The interface model presented here can account also for novel or yet undiscovered vulnerabilities, provided their impact results in the compromise of the interfaces defined within our framework. Also, the model is highly extensible: should new mechanisms for leaking information or injecting data be discovered—for instance, through specialized side-channel attacks—new dedicated interfaces can be seamlessly integrated into the model. Finally, the interface-based modelling and analysis methodology employed in this study can be readily applied to other eIDAS digital identity cards, providing a complement to the cross-country security analysis of eIDAS authentication schemes previously presented in [14].

To ensure reproducibility and facilitate further research, all scripts, ProVerif formal models, verification outputs and related materials are accessible in the accompanying reusable artefact [31]. A preliminary version of this work, focusing solely on Level 2 protocols, was previously published in [30].

2 CIE architecture and protocols

2.1 Architecture

As other eIDAS-compliant cards, the CIE authentication process leverages the SAML 2.0 standard to facilitate the secure exchange of authentication and authorization data across different security domains. This interaction involves both the CIE identity provider (IdP) and the various federated public administration service providers (SPs). The IdP, managed by the Italian Ministry of the Interior, serves as the primary entity responsible for generating security assertions. All communication between the IdP server and the devices involved in the login procedure is protected using TLS (Transport Layer Security), with specific exceptions documented where necessary.

CIE provides three distinct authentication levels for secure online access.

- Level 1 employs single-factor authentication, relying solely on a basic username and password combination.
- Level 2 introduces a second factor through a registered device, requiring the user to either enter a one-time passcode (OTP) received via SMS or authorize the login via the CieID application. The latter method necessitates entering a PIN specific to the app after receiving a push notification or scanning a QR code.
- Finally, Level 3 authentication also utilizes two factors but mandates access to the physical CIE smart card and the entry of its associated PIN. The CIE smart card can be accessed either by using the CieID app on an NFC-enabled smartphone or through a NFC card reader connected to a computer.

To ensure a stronger security posture, service providers generally mandate Level 2 or 3 multi-factor authentication. Level 1 is typically reserved for low-risk scenarios involving minimal access to personal information.

Following the approach established in [18], we give a model where entities like devices, computers and servers are abstracted as collections of interfaces enabling data ingress and egress, as shown in Fig. 1. Each interface can be independently and simultaneously compromised, thereby generating a multifaceted threat landscape.

The user's computer is modelled through the following interfaces:

PC Keyboard (pc-kbd) Represents user input via the keyboard;
PC Display (pc-displ) Represents the user's PC screen, only capable of data output;
PC TLS (pc-tls) Represents encrypted communication channels with remote entities like the IdP server. This interface offers simultaneous input and output functionality because compromising one direction (e.g. sending data) also compromises the other (receiving data) due to the nature of TLS.
PC SM (pc-sm) denotes bidirectional communication with the CIE smart card, employing the *secure message* protocol. This communication benefits from encryption and authentication mechanisms that operate independently of the underlying connection between the card reader and the PC (e.g. USB, Bluetooth); hence, it is necessary to model SM as a separate interface.

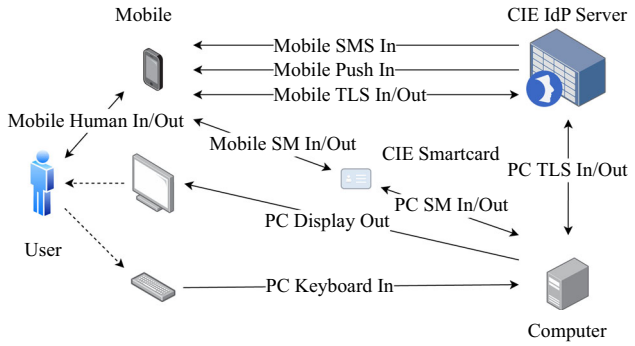

Fig. 1 CIE system interface model

Mobile devices are modelled with the following interfaces:

Mobile Human (mob-hum) Represents the mobile device's screen, encompassing both user input (e.g. through the touchscreen) and output. Unlike the separate PC keyboard and display interfaces, mob-hum covers both input and output communication channels simultaneously.
Mobile SMS (mob-sms) Represents the cellular network module used by the device to receive SMS messages from the IdP.
Mobile Push (mob-psh) Represents communication with a remote, receive-only push notification service (e.g. Firebase Cloud Messaging for Android).
Mobile TLS (mob-tls) Represents encrypted communication channels with remote entities, such as the IdP server, operating under the same assumptions as the computer's TLS interface.
Mobile SM (mob-sm) Similar to the computer's SM interface, this represents bidirectional communication with the CIE smart card using the *secure message* protocol via NFC.

The *Smart Card* is modelled with two interfaces: one for secure messaging (SM) in/out with the PC and another for SM in/out via NFC with the mobile device. Finally, the *IdP server*

is represented by its TLS interfaces for communication with both computers and mobile devices, along with supplementary interfaces Mobile SMS and Mobile Push specifically for mobile devices.

A key strength of this model lies in its ability to encompass future threats: any vulnerability that results in interface compromise is captured, regardless of the specific attack mechanism. This extensibility ensures that the framework can be easily updated to include specialized hardware interfaces, allowing for the rigorous modelling of advanced side-channel and fault-injection vectors that target sensitive data at the physical level.

2.2 Protocols

Building upon the architecture outlined above, the CIE defines three distinct protocols for Level 2 MFA and two protocols for Level 3 MFA. We now briefly recall their fundamental principles, with sequence diagrams and more detailed descriptions available in Figs. 2–6.

L2SMS This authentication scheme, utilizing an SMS OTP (Fig. 2), requires both the correct login credentials and control over the SIM card associated with the user's account.

L2PSH The push notification-based authentication (Fig. 3) necessitates knowledge of the CIE credentials and the CieID app's device 6-digit PIN, as well as possession of the device running the CieID application.

L2QRC Authentication via QR code scanning (Fig. 4) demands both possession of the registered smartphone with the CieID app and knowledge of the app's 6-digit device PIN.

L3APP This authentication scheme (Fig. 5) employs an NFC-enabled smartphone to interact with the physical CIE smart card, requiring knowledge of the smart card's PIN.

L3DSK This method (Fig. 6) utilizes a PKCS#11 smart card reader connected to a computer to unlock (via the card PIN) and communicate with the CIE smart card, eliminating the need for a mobile device.

3 Threat model

In this section, we elaborate on how an attacker can potentially compromise the interfaces of the CIE architecture, leading to a multitude of possible threat scenarios, following [18].

Each interface of the architecture in Fig. 1 may support two independent information flows: one for input and one for output. When an interface is compromised, an attacker can gain access to these flows in several ways: they might be able to passively observe and read data, actively inject and write data (allowing them to produce messages), or possess both read and write capabilities.

Our interface-based methodology centres on analysing the consequences of compromise—that is, *what happens if* interfaces are breached—by abstracting away the specific attack mechanisms used to gain control (the *how*). This abstraction allows for broad, consistent threat modelling. For example, a root-level malware infection on an actor (such as the PC or the identity provider) is represented by granting the attacker full control over all interfaces associated with that actor; the effect of a side-channel attack capturing plaintext between the PC and smart card is simply modelled as the attacker acquiring read-only (RO) access to the relevant PC output interface; a fault-injection attack which allows to alter the output of the smart card is modelled as the attacker acquiring write-only (WO) access to the PC's input interface.

Thus, the model defines an *uncompromised* system as one where the attacker possesses no read or write access to any interface. Conversely, a *fully compromised system* grants the attacker unrestricted read–write access to all interfaces.

More formally, we denote by $\mathcal{A}^{if}_{d:a}$ the level of control exercised by the attacker over an interface *if*, where:

- $d \in \{\mathsf{in}, \mathsf{out}, \mathsf{io}\}$ denotes the direction which the attacker gains control over: the input (in), the output (out) or both (io);
- $a \in \{\mathsf{RO}, \mathsf{WO}, \mathsf{RW}\}$ represents the access control acquired by the attacker: read-only (RO), write-only (WO) or read–write (RW).

Then, a *threat scenario* $\mathcal{S}$ where the attacker acquires *access control levels* $d_1{:}a_1, \ldots, d_n{:}a_n$ on interfaces $if_1, \ldots, if_n$ respectively, is represented by the set

$$\mathcal{S} = \{\mathcal{A}^{if_1}_{d_1:a_1}, \ldots, \mathcal{A}^{if_n}_{d_n:a_n}\}.$$

For instance, the set $\{\mathcal{A}^{\text{pc-displ}}_{\text{out:RW}}, \mathcal{A}^{\text{mob-hum}}_{\text{in:RO}}\}$ depicts a scenario where the attacker fully controls the computer's display (read and write) and can passively monitor the mobile device's touchscreen input (read-only). Conversely, the empty scenario {} represents the uncompromised system.

In principle, considering none, read-only (RO), write-only (WO) or read–write (RW) access on both directions of all interfaces results in a vast number of potential attack scenarios. For the CIE architecture illustrated in Fig. 1, there are 14 flows (as interfaces like keyboard and display have a single meaningful direction), leading to $4^{14} = 268\,435\,456$ possible combinations.

This figure represents the theoretical upper limit for combinatorial test cases. However, for practical evaluation, we focus our analysis on actively utilized interfaces within

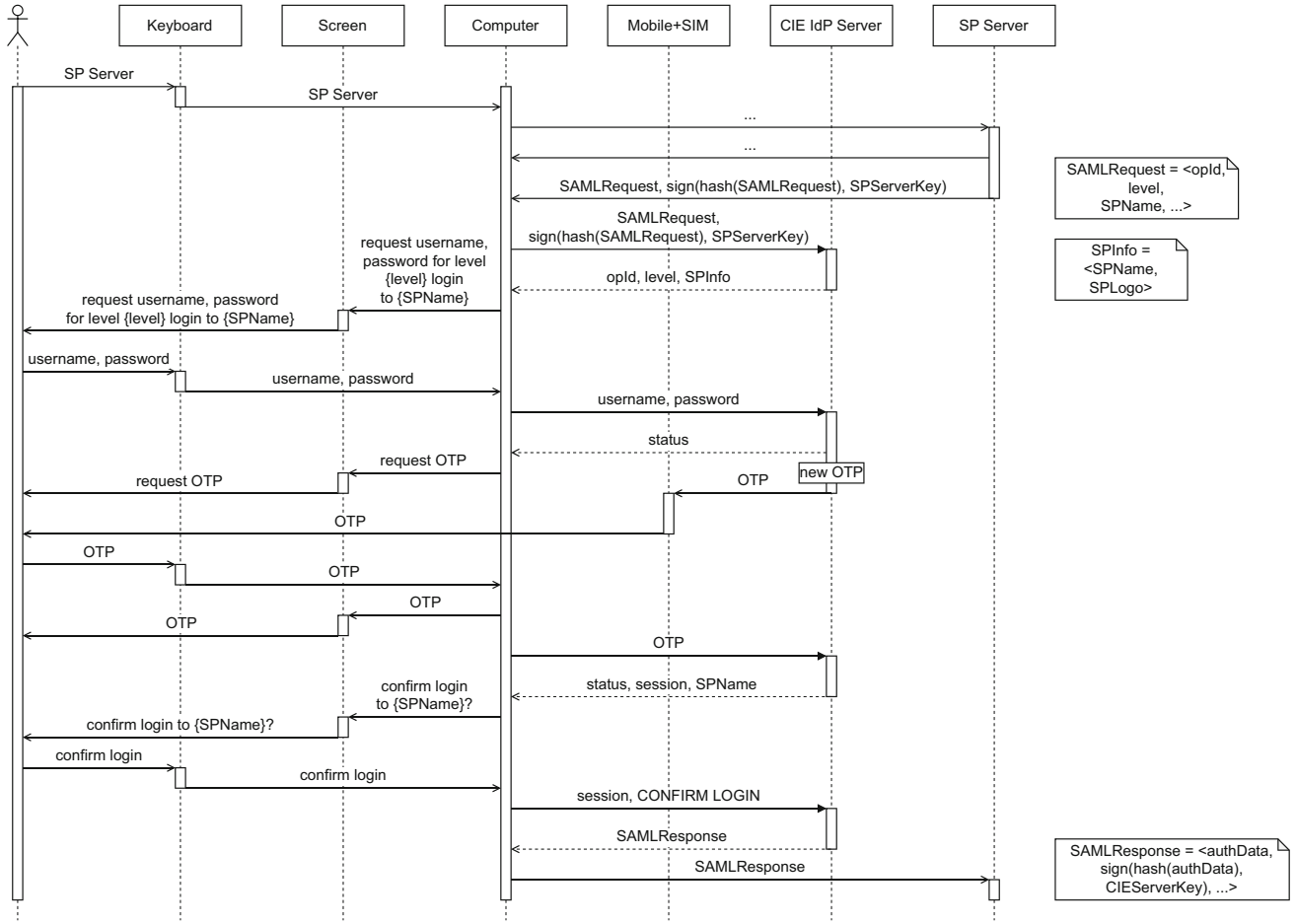

Fig. 2 Sequence diagram of the L2SMS protocol, Level 2 authentication via SMS OTP. After the user requests a connection to a specific service provider server (SP server), the IdP server prompts them to enter their Level 1 username and password. Upon successful verification, the IdP server generates a unique OTP code specifically associated with this login attempt and sends it via SMS to the registered phone number. The user then reads this OTP from their mobile device and enters it on their computer. The computer transmits the entered OTP back to the IdP server for verification. If the received OTP matches the generated one, the IdP server grants access to the account, completing the two-factor authentication process

each protocol. For instance, in the L2SMS protocol, we can exclude attacks targeting interfaces such as mobile TLS, mobile Push,and the secure messaging (SM) interfaces for both computer and mobile platforms, as these are not employed in this specific protocol. This approach reduces the search space to $4^7 = 16\,384$ scenarios for L2SMS. Analogous reasoning can be applied to the remaining four protocols.

Table 1 details the interfaces involved in each Level 2 and 3 protocol, along with the theoretical number of attack scenarios. (The final column is reserved for later discussion.) This table offers a comparative overview of the potential attack surfaces presented by the five protocols. Notably, L3DSK exhibits the smallest attack surface due to its minimal interface requirements, as it does not necessitate the use of a mobile device. Conversely, L2PSH and L3APP involve a considerably wider range of interfaces, consequently expanding their potential attack surfaces. Generally, the broader is the attack surface, the more likely an attacker can discover a successful attack path.

The data in this table reveals a total of nearly 1.4 million potential attack scenarios to consider. This still substantial number can be further reduced by leveraging two key observations.

First, gaining write access to a system interface typically demands higher privileges or the exploitation of specific platform vulnerabilities, situations that would also inherently grant the attacker read access. Consequently, write-only access attacks are not applicable to the CIE model, as acquiring write access invariably provides read access to the same interface. Therefore, we can effectively limit our analysis to three access control levels: none, RO and RW.

Secondly, acquiring control over an interface's input is generally less challenging than controlling its output. This

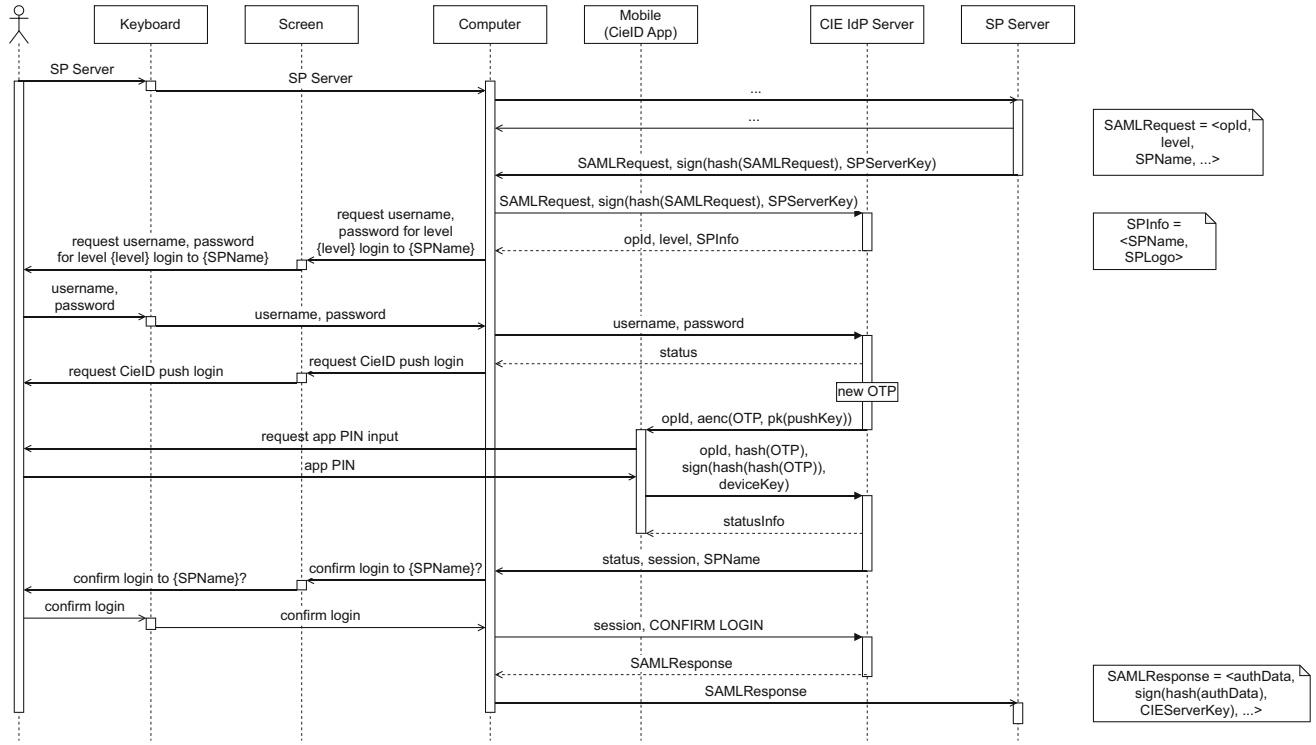

Fig. 3 Sequence diagram of the L2PSH protocol, Level 2 authentication via app push. Upon initiating the login process, the IdP server generates a unique access token. The user then authenticates using their login credentials and selects the push notification option. Subsequently, the server generates a 4-digit OTP code, encrypts it using the user's device's public key and transmits it along with the access token via a secure notification service to the registered smartphone. Decryption and user verification take place on the device, requiring the entry of the 6-digit CieID app PIN for authorization. Following successful PIN entry, the CieID app signs a hash of the OTP code and transmits it, along with the access token and the digital signature, back to the IdP server for verification. Successful validation of the signature and OTP hash by the server grants access to the requested service. It is important to note that this entire process is subject to time constraints, necessitating a timely response to maintain security

Table 1 Number of theoretical and reduced scenarios for each protocol

Protocol	pc-kbd	pc-displ	pc-tls	pc-sm	mob-hum	mob-sms	mob-psh	mob-tls	mob-sm	Theoretical number of scenarios	Reduced number of scenarios
L2SMS	✔	✔	✔		✔	✔				16 384	405
L2PSH	✔	✔	✔		✔		✔	✔		262 144	1215
L2QRC	✔	✔	✔		✔			✔		65 536	405
L3APP	✔	✔	✔		✔			✔	✔	1 048 576	1215
L3DSK	✔	✔	✔	✔						4096	81
Total										**1 396 736**	**3321**

disparity arises from the differing permission levels typically associated with each action. For example, reading data from a USB port usually only necessitates user-level code execution. In contrast, intercepting or modifying data transmitted over USB by another program typically requires administrator privileges or the exploitation of specific system vulnerabilities. Consequently, an attacker who successfully gains read–write control over an interface's output can also be assumed to have control over its input.

Based on this observation, we can establish a *strength ordering* for control access levels: read–write access to an interface's output (out:RW) is considered stronger (and more challenging for an attacker to achieve) than both read-only access to the output (out:RO) and read–write access to the input (in:RW) of that interface. Both out:RO and in:RW are

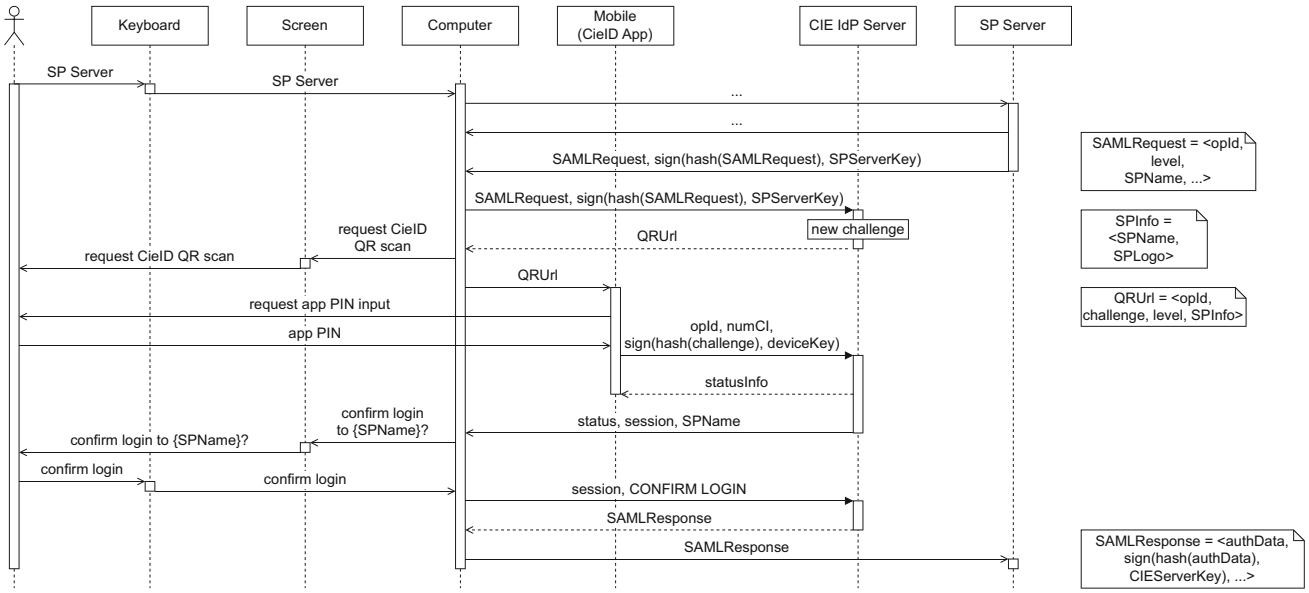

Fig. 4 Sequence diagram of the L2QRC protocol, Level 2 authentication via QR code recognition. The authentication process begins when the user initiates a login on the SP server from their computer. This action prompts the IdP server to generate a unique challenge specifically associated with this login attempt. This challenge is then displayed as a QR code, containing pertinent information such as the operation ID and the service provider's details. The user scans this QR code using the CieID application on their smartphone, enters their device PIN for authorization and confirms the login. Subsequently, the app signs a hash of the challenge using the device's private key and transmits this signed hash, along with the user's CIE card number, to the IdP server. The server then validates the digital signature using the registered device certificates. If the validation is successful, the server grants access to the user's account. To ensure security, the generated challenge has a two-minute expiration period, after which the login attempt is invalidated if not completed

stronger than read-only access (in:RO), which, in turn, is stronger than having no control access at all.

Formally, we can define a partial order $\Rightarrow$ on the set of access control levels: $\ell_1 \Rightarrow \ell_2$ means that achieving the access level ℓ_1 implies achieving also the access level ℓ_2. Two access levels ℓ_1, ℓ_2 are equivalent, $\ell_1 \Leftrightarrow \ell_2$, if $\ell_1 \Rightarrow \ell_2$ and $\ell_2 \Rightarrow \ell_1$. This order is shown in Fig. 7, where $\top$ represents no access at all.

This partial order can be extended to scenarios pointwise:

$$\mathcal{S}_1 \Rightarrow \mathcal{S}_2 \overset{\triangle}{\iff} \forall \mathcal{A}^{if}_{\ell_2} \in \mathcal{S}_2 . \exists \mathcal{A}^{if}_{\ell_1} \in \mathcal{S}_1 . \ell_1 \Rightarrow \ell_2.$$

Consequently:

- if a given protocol is proven secure under an attack scenario $\mathcal{S}$, then it is also secure under any scenario $\mathcal{S}'$ such that $\mathcal{S} \Rightarrow \mathcal{S}'$; in this case, we say *the security in $\mathcal{S}'$ is implied by a stronger scenario*;
- conversely, if a protocol is found to be vulnerable in scenario $\mathcal{S}$, then it is also vulnerable in any scenario $\mathcal{S}'$ such that $\mathcal{S}' \Rightarrow \mathcal{S}$; here, we say that *the vulnerability in $\mathcal{S}'$ is implied by a weaker scenario*.

Therefore, upon discovering an attack in a specific scenario, we can forgo the analysis of stronger scenarios. Conversely, if a protocol is verified as secure in a particular scenario, there is no need to examine weaker ones.

By leveraging these implications, we can observe that for each single direction interface (such as keyboard and display) there are three possible access control levels: none, RO and RW. On the other hand, bidirectional interfaces admit five possible access control levels. Overall, this leads to an upper bound of $3 \cdot 3 \cdot 3 \cdot 3 \cdot 5 \cdot 5 \cdot 5 \cdot 5 \cdot 5 = 253\,125$ scenarios for a protocol that would use all interfaces.

We can further reduce this search space by recognizing that for TLS channels, the only relevant access levels are io:RO and io:RW, since gaining access to one direction of a TLS channel inherently grants access to the other direction as well. Here, we utilize a simplified model that captures the core properties of TLS (confidentiality, authentication and message order) while acknowledging that attackers might be able to block or delay messages. A similar consideration applies to the SM channels between computer (or mobile) and the smart card. Overall, this allows us to reduce the upper bound to $3 \cdot 3 \cdot 3 \cdot 3 \cdot 3 \cdot 3 \cdot 3 \cdot 3 \cdot 5 = 32\,805$ scenarios, since for both SM and TLS interfaces the only direction is io.

Furthermore, it is important to note that each protocol utilizes only a specific subset of the system's available interfaces. Consequently, we can further refine our analysis by considering, for each protocol, only those attack scenarios

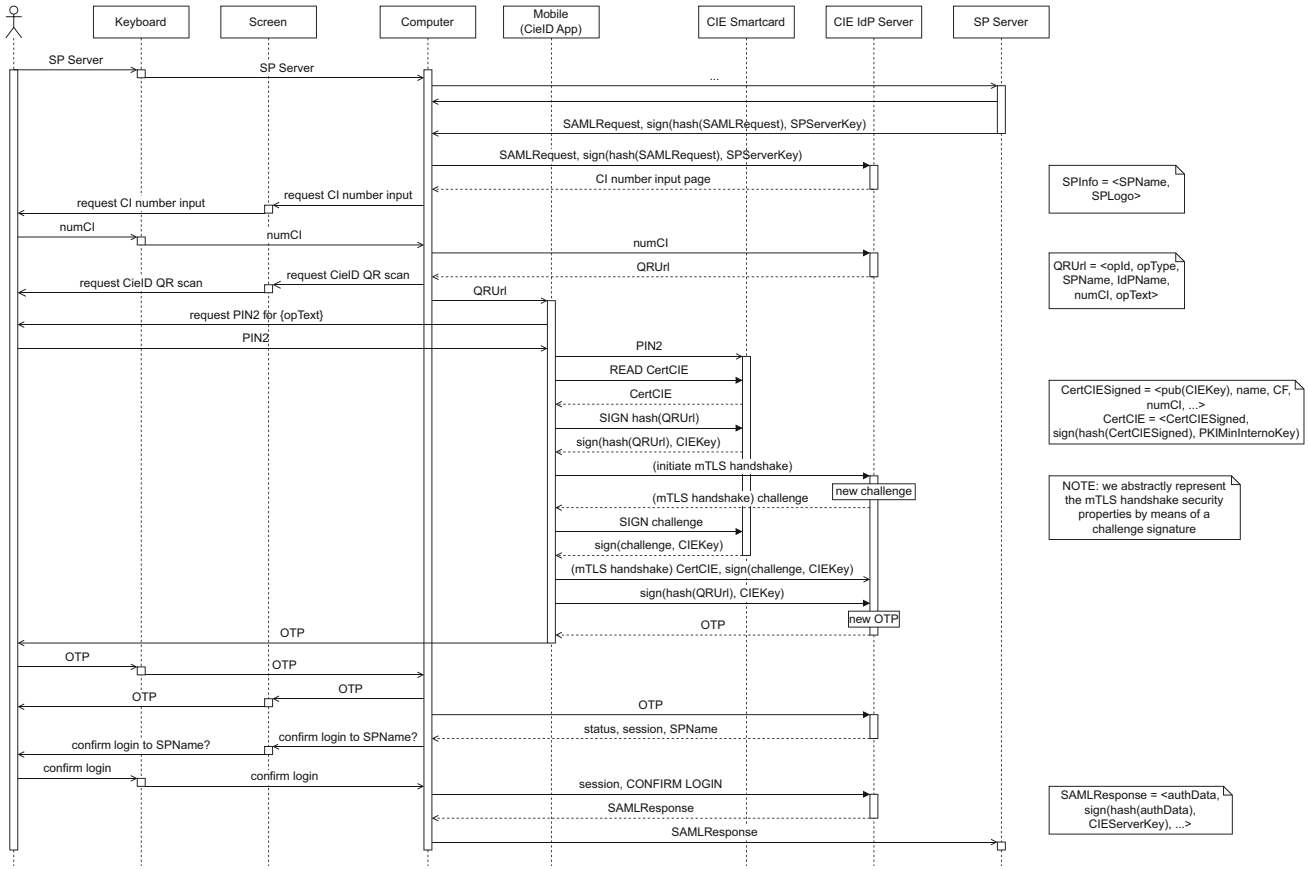

Fig. 5 Sequence diagram of the L3APP protocol, Level 3 authentication using the CIE smart card and PIN via a NFC-enabled smartphone. The IdP generates a QR code containing login details and the CIE card number, which is displayed on the user's computer screen. The user scans this QR code using the CieID application on their smartphone. The app then displays the service provider's information and prompts the user to enter the last four digits of their CIE PIN (PIN2; the first four digits, PIN1, are securely stored within the app). After the user enters PIN2 and places their smartphone near the CIE smart card for NFC reading, the app unlocks the card, reads its digital certificate and signs a cryptographic hash of the QR code's URL. Subsequently, the app initiates an HTTPS request to a fixed IdP server URL, requiring TLS client authentication using the CIE's digital certificate. The app transmits the signed QR code URL hash to the IdP server. Upon successful validation of the signature, the IdP server generates an OTP and sends it back to the CieID app. Finally, the user enters this OTP on their computer, which transmits it to the IdP server for final verification, granting access to the user's account upon successful OTP validation

that involve the interfaces actually used by that protocol. For example, in the case of L2SMS, the number of relevant scenarios reduces to $3 \cdot 3 \cdot 3 \cdot 3 \cdot 5 = 405$. Applying this same logic to the remaining protocols yields the figures presented in the last column of Table 1, resulting in a total of 3321 scenarios.

Phishing attacks. Even MFA systems are susceptible to human error and social engineering attacks; thus, it is important to evaluate the resilience of CIE's MFA mechanisms against these threats.

Phishing attacks pose a significant risk to users. In a phishing scenario, users might be deceived into granting access to fraudulent versions of legitimate services or unknowingly disclosing sensitive information such as one-time passcodes. To model this attack within our scenarios, we adopt the assumption that the adversary controls the targeted server, i.e. the server the user intends to authenticate with. Our phishing attack model is consistent with the approach taken in [18]; however, given the absence of fingerprint-related data in the CIE login process, we have excluded fingerprint bypass from our analysis.

While the CieID app attempts to mitigate phishing risks by displaying a three-second warning message prompting users to verify the displayed URL against the official CIE IdP URL, users may still overlook or disregard this warning. Furthermore, even with this protection in place, more sophisticated techniques such as IDN homograph attacks can deceive users into believing a malicious webpage is legitimate [17].

Formally, to represent a phishing attack scenario we use the symbol PH in the attack scenario notation. This leads to double the number of scenarios to analyse, for a total amount of 6642.

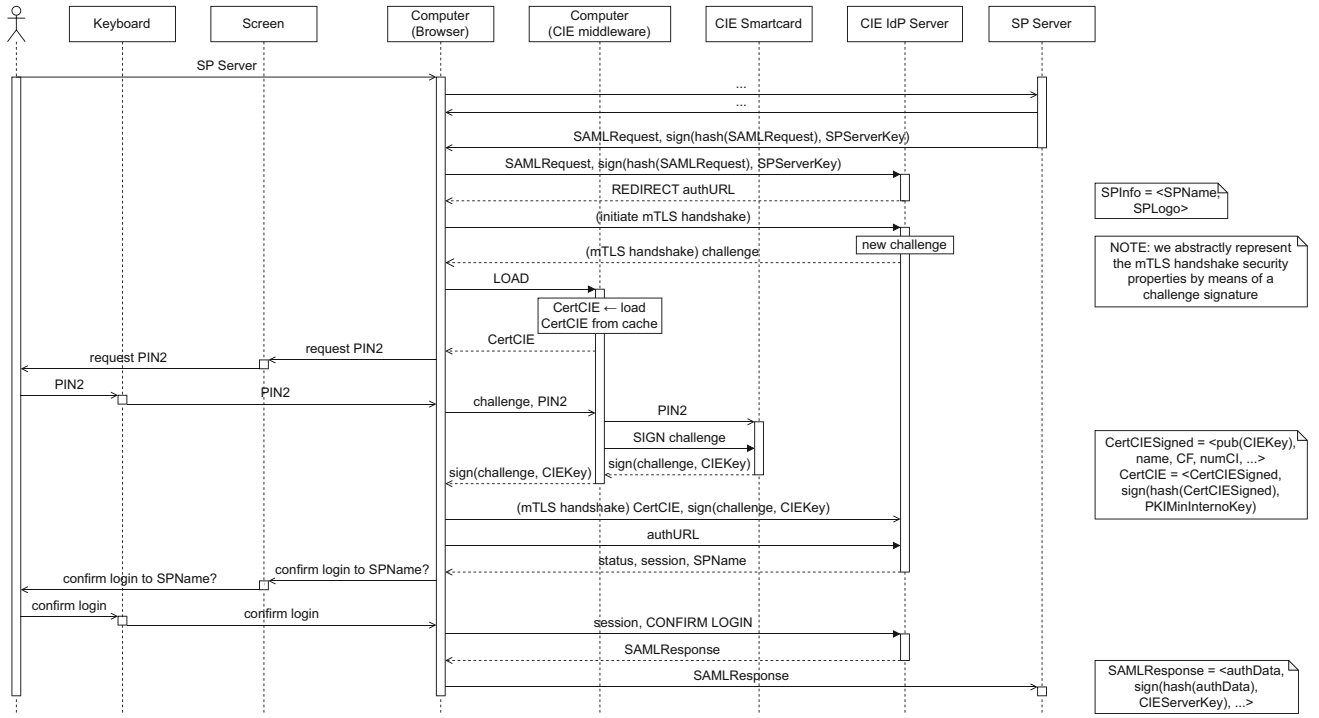

Fig. 6 Sequence diagram of the L3DSK protocol, Level 3 authentication using standard TLS client certificates via the PKCS#11 CIE middleware. Upon selecting this authentication method, the IdP redirects the user's browser to a secure HTTPS page that mandates a specific client certificate chain for access. The browser then initiates the CIE middleware, which prompts the user to enter the last four digits (PIN2) of their CIE smart card PIN. This middleware combines the entered PIN2 with the first four digits (PIN1), which are stored in encrypted form on the computer, to unlock the smart card. Subsequently, the browser completes the TLS handshake, utilizing the CIE smart card's digital certificate and a digital signature to verify ownership of the corresponding private key. Notably, during this TLS handshake the CIE smart card certificate, which contains the user's name, fiscal code (including birth information) and document ID, is transmitted in plaintext. This raises potential privacy concerns as it could reveal sensitive personal data and details of the login attempt to network eavesdroppers

4 Formal modelling CIE Level 2 and 3 MFA protocols

In this section, we discuss the formal modelling of the CIE Level 2 and 3 MFA protocols and various attack scenarios using ProVerif, a leading security protocol verifier. Our analysis evaluates the protocols' resilience against a range of scenarios involving pre-compromised user passwords, varying levels of attacker control over user devices and successful phishing attempts. By simulating these threats within our formal models, we aim to provide a comprehensive assessment of the effectiveness of CieID multi-factor approach in safeguarding user accounts. The complete formalization is publicly available at [31].

4.1 Introduction to ProVerif

We begin with a brief introduction to ProVerif, a robust tool designed for the symbolic verification of security protocols in the symbolic Dolev–Yao model; for more details, we refer to [8] and various tutorials available online.

In ProVerif, the protocol specification is written in a variant of the *applied* π*-calculus*, a process algebra used to formally describe how communicating processes exchange messages over channels. Messages and the operations performed on them are represented by abstract algebraic terms. We can define *term constructors* (e.g. for encryption, signing or other operations) along with *equations* that capture their algebraic properties (e.g. decryption undoing encryption).

Internally, ProVerif models the system by translating the protocol and the security properties into a set of Horn clauses, with the security properties specified as logical goals. A specialized resolution algorithm is then executed on this representation to search for a derivation of the negation of each goal. If no derivation is found, the security property is verified. However, if a derivation is found, ProVerif extracts a concrete attack trace and present it to the user.

4.1.1 Terms and processes

In ProVerif, the user can specify an algebraic theory of terms by declaring a *signature* Σ, i.e. a finite set of functions sym-

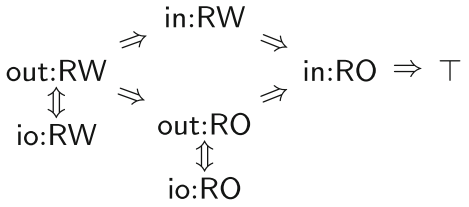

Fig. 7 Security implication between access control levels

bols and a set of algebraic laws which define how terms are reduced.

Given a signature Σ, along with a countable set of *names* and a countable set of *variables*, the set of *terms* is defined by the grammar:

$$\begin{array}{lll} M, N ::= a, b, \ldots & & \text{names} \\ \quad | x, y, \ldots & & \text{variables} \\ \quad | f(M_1, \ldots, M_n) & & \text{constructor application, with } (f : n) \in \Sigma \end{array}$$

Datatypes can be defined by suitable functions and equational laws; for instance, *pairs* are definable by means of a signature $\Sigma_p = \{pair : 2, \pi_1 : 1, \pi_2 : 1\}$ together with the reduction rules

$$\pi_1(pair(M, N)) = M \quad \text{and} \quad \pi_2(pair(M, N)) = N$$

where M, N are implicitly universally quantified. Since π_1, π_2 allow to eliminate a term constructor previously applied, they are called *destructors*.

Constructors and destructors can be used for modelling many cryptographic functions in the symbolic model; e.g. an hash function is modelled by a constructor $h : 1$ with no destructors nor reduction rules.

In particular, a *perfect symmetric encryption* scheme can be specified by $\Sigma = \{enc : 2, dec : 2\}$ together with the equation $dec(K, enc(K, M)) = M$, reflecting the fact that the only way to decrypt a message is to know the correct key: if $K \neq K'$, a term $dec(K, enc(K', M))$ cannot be reduced to M, nor can it be analysed or modified in any other way. This encryption function is called "perfect" because ProVerif has no way to extract any information from $enc(K, M)$, apart from decoding it with the correct key: it behaves as an ideal symmetric encryption scheme. Working in the symbolic model, ProVerif does not deal with computational aspects, like key or message sizes, probability distributions of messages, brute-force searches, etc.

The complete description of process specification language, including typing declarations, is provided in the ProVerif reference documentation. For convenience, a significant fragment of the process grammar—defined over a given term signature and its equations—is provided in Fig. 8.

The null process 0 does nothing (i.e. it is terminated). $\mathsf{in}(c, x)$; P receives a message M from channel c and then continues as $P[M/x]$; notice that this operation is blocking. $\mathsf{out}(c, M)$; P outputs the message M on channel c and then continues with process P. $P|Q$ is the parallel, interleaved execution of P and Q (which can interact on shared channels). The process $!P$ effectively behaves as an infinite number of copies of P running in parallel (*unbounded* replication) and $\mathsf{new}\, n$; P creates a local, new fresh value before proceeding with process P. The constructor $\mathsf{if}\ M\ \mathsf{then}\ P\ \mathsf{else}\ Q$ is a standard conditional, and $\mathsf{let}\ M = N\ \mathsf{in}\ P$ is a *pattern-matching*: the term N is matched on pattern N, and it proceeds with P only if the match succeeds, possibly instantiating variables in M. Finally, $\mathsf{event}(e(M_1, \ldots, M_n))$; P signals to ProVerif that a certain event e (with its parameters) has happened, and then proceeds with P; this is useful in the specification of security properties such as authentication or data integrity.

4.1.2 Security property analysis

As mentioned earlier, ProVerif's analysis is based on Horn clauses, that is, first-order formulas of the form

$$F_1 \wedge \ldots \wedge F_n \Rightarrow G_1 \vee \ldots \vee G_m$$

where each F_i, G_j is a literal. ProVerif provides some predefined literals; in particular, $\mathsf{attack}(M)$ will be used to denote that the attacker knows the term M. Other literals can be defined by the user; in particular, $\mathsf{event}\ e(M_1, \ldots, M_n)$ holds when a process has signalled a matching event in its execution.

ProVerif's verification core translates the entire system, including cryptographic primitives, attacker capabilities and the protocol description, into a comprehensive set of Horn clauses. Security properties are similarly expressed as queries that the tool attempts to prove. A common case is

$$\mathsf{query}\ \mathsf{attack}(M)$$

This goal aims to prove that the attacker can infer the message M (e.g. a secret key or a nonce). ProVerif attempts to deduce this fact from the underlying set of Horn clauses and the algebraic theory of terms. If the deduction succeeds, a corresponding attack trace is extracted from the clause inference. If no derivation of the query can be found, the security property is verified (i.e. M is proven secret from the attacker).

Another crucial type of query is the *correspondence assertion*, which uses symbolic events to express dependencies and ordering constraints between execution steps, even across different protocol participants. These assertions take the form of a universally quantified implication:

$$\mathsf{query}\ \mathsf{event}(e(M_1, \ldots, M_n)) \Rightarrow \mathsf{event}(e'(N_1, \ldots, N_k)).$$

Fig. 8 Syntax of ProVerif processes

$$
\begin{array}{rll}
P, Q ::= & 0 & \text{null process} \\
| & \mathsf{in}(c, x); P & \text{message input} \\
| & \mathsf{out}(c, M); P & \text{message output} \\
| & P|Q & \text{parallel composition} \\
| & !P & \text{replication} \\
| & \mathsf{new}\ n; P & \text{fresh name creation} \\
| & \mathsf{if}\ M\ \mathsf{then}\ P\ \mathsf{else}\ Q & \text{conditional} \\
| & \mathsf{let}\ M = N\ \mathsf{in}\ P & \text{local declaration with pattern matching} \\
| & \mathsf{event}(e(M)); P & \text{event}
\end{array}
$$

This statement is satisfied if the occurrence of $\mathsf{event}(e(\ldots))$ guarantees that event $\mathsf{event}(e'(\ldots))$, has already occurred. For example, a basic authenticity property can be modelled as: "For any message M, if the receiver logs $\mathsf{event}(\mathsf{MsgRcvd}(M))$, then the sender must have logged $\mathsf{event}(\mathsf{MsgSnt}(M))$ before". To verify this, ProVerif attempts to falsify the assertion, i.e. it tries to prove its negation:

$$\exists M_1, \ldots, M_n, N_1, \ldots, N_k.\mathsf{event}(e(M_1, \ldots, M_n)) \wedge \neg \mathsf{event}(e'(N_1, \ldots, N_k)).$$

If this inference succeeds, ProVerif extracts an attack trace showing a scenario where the first event happens but the second one does not, thereby demonstrating a violation of the security property.

Since the verification of security properties is an undecidable problem, ProVerif (like similar protocol verifiers) may not always yield a definitive result. Furthermore, verifying highly complex protocols may be constrained by practical time and memory limits. In practice, the tool produces one of three outcomes:

- True: The property is formally verified, and no attacks were found within the symbolic model.
- False: A vulnerability is identified, and a concrete attack trace is exhibited.
- "Cannot be proven": This inconclusive state typically occurs when the tool's over-approximation leads to a logical derivation of an attack (via Horn clause resolution) for which a valid execution trace cannot be reconstructed.

In the latter case, ProVerif provides intermediate information that assists developers in manually identifying potential vulnerabilities and refining the specification. By making non-essential changes to the model (such as reordering process operations or adding auxiliary checks), we can guide the tool's search algorithm while preserving the protocol's original semantics. This procedure ensures that we do not settle for inconclusive results but instead push the tool towards a definitive proof of security or the discovery of a valid attack trace.

In fact, extensive case studies over more than two decades have consistently demonstrated ProVerif's precision and efficiency for practical applications; see, for example, [5, 8, 21, 28] and the web page at [7] for hundreds examples.

4.2 Modelling CIE protocols in attack scenarios

Within ProVerif, participants in a protocol are modelled as parallel-executing processes that communicate by exchanging messages over *channels*. The primitive $\mathsf{out}(c, m)$ denotes the sending of message m on channel c, while $\mathsf{in}(c, x)$; P represents a process that, upon receiving a message from channel c and binding it to the variable x, proceeds with the execution of process P. Channels can be declared as either *public* or *private*. Adhering to the Dolev–Yao model, an attacker can intercept and manipulate any message transmitted over channels whose names are known to him. Consequently, even a private channel becomes accessible to the attacker if its name has been disclosed.

Each interface of the architecture in Fig. 1 is modelled as a pair of private channels: one for input and one for output. Moreover, we assume a public channel $public$, accessible to the attacker. Given a formalization P of a protocol and an attack scenario $\mathcal{S}$, we can derive a formalization $P_{\mathcal{S}}$ that incorporates new primitives representing the attacker's capabilities as defined by $\mathcal{S}$, as follows:

- To simulate an attacker gaining read-only access to an interface, we disclose all messages transmitted on the corresponding private channels. Formally, if *iface* represents a secret interface channel, then
  - each instance of $\mathsf{out}(iface, message)$ in the protocol is replaced by

    $\mathsf{out}(public, message); \mathsf{out}(iface, message);$

– each instance of in(*iface*, *message*) is replaced by

in(*iface*, *message*); out(*public*, *message*).

– Conversely, to grant the attacker read–write (i.e. complete) access to a channel, we disclose its private channel name to them, with an out(*public*, *iface*) at the beginning of the protocol.

Furthermore, modelling authenticated communication channels, specifically those secured by TLS, necessitates specific abstraction techniques.

– We exclude the details of TLS handshake protocols from this analysis (for which we refer to the dedicated study in [5]). Instead, we abstractly model a secure and authenticated TLS channel using a function `TLS(id, id) : channel`. Only authorized parties are permitted to invoke this function to obtain unique, unguessable channel names, thereby ensuring system integrity. When a TLS connection is established, one host generates a fresh session identifier, which is implicitly associated with all subsequent transmitted messages to prevent session confusion and data mixing.
– Since ProVerif's native channels are synchronous, a direct representation would preclude the simulation of critical network attacks, such as message blocking or delay by an active attacker. To enable these asynchronous network behaviours, we model each TLS channel by decoupling the communication using the π-calculus parallel composition operator.
– Finally, to accurately simulate scenarios where the attacker controls one communicating party and consistently initiates TLS sessions, we introduce an auxiliary TLS manager process. This manager is responsible for establishing TLS channel names with the compromised party and explicitly revealing these names to the attacker, facilitating targeted session attacks.

Finally, to simulate phishing attacks, the model retrieves the target server URL from a publicly accessible channel, mimicking potentially compromised online sources susceptible to manipulation. In a real-world scenario, a vigilant user would typically compare this obtained URL against the known and legitimate address of the IdP server. However, in attack scenarios involving phishing, our model assumes that the targeted user bypasses this verification step.

4.3 Modelling security properties

The main property we consider is the *injective correspondence* of login attempts and successful logins: every successful login authorization given by the IdP to a device must correspond to a unique login attempt initiated by the user on the very same device. In ProVerif, this is rendered by a query as follows:

$$\text{query } x : id;\ \text{inj-event}(Login(x)) \Rightarrow \text{inj-event}(UserLoginRequest(x))$$

where the $Login(x)$ event is raised by the IdP server when it successfully concludes a login attempt, and $UserLoginRequest(x)$ is generated by the user when they initiate a login attempt.

Sometimes, ProVerif may fail to verify the injective correspondence security goal, leading to either a timeout or a "Cannot be proven" result. In such cases, we can employ a strategy to guide the verifier towards discovering a concrete attack trace. Specifically, we modify the security query as follows:

$$\text{query } x : id;\ \text{event}(Login(x)) \Rightarrow \text{event}(UserLoginRequest(x)).$$

where $Login(x)$ and $UserLoginRequest(x)$ are defined as above.

This modified query compels ProVerif to search for an attack scenario where a successful login ($Login$ event) occurs without any preceding login initiation by the legitimate user ($UserLoginRequest$ event). Requiring a $Login$ event to happen without a prior $UserLoginRequest$ represents a more challenging setting for the attacker than the original injective correspondence goal. Consequently, if ProVerif discovers an attack trace satisfying this query, it automatically demonstrates a violation of the original injective correspondence goal as well, thus indicating a protocol vulnerability within the analysed scenario. On the other hand, a positive verification outcome (i.e. no attack found) for this modified query does not necessarily guarantee the protocol's security with respect to the original injective correspondence goal: the injective correspondence property might still be violated in ways that are not captured by this more specific query.

4.4 Dynamic generation and exploration of attack scenarios

As detailed in Sect. 3, a comprehensive security evaluation of the CieID protocols necessitates testing them against all attack scenarios representing the spectrum of control an attacker might gain over various interfaces, encompassing even human-induced threats like phishing. Consequently, for each protocol's formalization P, we must generate and analyse a corresponding variant $P_{\mathcal{S}}$ for every scenario $\mathcal{S}$ identified within our threat model. As illustrated at the end of Sect. 3, this can lead to analyse over 6600 distinct ProVerif models.

```
results ← new map[Scenario → Result]{}
for all sc ∈ Scenario do
    if results[sc] ≠ ⊥ then
        continue
    end if
    res ← VERIFY(sc)
    if res = TIMEOUT ∨ res = CANNOTBEPROVEN then
        res′ ← VERIFYNOINJ(sc)
        if res′ = VULN then
            res ← VULN
        end if
    end if
    results[sc] ← res
    if res = VULN then
        for all sc′ ∈ Scenario s.t. sc′ ⇒ sc do
            results[sc′] ← res
        end for
    else if res = SAFE then
        for all sc′ ∈ Scenario s.t. sc ⇒ sc′ do
            results[sc′] ← res
        end for
    end if
end for
```

Fig. 9 Pseudocode for the dynamic generation and exploration of attack scenarios

To streamline the generation of these numerous ProVerif models, we have developed a general *model template* and utilized the m4 macro processor [19]. Within this template, each potential attacker control and human-induced threat is associated with a specific set of pre-processor definitions. We have created a Python script to systematically iterate through the entire range of possible scenarios; the corresponding pseudocode is presented in Fig. 9.

For each scenario, the script employs m4 to generate the corresponding ProVerif model instance, which is then automatically verified by ProVerif. Furthermore, the script incorporates an optimization to automatically skip the verification of model instances whose outcome (security or vulnerability) can be logically inferred from the results of previously analysed scenarios, i.e.:

- if $P_{\mathcal{S}}$ is found to be vulnerable, we can skip the generation and analysis of $P_{\mathcal{S}'}$ for all $\mathcal{S}'$ stronger than $\mathcal{S}$ ($\mathcal{S}' \Rightarrow \mathcal{S}$);
- if $P_{\mathcal{S}}$ is found to be secure, we can skip the generation and analysis of $P_{\mathcal{S}'}$ for all $\mathcal{S}'$ weaker than $\mathcal{S}$ ($\mathcal{S} \Rightarrow \mathcal{S}'$).

5 Analysis results

This section presents the findings of our analysis concerning the security properties of the CIE Level 2 and Level 3 authentication protocols. We also discuss the scope and limits of formal proofs in the symbolic model.

5.1 Level 2 protocols: L2SMS, L2PSH, L2QRC

Although CIE level 2 protocols provide a stronger defence against unauthorized access than passwords alone, they are not without vulnerabilities. Table 2 reports the findings of our formal analysis of these authentication schemes.

L2SMS offers a degree of resistance against brute-force attacks targeting stolen credentials, but it remains vulnerable to more sophisticated techniques. Notably, malware such as USB keyloggers can present a significant threat by intercepting the one-time passcodes delivered via SMS. An attacker who captures these codes can bypass L2SMS authentication, rendering this security measure ineffective.

Compared to protocols utilizing TLS channels, L2SMS suffers from inherent transport-level security limitations. *SIM swapping* attacks, in which an attacker fraudulently transfers a victim's phone number to a SIM card they control, represent a significant vulnerability. Furthermore, L2SMS may be susceptible to SS7 attacks exploiting weaknesses within the signalling system and the weaker encryption protocols employed in older UMTS and GSM networks [36].

Our analysis revealed a problematic aspect of L2SMS: although it eliminates the need for on-screen code scanning by relying solely on SMS delivery of the one-time passcode, this very passcode is subsequently displayed on the computer screen, rendering it vulnerable to malware capable of capturing screen output.

Currently, the most problematic aspect of L2SMS lies in the fact that CIE users do not have the option to disable L2SMS logins even after registering CieID devices for stronger Level 2 protocols. This oversight introduces a potential attack vector, undermining the security benefits intended by the user's preference for more robust protocols. Furthermore, the process for enrolling new L2QRC devices relies on an authentication mechanism similar to L2SMS, raising questions about its efficacy as a genuine security upgrade.

L2QRC offers stronger security compared to L2SMS. It successfully avoids attacks passively extracting data from nearly all interfaces examined in the analysis, as demonstrated by the scenario $\{\mathcal{A}^{\text{pc-tls}}_{\text{io:RO}}, \mathcal{A}^{\text{pc-kbd}}_{\text{in:RO}}, \mathcal{A}^{\text{pc-displ}}_{\text{out:RO}}, \mathcal{A}^{\text{mob-hum}}_{\text{io:RO}}, \mathcal{A}^{\text{mob-tls}}_{\text{io:RW}}\}$. However, L2QRC struggles against active attackers, except in the case of the Mobile TLS interface which can be fully compromised without hindering the protocol security. Many identified attacks involve substituting (through direct or indirect control of available system interfaces) the login operation identifiers presented to the CieID app. These attacks exploit the fact that the CieID app does not show to the user any element to identify the login session (such as device fingerprints [2]) during the L2QRC authentication phase: this prevents

Table 2 Analysis results of Level 2 protocols under different threat scenarios

Threat Scenario	L2SMS	L2PSH	L2QRC
	✓	✗	✓
$\mathcal{A}^{\text{mob-sms}}_{\text{in:RO}}$	✗	–	–
$\mathcal{A}^{\text{mob-hum}}_{\text{in:RW}}$	✓	✗	✗
$\mathcal{A}^{\text{mob-hum}}_{\text{io:RO}}$	✗	✗	✓
$\mathcal{A}^{\text{pc-displ}}_{\text{out:RO}}$	✗	✗	✓
$\mathcal{A}^{\text{pc-displ}}_{\text{out:RW}}$	✗	✗	✗
$\mathcal{A}^{\text{pc-kbd}}_{\text{in:RO}}$	✗	✗	✓
$\mathcal{A}^{\text{pc-kbd}}_{\text{in:RW}}$	✗	✗	✗
$\mathcal{A}^{\text{pc-tls}}_{\text{io:RO}}$	✗	✗	✓
$\mathcal{A}^{\text{pc-tls}}_{\text{io:RO}}$, $\mathcal{A}^{\text{pc-kbd}}_{\text{in:RO}}$, $\mathcal{A}^{\text{pc-displ}}_{\text{out:RO}}$, $\mathcal{A}^{\text{mob-hum}}_{\text{io:RO}}$, $\mathcal{A}^{\text{mob-tls}}_{\text{io:RW}}$	–	✗	✓
$\mathcal{A}^{\text{pc-tls}}_{\text{io:RW}}$	✗	✗	✗
PH	✗	✗	✗

Results:

- ✓ security proved
- ✓ security implied by the security proved in a stronger scenario
- ✗ attack found
- ✗ attack found in a weaker scenario
- – threat scenario not applicable to the protocol

Scenarios:

- PH: Phishing Attack
- $\mathcal{A}^{if}_{d:a}$: Attacker control over a system interface, where
  - *if* is the name of the system interface
  - *d* represents control over interface's inputs (in), outputs (out) or both (io)
  - *a* is the access control over the interface: read-only (RO) or read–write (RW)

security-conscious users to recognize suspicious operations (like concurrent login attempts) and abort the authentication.

Our analysis identified an interesting attack applicable in scenario $\{\mathcal{A}^{\text{pc-kbd}}_{\text{in:RW}}\}$. By modifying the keyboard input, an attacker can manipulate the URL a user intends to visit, essentially redirecting them to a malicious webpage. While similar to phishing attacks, this method operates on a lower level: phishing deceives users into visiting malicious websites by exploiting human behaviour, whereas this attack directly alters user input, bypassing the user's awareness.

L2PSH exhibits the weakest security properties: an attacker can achieve a Level 2 login by possessing only Level 1 credentials, without needing to compromise any communication interface. Indeed, ProVerif identified a successful attack trace even within the *empty* scenario (representing an uncompromised system), the corresponding sequence diagram of which is shown in Fig. 10.

This attack is quite similar to the timing attack identified on Google's G2OT, as detailed in [18]. In fact, L2PSH may be even more vulnerable due to its handling of concurrent access attempts: the CieID app overwrites previous notifications, thereby hindering the user's ability to detect potentially fraudulent duplicate login attempts. Furthermore, L2PSH fails to provide the user with any information regarding the device fingerprint or the specifics of the attempted operation within the login request, further increasing the risk.

Therefore, to circumvent L2PSH authentication, an attacker merely needs to monitor network traffic for a legitimate access attempt initiated by the user. Upon detecting such an attempt, the attacker swiftly initiates a malicious login attempt using the user's Level 1 credentials. The legitimate access notification displayed to the user is then subtly overwritten by the notification from the attacker's malicious attempt. The unsuspecting user proceeds to authorize what they believe is their own login attempt via the notification, inadvertently granting the attacker access to their account.

The absence of any obvious anomaly during this process renders the attack highly deceptive. The sole indication is a brief moment where the legitimate notification is replaced by an identical-looking one; however, this subtle change is unlikely to be noticed by the majority of users, even security-conscious ones. Notably, this attack has been successfully implemented as a proof of concept on one of the authors' CIE accounts, demonstrating its actual feasibility.

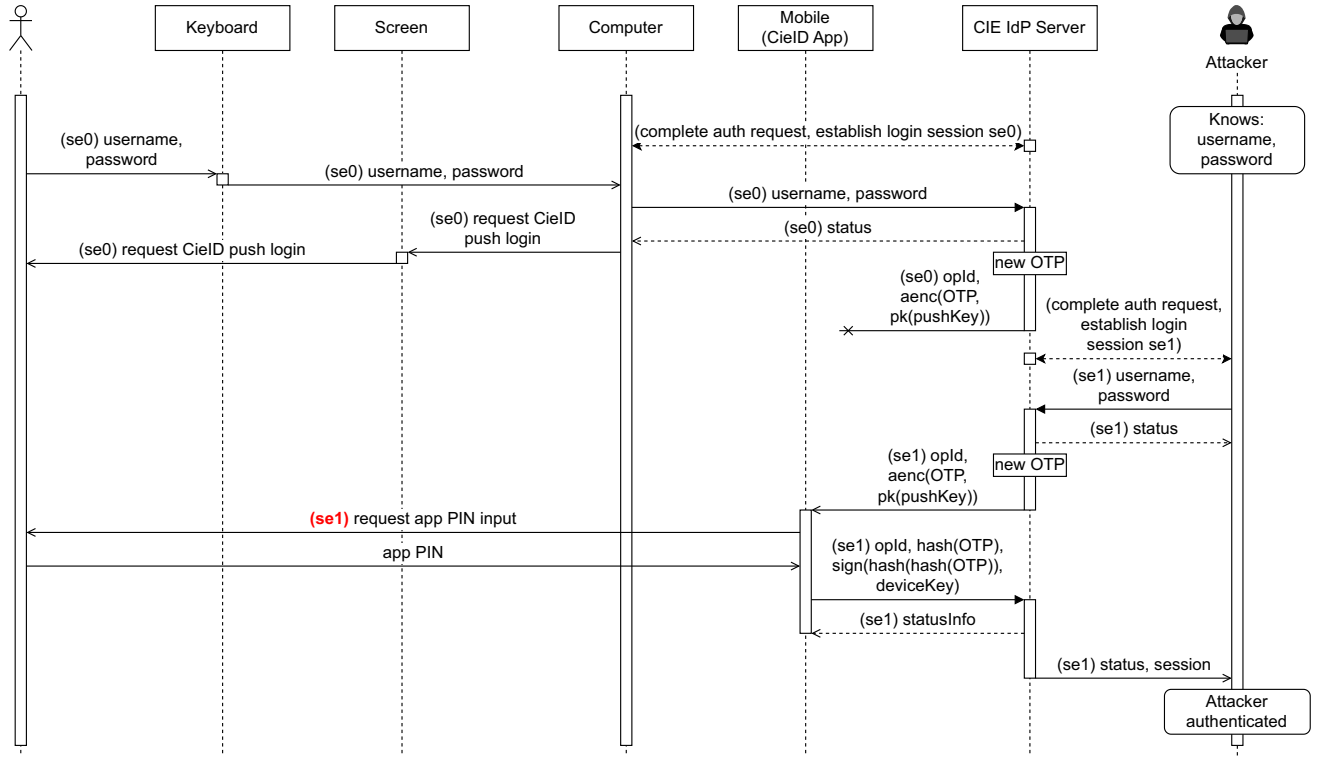

Fig. 10 Overview of the L2PSH attack in the empty scenario

5.2 Level 3 protocols: L3APP, L3DSK

The CIE Level 3 protocols represent a notable security enhancement over Level 2, primarily due to the integration of the physical CIE smart card as the additional authentication factor. To successfully compromise Level 3 authentication, an attacker must either gain control over the communication channel with this more secure physical device or manipulate the user into performing actions with the smart card on their behalf. This latter strategy falls under the umbrella of social engineering; however, when it involves a physical smart card, users might perceive it as a more unusual, potentially risky and therefore more suspicious activity compared to purely digital social engineering tactics. The findings of our analysis concerning the CIE Level 3 protocols are summarized in Table 3.

L3APP employs a significant number of interfaces and exhibits a notably complex flow. This complexity contributes to a broader attack surface compared to protocols with simpler architectures. Specifically, the transmission of sensitive data across this multitude of interfaces and devices creates numerous opportunities for attackers to exploit different techniques at various levels. This is clearly illustrated in Table 3 by the substantial number of "proven" attacks (i.e. those not implied by weaker scenarios), each corresponding to a distinct attack path.

Actually, several of these attacks share the aim of recovering the PIN and the OTP through various means. Consider, for instance, an attacker capable of initiating communication with the smart card and forcing it to sign the IdP challenge (effectively having access to the mob-hum interface input stream). This capability alone is insufficient for successful authentication, as while the smart card PIN may be acquired by observing the input provided by the user during a legitimate login attempt, the attacker would still need to gain access to output OTPs. The OTP can be obtained through multiple ways: an attacker might gain full read–write access to the user's smartphone, enabling them to observe the screen and capture displayed information ($\{\mathcal{A}^{\text{mob-hum}}_{\text{io:RW}}\}$); alternatively, they could log keystrokes on the computer's keyboard, recording the OTP as it is entered ($\{\mathcal{A}^{\text{pc-kbd}}_{\text{in:RO}}, \mathcal{A}^{\text{mob-hum}}_{\text{in:RW}}\}$); or the OTP could be intercepted by compromising the TLS interface between the computer and the IdP server ($\{\mathcal{A}^{\text{pc-tls}}_{\text{io:RO}}, \mathcal{A}^{\text{mob-hum}}_{\text{in:RW}}\}$).

As for L2SMS, the OTP code is displayed on the computer' screen, and hence, it can be observed by an attacker that has control over the screen's output channel, e.g. by some malware capable of capturing screen output.

Unfortunately, due to the complex handling and transmission of secrets, the identified vulnerabilities in L3APP cannot be fixed with some minor adjustments. Nevertheless, L3APP offers at least the same level of security as L2QRC.

Table 3 Analysis results of Level 3 protocols under different threat scenarios. See Table 2 for the legend

Threat Scenario									L3APP	L3DSK
						$\mathcal{A}^{\text{mob-sm}}_{\text{io:RW}}$		✗	–
					$\mathcal{A}^{\text{mob-hum}}_{\text{in:RW}}$			✓	–
					$\mathcal{A}^{\text{mob-hum}}_{\text{in:RW}}$		$\mathcal{A}^{\text{mob-tls}}_{\text{io:RO}}$	✗	–
					$\mathcal{A}^{\text{mob-hum}}_{\text{in:RW}}$	$\mathcal{A}^{\text{mob-sm}}_{\text{io:RO}}$		✗	–
					$\mathcal{A}^{\text{mob-hum}}_{\text{io:RW}}$			✗	–
				$\mathcal{A}^{\text{pc-displ}}_{\text{out:RO}}$	$\mathcal{A}^{\text{mob-hum}}_{\text{in:RW}}$			✗	–
				$\mathcal{A}^{\text{pc-displ}}_{\text{out:RW}}$				✗	✓
			$\mathcal{A}^{\text{pc-kbd}}_{\text{in:RO}}$		$\mathcal{A}^{\text{mob-hum}}_{\text{in:RW}}$			✗	–
			$\mathcal{A}^{\text{pc-kbd}}_{\text{in:RW}}$					✗	✗
		$\mathcal{A}^{\text{pc-tls}}_{\text{io:RO}}$			$\mathcal{A}^{\text{mob-hum}}_{\text{in:RW}}$			✗	–
		$\mathcal{A}^{\text{pc-tls}}_{\text{io:RO}}$	$\mathcal{A}^{\text{pc-kbd}}_{\text{in:RO}}$	$\mathcal{A}^{\text{pc-displ}}_{\text{out:RO}}$	$\mathcal{A}^{\text{mob-hum}}_{\text{io:RO}}$	$\mathcal{A}^{\text{mob-sm}}_{\text{io:RO}}$	$\mathcal{A}^{\text{mob-tls}}_{\text{io:RW}}$	✓	–
		$\mathcal{A}^{\text{pc-tls}}_{\text{io:RW}}$						✗	✓
	$\mathcal{A}^{\text{pc-sm}}_{\text{io:RW}}$							–	✗
PH								✗	✓
PH	$\mathcal{A}^{\text{pc-sm}}_{\text{io:RO}}$	$\mathcal{A}^{\text{pc-tls}}_{\text{io:RW}}$	$\mathcal{A}^{\text{pc-kbd}}_{\text{in:RO}}$	$\mathcal{A}^{\text{pc-displ}}_{\text{out:RW}}$				–	✓

The enhanced security stems from the requirement that the smart card must be in physical contact with the mobile device during authentication. This necessity of a physical token introduces a deliberate and somewhat unusual user interaction, potentially increasing the difficulty for attackers due to the inherent suspicion such an action might evoke in a malicious context.

L3DSK stands out as the most robust protocol within the CIE suite, notably guaranteeing the absence of successful attacks even when phishing is possible and the attacker gains access to multiple interfaces (as indicated in the last row of Table 3). Still, our analysis reveals two distinct potential failure modes. The first attack scenario arises when the communication channel with the smart card is compromised with read–write access ($\{\mathcal{A}^{\text{pc-sm}}_{\text{io:RW}}\}$). This finding is expected, as such a level of control would allow the attacker to fully emulate the operating system's smart card communication middleware. The second vulnerability occurs when an attacker successfully obtains the card's PIN and subsequently re-enters it to authorize a new session, by exploiting the computer's keyboard channel ($\{\mathcal{A}^{\text{pc-kbd}}_{\text{in:RW}}\}$). These limitations appear to be intrinsic to the protocol's design and are not easily resolvable through minor modifications.

Addressing these security concerns, especially the PIN retrieval and replay attack, necessitates a fundamental architectural shift. One potential solution involves replacing the smart card with a personal, tamper-resistant authenticator device. This dedicated hardware could feature an embedded screen to display session-specific information, such as a dynamic code, enhancing user awareness of the transaction. Additionally, an integrated, tamper-proof keyboard would provide a secure method for PIN entry, mitigating the risk of eavesdropping. While likely impractical for current eIDAS implementations, such a hardware-based approach could offer enhanced security guarantees in smaller, controlled, high-security environments where dedicated hardware solutions are feasible.

5.3 On the scope of symbolic modelling

In our framework, cryptographic primitives are treated as "perfect black boxes" following the standard Dolev–Yao algebraic assumptions. This approach intentionally abstracts away the computational complexity of specific attack techniques—such as side-channel analysis or differential cryptanalysis—to enable a rigorous verification of the protocol's structural logic.

The alignment between symbolic abstractions and real-world deployments is a foundational topic in formal methods [1, 6, 38]. As with any scientific model, a fundamental trade-off exists: excessive abstraction risks overlooking implementation-specific vulnerabilities, while excessive granularity often renders systematic reasoning computationally infeasible.

We position the symbolic model as a *functional specification* that defines the mandatory security properties any compliant implementation must satisfy. For instance, if a protocol is proven secure assuming perfect encryption, the subsequent use of a compromised primitive constitutes a fail-

ure of the implementation to meet the specification, rather than a flaw in the model itself.

However, if a model fails to capture a relevant class of threats, it must be refined. Our interface-based methodology is designed specifically for this type of evolution; it is inherently modular and extensible, allowing for the integration of new components or interfaces to formalize emerging threat vectors (e.g. side-channel or fault-injection) as they are identified.

This approach establishes a clear *hierarchy of security*: a symbolic proof provides a necessary, though not exhaustive, precondition for computational and physical security. A protocol that is logically flawed at the symbolic level cannot be "rescued" by any cryptographic implementation. Therefore, according to the "fail-fast" principle, our work serves as a critical first filter: it identifies which protocol architectures are fundamentally robust against defined threat models.

This analysis is intended to be *complementary* to heuristic and computational methods [10, 38]. The primary strength of our framework lies in its systematic exhaustiveness: by abstracting the mechanism of an attack into its logical consequences (interface compromise), we can automatically evaluate thousands of scenarios in hours—a task that would require years of manual heuristic effort.

By establishing this baseline, our work provides a robust architectural blueprint for eIDAS-compliant MFA. Once a protocol is symbolically verified, heuristic analysis can be targeted towards the high-value task of bridging the "last mile" between these verified specifications and their actual realizations, ensuring that implementation-specific details, such as cryptographic agility and side-channel resistance, satisfy the high-assurance requirements of the eIDAS ecosystem.

6 Protocol improvement

The analysis detailed in Sect. 5 reveals that all Level 2 and Level 3 protocols exhibit certain vulnerabilities under specific attack scenarios. For instance, in L2SMS, if an attacker gains control of the user's telephone number, the SMS-delivered code can be intercepted. Similarly, in L2PSH, if the user's smartphone is compromised by malware, the attacker could potentially forward their own push notification to the user. In L2QRC, an attacker capable of displaying arbitrary data on the user's computer screen could present a malicious QR code. These examples illustrate the weaknesses under specific compromise conditions.

It is important to note that most of these vulnerabilities stem from attacks targeting the communication channels and the devices themselves, rather than from logical flaws inherent to the protocols. Consequently, addressing these vulnerabilities would necessitate a redesign of the underlying architecture. Nevertheless, by including these attack vectors in our threat model, we can analyse the protocols' resilience even when the communication channels or the platforms have been compromised. For example, our analysis indicates that L2QRC remains secure even if an attacker manages to decrypt TLS traffic or intercept keyboard input, and L3DSK remains secure even when phishing is possible and the attacker gains access to multiple interfaces.

Still, certain weaknesses are intrinsic to the design of the protocols, and these can potentially be mitigated by minor changes. We propose an improvement of L2SMS, L3APP and L2PSH protocols, in order to address the display channel vulnerability of the formers and the "notification overwrite" of the latter.

6.1 L2SMS*: L2SMS improvement

The ProVerif trace from our analysis reveals that the vulnerability in L2SMS lies in the display of the OTP code on the user's screen, thereby exposing it to attackers employing read-only screen access methods such as screen sharing software or "shoulder surfing".

To rectify this issue, it is sufficient to eliminate the step of displaying the OTP code on the screen after keyboard input. Formal verification of this modified protocol, which we denote as L2SMS*, confirms the effective mitigation of this vulnerability (see Table 4). This implies that an attacker with read-only or read–write access to the display channel can no longer achieve unauthorized access. The verification of the revised protocol shows no further security concerns.

To prevent attackers from observing the OTP during user input, a possibility is to give no feedback at all, by eliminating any character display on the screen while typing the code. Alternatively, characters can be masked, by replacing the actual digits with asterisks or bullets as they are typed. This keeps the actual code hidden from view, while providing some basic user confirmation.

6.2 L3APP*: L3APP improvement

Our analysis of L3APP uncovered a vulnerability similar to the one addressed by L2SMS*: both L2SMS and L3APP present the OTP on the screen, making it vulnerable to screen observers. To fix this, mirroring our approach with L2SMS, we propose masking the characters of the OTP as the user types them. This measure would prevent attackers with screen access from observing and potentially exploiting the OTP. Formal verification of L3APP*, the modified version of L3APP incorporating this masking, demonstrates the successful mitigation of this vulnerability, confirming the protocol's security even in scenarios where an attacker has full control over the display channel.

Table 4 Comparison with the improved protocols. See Table 2 for the rest of legend

Threat Scenario							L2SMS	L2SMS*	L2PSH	L2PSH*	L2QRC	L3APP	L3APP*	L3DSK
							✓	✓	✗	✓	✓	✓	✓	✓
$\mathcal{A}^{\text{mob-sms}}_{\text{in:RO}}$							✗	✗	–	–	–	–	–	–
$\mathcal{A}^{\text{mob-sm}}_{\text{io:RW}}$							–	–	–	–	–	✗	✗	–
					$\mathcal{A}^{\text{mob-hum}}_{\text{in:RW}}$		✓	✓	✗	(✗)	✗	✓	✓	–
					$\mathcal{A}^{\text{mob-hum}}_{\text{in:RW}}$	$\mathcal{A}^{\text{mob-tls}}_{\text{io:RO}}$	–	–	✗	✗	✗	✗	✗	–
$\mathcal{A}^{\text{mob-sm}}_{\text{io:RO}}$					$\mathcal{A}^{\text{mob-hum}}_{\text{in:RW}}$		–	–	–	–	–	✗	✗	–
					$\mathcal{A}^{\text{mob-hum}}_{\text{io:RO}}$		✗	✗	✗	✓	✓	✓	✓	–
					$\mathcal{A}^{\text{mob-hum}}_{\text{io:RW}}$		✗	✗	✗	✗	✗	✗	✗	–
				$\mathcal{A}^{\text{pc-displ}}_{\text{out:RO}}$			✗	✓	✗	✓	✓	✓	✓	✓
				$\mathcal{A}^{\text{pc-displ}}_{\text{out:RO}}$	$\mathcal{A}^{\text{mob-hum}}_{\text{in:RW}}$		✗	✓	✗	✗	✗	✗	✓	–
				$\mathcal{A}^{\text{pc-displ}}_{\text{out:RW}}$			✗	✓	✗	✗	✗	✗	✓	✓
				$\mathcal{A}^{\text{pc-displ}}_{\text{out:RW}}$		$\mathcal{A}^{\text{mob-tls}}_{\text{io:RO}}$	–	–	✗	✗	✗	✗	✗	–
$\mathcal{A}^{\text{mob-sm}}_{\text{io:RO}}$				$\mathcal{A}^{\text{pc-displ}}_{\text{out:RW}}$			–	–	–	–	–	✗	✗	–
				$\mathcal{A}^{\text{pc-displ}}_{\text{out:RW}}$	$\mathcal{A}^{\text{mob-hum}}_{\text{in:RW}}$		✗	✓	✗	✗	✗	✗	✓	–
				$\mathcal{A}^{\text{pc-displ}}_{\text{out:RW}}$	$\mathcal{A}^{\text{mob-hum}}_{\text{io:RO}}$		✗	✗	✗	✗	✗	✗	✗	–
			$\mathcal{A}^{\text{pc-kbd}}_{\text{in:RO}}$				✗	✗	✗	✓	✓	✓	✓	✓
			$\mathcal{A}^{\text{pc-kbd}}_{\text{in:RO}}$		$\mathcal{A}^{\text{mob-hum}}_{\text{in:RW}}$		✗	✗	✗	✗	✗	✗	✗	–
			$\mathcal{A}^{\text{pc-kbd}}_{\text{in:RO}}$	$\mathcal{A}^{\text{pc-displ}}_{\text{out:RW}}$			✗	✗	✗	✗	✗	✗	✗	✓
			$\mathcal{A}^{\text{pc-kbd}}_{\text{in:RW}}$				✗	✗	✗	✗	✗	✗	✗	✗
		$\mathcal{A}^{\text{pc-tls}}_{\text{io:RO}}$					✗	✗	✗	✓	✓	✓	✓	✓
		$\mathcal{A}^{\text{pc-tls}}_{\text{io:RO}}$			$\mathcal{A}^{\text{mob-hum}}_{\text{in:RW}}$		✗	✗	✗	✗	✗	✗	✗	–
		$\mathcal{A}^{\text{pc-tls}}_{\text{io:RO}}$		$\mathcal{A}^{\text{pc-displ}}_{\text{out:RW}}$			✗	✗	✗	✗	✗	✗	✗	✓
		$\mathcal{A}^{\text{pc-tls}}_{\text{io:RO}}$	$\mathcal{A}^{\text{pc-kbd}}_{\text{in:RO}}$	$\mathcal{A}^{\text{pc-displ}}_{\text{out:RO}}$	$\mathcal{A}^{\text{mob-hum}}_{\text{io:RO}}$	$\mathcal{A}^{\text{mob-tls}}_{\text{io:RW}}$	–	–	✗	✓	✓	✓	✓	–
$\mathcal{A}^{\text{mob-psh}}_{\text{in:RW}}$		$\mathcal{A}^{\text{pc-tls}}_{\text{io:RO}}$	$\mathcal{A}^{\text{pc-kbd}}_{\text{in:RO}}$	$\mathcal{A}^{\text{pc-displ}}_{\text{out:RO}}$	$\mathcal{A}^{\text{mob-hum}}_{\text{io:RO}}$	$\mathcal{A}^{\text{mob-tls}}_{\text{io:RW}}$	–	–	✗	✓	–	–	–	–
$\mathcal{A}^{\text{mob-sm}}_{\text{io:RO}}$		$\mathcal{A}^{\text{pc-tls}}_{\text{io:RO}}$	$\mathcal{A}^{\text{pc-kbd}}_{\text{in:RO}}$	$\mathcal{A}^{\text{pc-displ}}_{\text{out:RO}}$	$\mathcal{A}^{\text{mob-hum}}_{\text{io:RO}}$	$\mathcal{A}^{\text{mob-tls}}_{\text{io:RW}}$	–	–	–	–	–	✓	✓	–
		$\mathcal{A}^{\text{pc-tls}}_{\text{io:RW}}$					✗	✗	✗	✗	✗	✗	✗	✓
$\mathcal{A}^{\text{pc-sm}}_{\text{io:RW}}$							–	–	–	–	–	–	–	✗
NC							–	–	–	✗	–	–	–	–
	PH						✗	✗	✗	✗	✗	✗	✗	✓
$\mathcal{A}^{\text{pc-sm}}_{\text{io:RO}}$	PH	$\mathcal{A}^{\text{pc-tls}}_{\text{io:RW}}$	$\mathcal{A}^{\text{pc-kbd}}_{\text{in:RO}}$	$\mathcal{A}^{\text{pc-displ}}_{\text{out:RW}}$			–	–	–	–	–	–	–	✓

Results:

-(✗) ProVerif finds attack after a minor manual modification to the scenario's code

Scenarios:

- NC: no-check scenario (the user skips the verification of the session identifier)

6.3 L2PSH*: L2PSH improvement

The attack trace visualized in Fig. 10 reveals a critical vulnerability intrinsic to the protocol's design, as it remains susceptible to exploitation even without any interfaces being compromised by an attacker.

To address this inherent vulnerability and enhance the protocol's security, we propose implementing a verification mechanism for push notifications. This mechanism would enable users to confirm that the push notification received on their device corresponds to their initiated login attempt. Should the verification fail, the user must cancel the login attempt to prevent unauthorized access. We call this enhanced protocol L2PSH*, and it is illustrated in Fig. 11.

The proposed session code mechanism, while theoretically robust, remains susceptible to human error, as users may approve requests without verifying the codes. To ensure active engagement, a selection-based approach is recommended. In this model, instead of passively viewing a code, the user must identify and select the correct matching code from a randomized grid displayed on their mobile device. This forces a cognitive check, ensuring the user confirms the

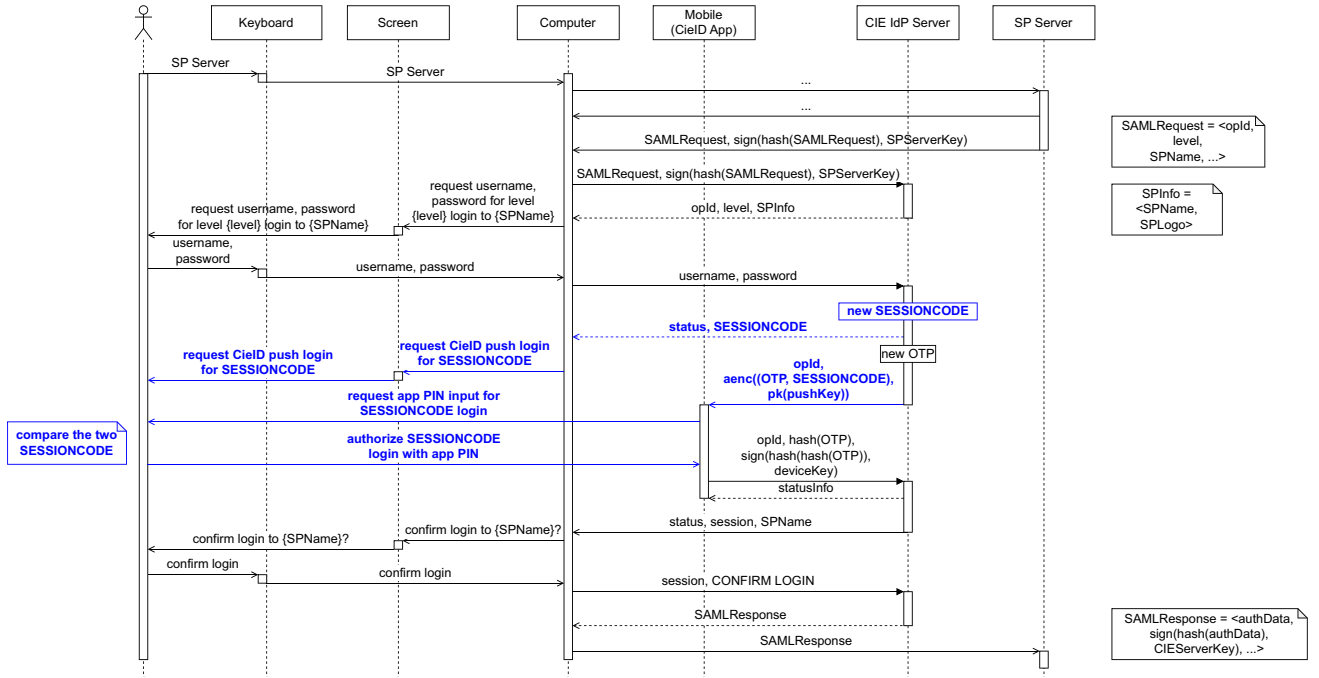

Fig. 11 Sequence diagram of the L2PSH* protocol, Level 2 authentication via app push notification improved with unique session code. In blue the exchange steps that have been added or modified. During the login process, the code (e.g. a number or a short passphrase) is generated by the IdP server and displayed on the user's computer screen. Simultaneously, the same code is sent along within the push notification and shown to the user on the mobile device. The user should then check that the two codes match, before proceeding with authorizing the login on the CieID mobile app

link between the PC session and the mobile authorization before the login proceeds.

As demonstrated in Table 4, the formal analysis in ProVerif of L2PSH* confirms that the vulnerabilities inherent in L2PSH have been successfully mitigated; thus, L2PSH* achieves a security level equivalent to that of L2QRC.

The verification of L2PSH* in the $\{\mathcal{A}^{\text{mob-hum}}_{\text{in:RW}}\}$ scenario warrants a detailed discussion. Among the hundreds of scenarios automatically generated by the algorithm in Sect. 4, this is the sole case where ProVerif remained inconclusive, terminating with "Cannot be proven". As noted in Sect. 4.1, this result occurs when the tool identifies a logical derivation of an attack but is unable to reconstruct a corresponding execution trace due to the abstractions employed.

As a general principle, an inconclusive "Cannot be proven" result often signals a potential vulnerability. This can be confirmed by a manual analysis of the logical derivation provided by ProVerif. Moreover, targeted model refinements can be employed to assist the tool in successfully reconstructing a concrete attack trace, thereby confirming the threat.

In the case of L2PSH* in the $\{\mathcal{A}^{\text{mob-hum}}_{\text{in:RW}}\}$ scenario, the analysis of the logical derivation indeed suggested a vulnerability. We then applied a minor refinement to the model to guide ProVerif's reconstruction algorithm. Specifically, we introduced parallelism immediately preceding the OTP output on the mobile screen. This modification effectively grants the attacker temporary Write access to the mobile screen ($\{\mathcal{A}^{\text{mob-hum}}_{\text{out:RW}}\}$). While this represents an over-approximation of the intended threat model (since the attacker should not possess write access to the screen in this scenario), it served as the necessary catalyst for ProVerif's search process: in this extended model ProVerif is able to reconstruct the attack trace.

The attack consists in retrieving the app PIN by exploiting read access to the mobile device's touchscreen and subsequently using it in a separate session to authorize a login attempt initiated by the attacker. A high-level visualization of this semi-automatically discovered attack trace is shown in Fig. 12.

Crucially, this attack trace *does not* exploit the artificially introduced capability $\mathcal{A}^{\text{mob-hum}}_{\text{out:RW}}$: no write operation on the mobile device's screen is done by the attacker. This proves that the attack is feasible even within the more restrictive original scenario $\{\mathcal{A}^{\text{mob-hum}}_{\text{in:RW}}\}$. This confirms that the initial "Cannot be proven" result was indeed masking a real protocol flaw rather than a false positive.

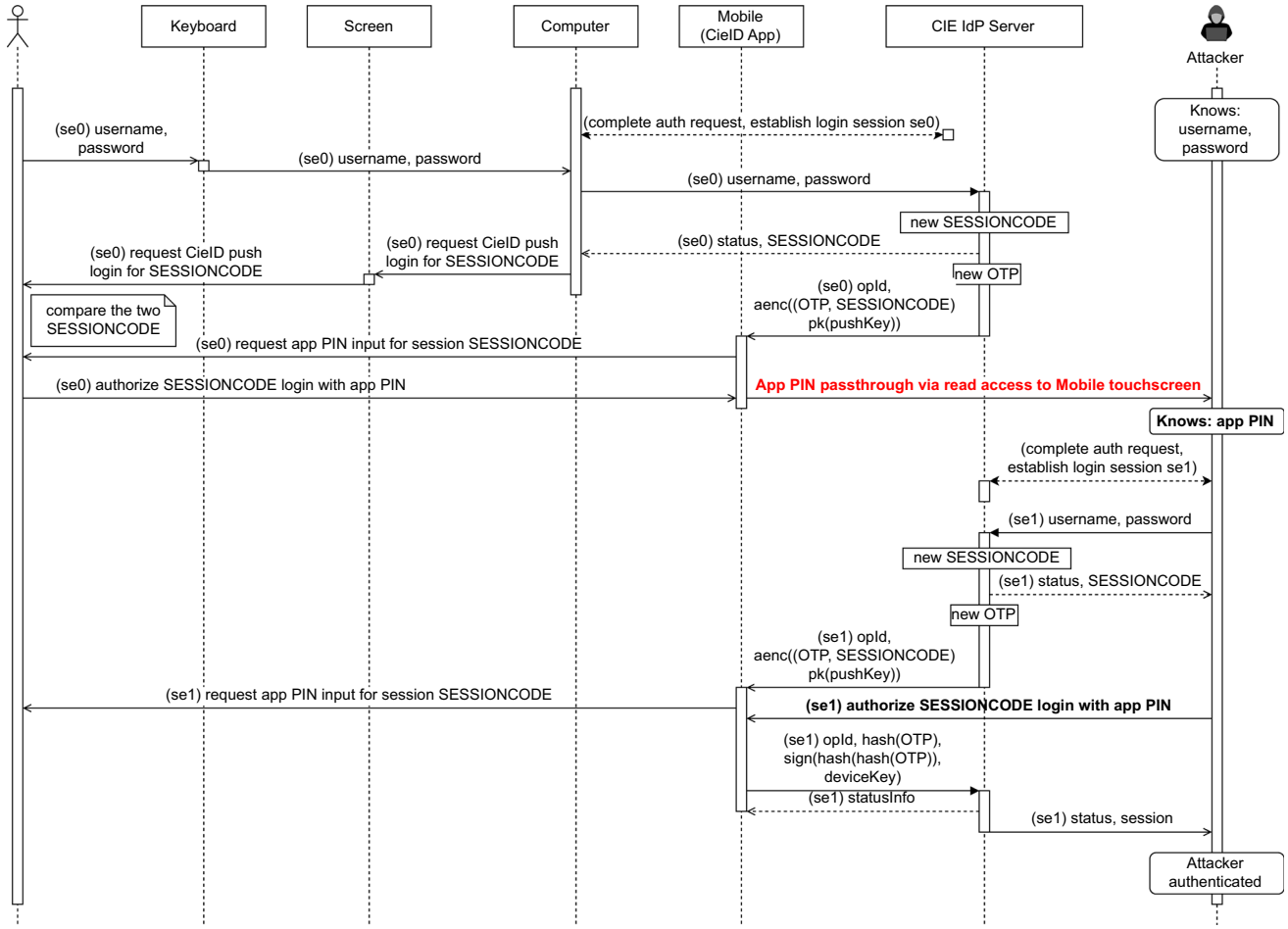

Fig. 12 Overview of the attack to L2PSH* in the $\{\mathcal{A}^{\text{mob-hum}}_{\text{in:RW}}\}$ scenario

7 Related work

Our work builds upon and contributes to the extensive body of research on the formal verification of security protocols, particularly those employing multi-factor authentication. The symbolic Dolev–Yao model, as implemented in tools like ProVerif [8], provides a powerful framework for analysing the security properties of cryptographic protocols against passive and active adversaries.

The relationship between symbolic and computational methodologies remains a cornerstone of cryptographic research, often framed as the "computational justification of the symbolic model". The seminal work of Abadi and Rogaway [1] established the initial formal conditions under which symbolic equivalence implies computational indistinguishability. Subsequent research has expanded this scope to address increasingly complex security properties, such as anonymity by Wang et al. [38] and the development of "computationally sound" automated verification techniques within tools like ProVerif [6].

Our work builds upon this established tradition by utilizing the symbolic model as a rigorous logical filter. By distinguishing between the mechanism of an attack and its logical consequences, we provide a necessary—though not exhaustive—precondition for computational and hardware security. This approach allows us to ensure that a protocol is structurally sound before addressing implementation-specific heuristic concerns, effectively bridging the gap between abstract protocol design and secure real-world deployment [10].

Several studies have applied formal verification techniques to analyse the security of various (multi-factor) authentication schemes [4, 13, 16, 22, 32, 34]. Inspiring for us has been the work by Jacomme and Kremer [18], which presents a comprehensive formal analysis of Google's G2OT protocol. Our work shares a similar goal of rigorously analysing real-world MFA protocols, but focuses on eIDAS MFA, and specifically the CIE Level 2 and 3 authentication mechanisms.

Sinigaglia et al. [34] explored the idea of aggregating fundamental attacker capabilities to simulate various

attack scenarios for MFA in online banking. Besides for the applicative context, our approach distinguishes itself also by employing automated tools to generate formal (ProVerif) models of these scenarios, thereby streamlining the process significantly.

Bhargavan et al. [5] have extensively worked on the formal verification of TLS 1.3, a critical underlying protocol for many web-based authentication schemes. While our work abstracts away some of the complexities of TLS by using a dedicated channel model, we acknowledge the importance of their work in understanding the security foundations of web communication.

The formal analysis of mobile authentication protocols has also been a significant area of research. Laud and Roos [22] presented a formal verification of the Estonian Mobile-ID protocol, a system deployed since 2008 that enables Estonian citizens to authenticate and issue digital signatures. Our analysis of L2PSH and L2SMS contributes to this body of work by formally modelling and verifying these specific implementations within the CIE architecture, revealing vulnerabilities related to notification handling and OTP display.

The vulnerabilities we identified in L2SMS related to SMS interception and SIM swapping are consistent with known weaknesses of out-of-band authentication methods relying on the SS7 network, as discussed by Ullah et al. [36]. Our formal analysis provides a concrete instantiation of these general concerns within the context of the CIE architecture.

The vulnerability of hardware to physical access is a well-documented challenge. In the context of wireless sensor networks (WSNs), Wang et al. [37] provide a foundational perspective on node capture attacks, focusing on the physical extraction of secrets and the resulting collapse of network resilience. These traditional studies typically employ probabilistic or computational models to analyse the mechanisms and difficulty of the breach. Our work shares these concerns but diverges in its level of abstraction. While WSN literature examines the process of capturing a node, our symbolic interface-based methodology abstracts away the specific attack vector (e.g. side-channel or brute-force) to focus exclusively on the logical consequences of the compromise. In our framework, a "node capture" is represented as a resulting state where the attacker possesses read-only or read–write access to an actor's interfaces. This allows us to formally verify the structural integrity of MFA protocols within the eIDAS framework, ensuring protocol logic remains robust even when critical actors are assumed to be compromised. In this sense, our work complements hardware security literature by providing a formal framework for the post-compromise phase.

Research in usable security [39] emphasizes the importance of designing mechanisms that are both effective and user-friendly. Our proposed enhancements to L2PSH aim to strike a balance between security and usability.

8 Conclusions

In this paper, we have proposed a methodology for formal modelling and analysis of multi-factor authentication schemes of eIDAS digital identity cards, using Italian CIE as a guiding example. More specifically, we have provided a threat model for CIE that covers vulnerabilities based on an attacker's access to interfaces of the system architecture. We provided a complete formalization of CIE's Level 2 and 3 authentication schemes using both sequence diagrams and ProVerif code. Then, we have automatically carried out an extensive analysis of these schemes in ProVerif, under thousands of possible attack scenarios identified by the interface-based threat model. This analysis exposes specific security vulnerabilities within the protocols L2SMS, L2PSH and L3APP, for which we have proposed targeted improvements. Moreover, our analysis has highlighted that L3APP has a broader attack surface than L3DSK.

Our methodology is general, as it focuses on *what happens if* interfaces have been compromised, abstracting away the specifics of *how* the control was acquired. This abstraction allows for broad threat modelling, besides traditional sniffing or message forging. For instance, a malware that grants an attacker root access to an actor can be seen as a scenario where the attacker fully controls all interfaces of that actor; a side-channel attack which allows to capture plaintext data flowing from the PC to the smart card is modelled as the attacker acquiring read-only access to the PC's output interface connected to the smart card.

A primary advantage of the proposed methodology is its future-proof design: it inherently accounts for novel or undiscovered vulnerabilities, provided their impact manifests as a compromise of the interfaces defined within the model. Furthermore, the model is highly *extensible*: if new attackable interfaces are discovered, the system can be updated without compromising the validity of previous results, since previously analysed scenarios can be interpreted within the new architecture as assuming the security of the new interfaces.

From our analysis, we may observe that certain channels appear to be non-essential for achieving the security of a protocol. For example, security of the mobile TLS channel appears to be not strictly required for the security of the L2QRC protocol. However, this should not lead us to conclude that the data can be transmitted unprotected, since it is always advisable to minimize the information accessible by the attacker, regardless of whether the security of a specific channel appears strictly necessary for the primary security properties under consideration. This principle of information minimization acts as a crucial defence-in-depth strategy, to avoid to provide the attacker with additional intelligence allowing them to exploit unforeseen attack scenarios.

The methodology we have followed in this work can be applied also to MFA schemes of other eIDAS-compliant dig-

ital identity cards. Scripts are available in the accompanying artefact [31] and should be easily adaptable.

We have informed the Italian National Cybersecurity Authority (ACN) and the IPZS about the results of this research, and we are collaborating for the improvement of the CieID protocols and mobile applications.

Future work. It is widely recognized that side-channel attacks (on timing, power consumption, electromagnetic emissions, etc.) constitute a significant security challenge for smart cards [11, 25, 26, 29]. While an analysis in the symbolic model does not focus on hardware-level vulnerabilities, the interface model used in this work inherently accounts for attacks that leak information over established communication paths. Furthermore, the framework is designed for easy extensibility to accommodate more specialized hardware threats. For instance, a side-channel attack leading to the leakage of a private key stored within the smart card can be formalized by introducing a dedicated, isolated interface connected to the card, granting the attacker read-only access to its output. Similarly, a fault-injection attack that alters the smart card's output can be modelled as the attacker gaining write-only control over the receiving host's input interface. We reserve the formalization and analysis of these specialized extensions for future work.

A complete security analysis of an MFA infrastructure has to consider the full system lifecycle, particularly the often-vulnerable enrolment and recovery procedures [20, 40]. However, these procedures are inherently separate from the core authentication protocols, as they often involve distinct security principals (e.g. dedicated recovery agents or administrators) who do not participate in the authentication exchange between the client, servers and identity provider. Consequently, analysing the security of enrolment and recovery requires a quite different interface-based architecture than the one for the authentication protocols. Moreover, a formal analysis is currently unfeasible for certain systems, such as the CIE, because the underlying enrolment and recovery processes lack the public documentation necessary to accurately model the agents and their interactions. For these reasons, a rigorous analysis of enrolment and recovery procedures deserves a separate and dedicated future work.

Investigating the implementation of official CIE authentication for adherence to security best practices holds promise as a research topic. By verifying such adherence, we can minimize potential attack vectors.

The interface-access threat model could be applied in other domains. A captivating case pertains to *containers*, which inherently possess well-defined interfaces for interaction. We envision integrating the interface-access threat model with formal models of container compositions, such as that from [9].

Acknowledgements This work has been partially supported by the Department Strategic Project on Artificial Intelligence of the University of Udine (2020-25) and the M4C2 I1.3 "SEcurity and RIghts In the CyberSpace - SERICS" (PE00000014 - CUP H73C2200089001), under the National Recovery and Resilience Plan (NRRP) funded by the European Union - NextGenerationEU.

Funding Open access funding provided by Università degli Studi di Udine within the CRUI-CARE Agreement.

Data Availability The artefact [31] includes ProVerif templates for all Level 2 and 3 CieID protocols, scripts to instantiate them into formal models for all threat scenarios and the complete results of the verification process.

Declarations

Conflict of interest The authors have no conflict of interest to declare that are relevant to the content of this article.

References

1. Abadi, M., Rogaway, P.: Reconciling two views of cryptography (the computational soundness of formal encryption). J. Cryptol. **15**(2), 103–127 (2002)
2. Alaca, F., van Oorschot, P.C.: Device fingerprinting for augmenting web authentication: classification and analysis of methods. In: Proc. 32nd Conference on Computer Security Applications. p. 289–301. ACM (2016)
3. Armando, A., Carbone, R., Compagna, L., Cuellar, J., Tobarra, L.: Formal analysis of SAML 2.0 web browser single sign-on: breaking the SAML-based single sign-on for Google apps. In: Proc. 6th ACM workshop on Formal methods in security engineering. pp. 1–10 (2008)
4. Armando, A., Carbone, R., Zanetti, L.: Formal modeling and automatic security analysis of two-factor and two-channel authentication protocols. In: Lopez, J., Huang, X., Sandhu, R. (eds.) Proc. Network and System Security. pp. 728–734. Springer (2013)
5. Bhargavan, K., Blanchet, B., Kobeissi, N.: Verified models and reference implementations for the TLS 1.3 standard candidate. In: 2017 IEEE Symposium on Security and Privacy (S&P). pp. 483–502 (2017)
6. Blanchet, B.: A computationally sound mechanized prover for security protocols. IEEE Trans. Dependable Secure Comput. **5**(4), 193–207 (2008)
7. Blanchet, B.: Proverif users - Research papers (2025), available at https://bblanche.gitlabpages.inria.fr/proverif/proverif-users.html

8. Blanchet, B., et al.: Modeling and verifying security protocols with the applied pi calculus and ProVerif. Foundations and Trends in Privacy and Security **1** (2016)
9. Burco, F., Miculan, M., Peressotti, M.: Towards a formal model for composable container systems. In: Proceedings of the 35th Annual ACM Symposium on Applied Computing. pp. 173–175 (2020)
10. Cortier, V., Kremer, S., Warinschi, B.: A survey of symbolic methods in computational analysis of cryptographic systems. J. Autom. Reason. **46**(3), 225–259 (2011)
11. Dhem, J.F., Koeune, F., Leroux, P.A., Mestré, P., Quisquater, J.J., Willems, J.L.: A practical implementation of the timing attack. In: International Conference on Smart Card Research and Advanced Applications. pp. 167–182. Springer (1998)
12. Dumortier, J.: Regulation EU no 910/2014 on electronic identification and trust services for electronic transactions in the internal market (eIDAS regulation). In: EU Regulation of E-Commerce, pp. 256–289. Edward Elgar Publishing (2017)
13. van Eeden, R., Paier, M., Miculan, M.: A formal analysis of CIE level 2 multi-factor authentication via SMS OTP. In: De Capitani di Vimercati, S., Samarati, P. (eds.) Proc. 21st Inter. Conference on Security and Cryptography, SECRYPT 2024. pp. 483–491 (2024). https://doi.org/10.5220/0012768300003767
14. Engelbertz, N., Erinola, N., Herring, D., Somorovsky, J., Mladenov, V., Schwenk, J.: Security analysis of eIDAS — the Cross-Country authentication scheme in Europe. In: 12th USENIX Workshop on Offensive Technologies (2018)
15. Gregušová, D., Halásová, Z., Peráček, T.: eIDAS regulation and its impact on national legislation: the case of the Slovak republic. Admin. Sci. **12**(4), 187 (2022)
16. Hao, F., Clarke, D.: Security analysis of a multi-factor authenticated key exchange protocol. In: Bao, F., Samarati, P., Zhou, J. (eds.) Applied Cryptography and Network Security. pp. 1–11. Springer (2012)
17. Holgers, T., Watson, D.E., Gribble, S.D.: Cutting through the confusion: A measurement study of homograph attacks. In: USENIX Annual Technical Conference. pp. 261–266 (2006)
18. Jacomme, C., Kremer, S.: An extensive formal analysis of multi-factor authentication protocols. ACM Trans. Privacy Sec. **24**, 1–34 (2021)
19. Kernighan, B., Ritchie, D.: The m4 macro processor. Tech. rep, Bell Laboratories Murray Hill, NJ (1977)
20. Klivan, S., Höltervennhoff, S., Huaman, N., Krause, A., Simko, L., Acar, Y., Fahl, S.: "we've disabled MFA for you": An evaluation of the security and usability of multi-factor authentication recovery deployments. In: Proceedings of the 2023 ACM conference on Computer and Communications Security. p. 3138–3152 (2023). https://doi.org/10.1145/3576915.3623180
21. Kobeissi, N., Bhargavan, K., Blanchet, B.: Automated verification for secure messaging protocols and their implementations: A symbolic and computational approach. In: 2nd IEEE European Symposium on Security and Privacy . pp. 435–450 (2017). https://doi.org/10.1109/EuroSP.2017.38
22. Laud, P., Roos, M.: Formal analysis of the Estonian mobile-id protocol. In: Jøsang, A., Maseng, T., Knapskog, S.J. (eds.) Identity and Privacy in the Internet Age. pp. 271–286. Springer (2009)
23. Lips, S., Bharosa, N., Draheim, D.: eIDAS implementation challenges: The case of Estonia and the Netherlands. In: Electronic Governance and Open Society: Challenges in Eurasia. pp. 75–89. Springer (2020)
24. Mainka, C., Mladenov, V., Feldmann, F., Krautwald, J., Schwenk, J.: Your software at my service: Security analysis of SaaS single sign-on solutions in the cloud. In: Proc. 6th ACM Workshop on Cloud Computing Security. pp. 93–104 (2014)
25. Mangard, S., Oswald, E., Popp, T.: Power analysis attacks: Revealing the secrets of smart cards. Springer (2007)
26. Messerges, T.S., Dabbish, E.A., Sloan, R.H.: Examining smart-card security under the threat of power analysis attacks. IEEE Trans. Comput. **51**(5), 541–552 (2002)
27. Miculan, M., Urban, C.: Formal analysis of Facebook Connect single sign-on authentication protocol. In: SofSem 2011, Proceedings of Student Research Forum. pp. 99–116. OKAT (2011)
28. Miculan, M., Vitacolonna, N.: Automated verification of Telegram's MTProto 2.0 in the symbolic model. Comput. Secur. **126**, 103072 (2023). https://doi.org/10.1016/J.COSE.2022.103072
29. Mozipo, A.T., Acken, J.M.: Analysis of countermeasures against remote and local power side channel attacks using correlation power analysis. IEEE Trans. Dependable Secure Comput. **21**(6), 5128–5142 (2024)
30. Paier, M., van Eeden, R., Miculan, M.: Formal analysis of multi-factor authentication schemes in digital identity cards. In: Madeira, A., Knapp, A. (eds.) Software Engineering and Formal Methods - 22nd International Conference, SEFM 2024, Proceedings. Lecture Notes in Computer Science, vol. 15280, pp. 423–440. Springer (2024). https://doi.org/10.1007/978-3-031-77382-2_24
31. Paier, M., van Eeden, R., Miculan, M.: Formal Modelling and Verifying eIDAS Multi-Factor Authentication with Interface-Based Threat Analysis - Artefact (2025). https://doi.org/10.5281/zenodo.15403641
32. Sciarretta, G., Carbone, R., Ranise, S., Viganò, L.: Design, formal specification and analysis of multi-factor authentication solutions with a single sign-on experience. In: International Conference on Principles of Security and Trust. pp. 188–213. Springer (2018)
33. Sharif, A., Ranzi, M., Carbone, R., Sciarretta, G., Marino, F.A., Ranise, S.: The eIDAS regulation: a survey of technological trends for European electronic identity schemes. Appl. Sci. **12**(24), 12679 (2022)
34. Sinigaglia, F., Carbone, R., Costa, G., Zannone, N.: A survey on multi-factor authentication for online banking in the wild. Computers & Security **95** (2020)
35. Somorovsky, J., Heiderich, M., Jensen, M., Schwenk, J., Gruschka, N., Lo Iacono, L.: All your clouds are belong to us: security analysis of cloud management interfaces. In: Proceedings of the 3rd ACM workshop on Cloud computing security workshop. pp. 3–14 (2011)
36. Ullah, K., Rashid, I., Afzal, H., Iqbal, M.M.W., Bangash, Y.A., Abbas, H.: SS7 vulnerabilities-a survey and implementation of machine learning vs rule based filtering for detection of SS7 network attacks. IEEE Communications Surveys & Tutorials **22**(2), 1337–1371 (2020)
37. Wang, C., Wang, D., Tu, Y., Xu, G., Wang, H.: Understanding node capture attacks in user authentication schemes for wireless sensor networks. IEEE Trans. Dependable Secure Comput. **19**(1), 507–523 (2020)
38. Wang, D., He, D., Wang, P., Chu, C.H.: Anonymous two-factor authentication in distributed systems: Certain goals are beyond attainment. IEEE Trans. Dependable Secure Comput. **12**(4), 428–442 (2014)
39. Whitten, A., Tygar, J.D.: Why Johnny can't encrypt: A usability evaluation of PGP 5.0. In: USENIX security symposium. vol. 348, pp. 169–184 (1999)
40. Zhu, L., Wang, D.: Robust multi-factor authentication for WSNs with dynamic password recovery. IEEE Transactions on Information Forensics and Security (2024)

Matteo Paier is a PhD student of the Italian National PhD Program in Cybersecurity at the IMT School for Advanced Studies in Lucca, Italy. He received his Bachelor's degree in Computer Science from the University of Udine, Italy, in 2019, and his Master's Double Degree in Computer Science from the University of Udine and the University of Klagenfurt, Austria, in 2022. His research focuses on computer networks, cybersecurity, and formal models for the verification of protocols, in particular in heterogeneous systems (IoT, cloud, edge, etc.). He is also one of the founding members of MadrHacks, the University of Udine's ethical hacking team. He is actively involved in Capture-The-Flag competitions as both a player and organizer, aiming to promote cybersecurity education and innovation.

Roberto L.G. Van Eeden is a student in the Double M.Sc. Degree in Artificial Intelligence & Cybersecurity at the University of Udine and the University of Klagenfurt (Austria). He holds a Bachelor's degree in Computer Science from the University of Udine. His research interests include formal verification of protocols, software security, cryptography and embedded systems. Since 2024 he has joined MadrHacks, the University of Udine's CTF team, participating in Capture The Flag competitions alongside other members.

Marino Miculan (Ph.D. in Computer Science, University of Pisa, 1997) is a Full Professor of Computer Science at the University of Udine, Italy, where he leads the Laboratory of Models and Applications of Distributed Systems, and the local unit of the Italian National Cybersecurity Laboratory. His research interests include formal methods, concurrency theory, programming language semantics, type systems, and cybersecurity. He has (co-)authored more than 100 publications in international journals and conference proceedings. He has participated in many national and international research projects and regularly serves on the program committees of leading international conferences and workshops. Currently he is co-editor of the Concurrency Column of the Bulletin of EATCS.