applied sciences

Article

A Novel Protocol Using Captive Portals for FIDO2 Network Authentication †

Martiño Rivera-Dourado [1,2,*], Marcos Gestal [1,2,3], Alejandro Pazos [1,2,3,4] and Jose Vázquez-Naya [1,2]

1 Grupo RNASA-IMEDIR, Facultade de Informática, Universidade da Coruña, Campus de Elviña, 15071 A Coruña, Spain
2 Centro de Investigación CITIC, Universidade da Coruña, Campus de Elviña, 15071 A Coruña, Spain
3 IKERDATA S.L., ZITEK, University of Basque Country UPV/EHU, Rectorate Building, 48940 Leioa, Spain
4 Biomedical Research Institute of A Coruña (INIBIC), University Hospical Complex (CHUAC), 15006 A Coruña, Spain
* Correspondence: martino.rivera.dourado@udc.es

† This paper is an extended version of the paper published in the international conference: 2023 JNIC Cybersecurity Conference (JNIC), Vigo, Spain, 21–23 June 2023.

Abstract: FIDO2 authentication is starting to be applied in numerous web authentication services, aiming to replace passwords and their known vulnerabilities. However, this new authentication method has not been integrated yet with network authentication systems. In this paper, we introduce FIDO2CAP: FIDO2 Captive-portal Authentication Protocol. Our proposal describes a novel protocol for captive-portal network authentication using FIDO2 Authenticators as security keys and passkeys. For validating our proposal, we have developed a prototype of FIDO2CAP authentication in a mock scenario. Using this prototype, we performed a usability experiment with 15 real users. This work makes the first systematic approach for adapting network authentication to the new authentication paradigm relying on FIDO2 authentication.

Keywords: WebAuthn; network authentication; captive portal; protocol; FIDO2; security key; authenticators; passkey

check for updates

Citation: Rivera-Dourado, M.; Gestal, M.; Pazos, A.; Vázquez-Naya, J. A Novel Protocol Using Captive Portals for FIDO2 Network Authentication. *Appl. Sci.* **2024**, *14*, 3610. https://doi.org/10.3390/app14093610

Academic Editors: Zhe Xia, Mingwu Zhang and Lein Harn

Received: 20 February 2024
Revised: 19 April 2024
Accepted: 22 April 2024
Published: 24 April 2024

1. Introduction

The society often associates the concept of cybersecurity with passwords. They have been omnipresent in information systems all around the world for quite a long time. For a user, they are easy to use and understand and, without a doubt, passwords have become a well-accepted paradigm as an authentication standard.

1.1. Passwords and Their Vulnerabilities

Passwords as an authentication method have become vulnerable to numerous attacks [1]. The most famous is phishing, where an attacker misleads the victim to reveal their credentials. However, phishing is not the only existing threat that passwords are subject to. Keyloggers are hardware or software tools that attackers use to log the user input into a system. Some hardware keyloggers can be physically placed between the keyboard and the computer to log keystrokes that are directly sent to the attacker. Moreover, it is a reality that information leaks often appear in the news. These leaks can threaten passwords if they contain password hashes used by authentication servers [1]. Although an attacker is not able to directly use password hashes in the authentication portal, the password can be obtained from them with different techniques. Password-cracking techniques are based on automated trial and error methods for finding the original password.

In conclusion, attacks like phishing, password stealing, and cracking attacks entail a security risk for password-based authentication methods that leave the user unprotected and without control over their credentials. This makes the attacker able to access these systems and, therefore, to steal the information and data they hold.

1.2. FIDO2: A New Standard for Web Authentication

Considering that information systems protected by passwords are heavily threatened, some relevant technology companies such as Google, Microsoft, Meta, or Yubico have started to develop a new authentication method. Back in 2014, these companies created the FIDO Alliance [2], which has published two versions of the FIDO Client To Authenticator Protocol (CTAP) standard during the last few years: CTAP1, previously known as Universal Second Factor (U2F), and CTAP2. Since then, some security keys compliant with these standards have been developed, such as the Yubikey, the Solokey, or the Google Titan Security Keys.

In this context, in 2019, the W3C Consortium developed WebAuthn, a new standard compatible with these hardware security keys [3] and other software-based authenticators. WebAuthn is a new W3C standard that aims to complement or even replace passwords as a web authentication method. Relying on the FIDO standards, WebAuthn defines a browser API that has already been implemented in famous browsers such as Firefox and Chrome [4] (see Figure 1). In addition, it describes a protocol that allows its usage as an authentication method in web application servers. Consequently, some relevant web applications have already added WebAuthn security keys as an authentication method to be used together with passwords or even to replace them [5].

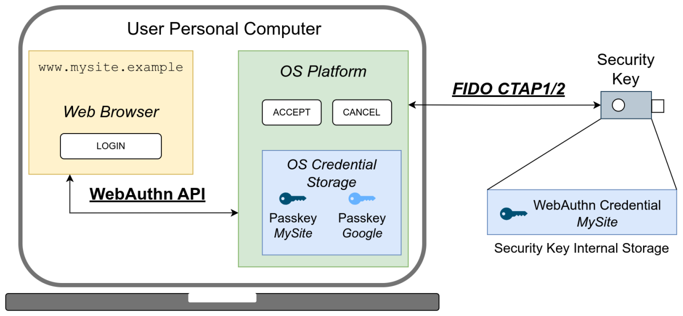


Figure 1. User personal computer compatible with FIDO2: W3C WebAuthn and FIDO CTAP standards. The web application interacts with the OS via the WebAuthn API of the web browser. The personal computer can have a compatible credential storage for WebAuthn credentials, or interact with a security key via FIDO CTAP1/2.

The adoption of WebAuthn has evolved during these few years. "Passkeys" are the new commercial name of WebAuthn credentials for passwordless authentication. During the last year, we have seen a great increase in passkeys, which increased by 50% from 2019 to 2022 according to a Duo report [6], for instance, Google [7], Microsoft and Apple accounts, some cloud provider portals like Amazon AWS, but also in social networks like Facebook and Twitter.

FIDO CTAP standards and the W3C WebAuthn standard form together a new authentication paradigm under the name of FIDO2 authentication, also known as the "passwordless" paradigm. It opens new possibilities for authentication mechanisms in multiple systems. Although the efforts were directed to web application user authentication, there are other computer systems where it could be integrated. One of them is authentication in computer networks, where access to the network or its resources are controlled.

1.3. Network Authentication

There are distinct ways to provide access control to a network. There are different systems whose aim is to authorise the end devices to send and receive traffic on the network. Wireless home personal networks can use different network authentication protocols for controlling connectivity. Although there has been an evolution in the protocols, from WEP to the last WPA3-Personal, they still rely on long-term passwords to authenticate users.

Usually, corporate networks have other solutions. The Extensible Authentication Protocol (EAP) with IEEE 802.1X can control the connectivity at the link level. This system is used by different systems, like the EDUROAM project. They can control access on Ethernet and Wi-Fi networks. However, like wireless home protection, the most common corporate EAP systems still rely on passwords. EAP methods such as EAP-MD5 and LEAAP or some tunneled methods based on PEAP or EAP-TTLS, like MSCHAPv2 or PAP, use password-based credentials.

1.4. Captive Portals

Apart from EAP, there exist other alternatives for access control to networks. Captive portal systems filter the traffic at the network or link layer to control the access to network resources [8]. Captive portals are websites displayed in end devices after network attachment to an Access Point (AP) before granting access to network resources or the Internet. These systems are typically found in hotels, airports or in shopping centers.

There are several ways to implement a captive portal. Most of the solutions nowadays are proprietary implementations delivered in business routers, but other open-source or custom router firmware offer this solution. For this work, we are interested in open-source alternatives for integrating WebAuthn authentication within the captive-portal system. The most well-known solutions are pfSense and OpenWRT. They both implement a captive-portal system, filtering access to network resources and displaying a captive-portal web page to the user. In our work, we will modify this captive-portal web page and integrate WebAuthn authentication in this system.

pfSense is one of the most popular open-source firmware available in the market, mainly developed by Netgate. Similar to other proprietary solutions, pfSense offers authentication based on local-configured accounts or an external RADIUS server [9]. Their firmware is compatible with several routers, including Netgate, but also others like some TP-Link and Protectli models.

On the other hand, OpenWRT is a Linux-based operating system that targets embedded systems [10]. This firmware can be installed in some routers from Lynksis, Asus, TP-Link, and Netgear, among others. OpenWRT has many available add-on packages that add features to the core light firmware. One of the available packages, built as a separate open-source project, is OpenNDS. OpenNDS, based on NoDogSplash, is captive-portal software developed in C compatible with this firmware. Among their options, it is worth mentioning that it adds the possibility to externalise the web server used in the captive portal. This configuration is known as the Forwarding Authentication Service (FAS) [11].

1.5. Authenticators: Security Keys and Platforms

FIDO security keys can be used in web authentication thanks to the WebAuthn API, developed by the W3C. These hardware devices are the original idea of having a dedicated hardware known as the authenticator to generate and store WebAuthn credentials. There are some commercial hardware security keys in the market, like the Yubikey and the Solokey, which can generate WebAuthn credentials.

Apart from hardware authenticators, there are operating systems that already support creating WebAuthn credentials, which are generated by a software known as a platform authenticator. These credentials can be stored in a secure storage at the operating system, or stored in some cloud secure storage. These are known as "passkeys". Passkeys are WebAuthn credentials created in compatible smartphones or laptops [7], and stored in a cloud provider to be synchronised among different devices of the user.

1.6. The WebAuthn Protocol and Types of Credentials

Both software-based passkeys and credentials generated in a security key (for simplicity, in this work we are also using the term *security key* when we refer to a WebAuthn credential generated and managed by a security key) are involved in the two main ceremonies of the protocol: registration (or attestation) and authentication (or assertion).

During registration, a credential is created and linked with a user account. This involves storing the created public key and the credential identifier in the authentication server. During authentication, the user can verify their identity using the previously registered credential. Specifically, they use the private key to sign a challenge, which is verified by the authentication server. This challenge–response mechanism of public-key cryptography ensures the authenticity and integrity of the authentication operation.

A FIDO authenticator generates an RSA or ECDSA key pair, forming a FIDO credential in a secure environment. In the case of hardware security keys, this is an independent cryptographic chip. Other devices may implement Trusted Execution Environments (TEEs) to implement secure cryptographic capabilities. A secure environment protects the private key in a secure storage to ensure its confidentiality and allow to perform all operations without the private key leaving the authenticator. During authentication, the challenge and other options enter the secure environment, where they are signed with the private key. After the operation, only the signature leaves the authenticator in the form of a response.

As said, a FIDO authenticator can generate these credentials during the registration ceremony. If the credential is stored into the hardware device memory, then the credential is known as a resident (or discoverable) credential. If the credential does not use the security key memory, then it is known as a non-resident (or non-discoverable) credential. For a user, the type of credential is almost transparent. However, from the development point of view, each type of credential involves a different registration and authentication flow. Also, not all security keys and operating systems support resident credentials. In this paper, we developed support for both types of credentials to increase the compatibility of the system.

1.7. Contributions

In this work, we designed FIDO2CAP: FIDO2 Captive-portal Authentication Protocol. Our proposal is to adapt FIDO2 authentication to the network authentication performed via a captive portal. Then, based on our previous work [12], we present a prototype of the OpenNDS captive portal using FIDO2CAP. Our system makes the authentication in captive portals resistant to common attacks for which password-based systems are vulnerable.

For validation, we installed our prototype in the networking hardware. We validated the compatibility with different operating systems and web browsers, and we conducted a usability experiment with 15 users.

Therefore, our contributions can be summarised as follows:

- **A novel protocol that integrates FIDO2 with captive-portal authentication: FIDO2CAP**. We provide a design of the FIDO2 Captive-portal Authentication Protocol (FIDO2CAP). We based our architecture design in the captive portal specification RFC 8952 [8], and then we specified the authentication and registration protocols.
- **Prototype of our protocol with OpenNDS**. We developed, tested and validated with 15 real users a prototype implementation of our protocol. For this, we used existing captive portal software known as OpenNDS.

1.8. Paper Organisation

The remaining sections of the paper are organised as follows. Firstly, Section 2 analyses the related work to our contribution. Section 3 describes the required software and hardware for this work, together with the methodology for all the research, implementation and validation tasks. Then, Section 4 shows the proposed FIDO2CAP protocol: both the architecture and the message flows. Then, we present a developed prototype of FIDO2CAP in Section 5. We validated the prototype as explained in the Section 5.4.2. All the results are presented in Section 6, which are discussed in Section 7. Finally, we draw some future research lines in Section 8.

2. Related Work

There exist several previous works that have addressed network authentication security improvement before, but there are few of them that have applied FIDO authenticators to network authentication.

2.1. Network Authentication with FIDO

In 2018, Chifor [13] presented a solution that makes use of the FIDO UAF protocol, a previous version of the current FIDO CTAP protocol, to network authentication. In this work, the protocol is used for authenticating guest users in an enterprise network by employing their personal Android smartphones. For this purpose, they created a scheme where guests can register themselves to be authenticated in a different Wi-Fi network.

FIDO protocols were also applied to IoT network authentication. Also in 2018, Chifor proposed a security authorisation scheme for IoT devices, which interacts with the user Android smartphone to perform FIDO CTAP1 authentication [14]. Then, in 2021, Luo et al. [15] used FIDO CTAP1 security keys to provide user authentication in an IoT Smart Home environment. In their work, a gateway node managing IoT devices authenticates a user when performing management operations. This authentication is performed with FIDO CTAP1 security keys physically connected to the gateway node.

Furthermore, it is worth mentioning the work of Huseynov [16], who proposed in 2022 a solution for connecting to a VPN service by authenticating the user with FIDO security keys. Their approach is to create an intermediary web portal that provides a temporal username and password pair after authenticating with security keys. Their approach is a wrapper solution to the problem, as the authentication of the VPN service is still based on the existing compatible authentication method of the VPN software.

On the other hand, it is interesting how captive portals have been adapted to additional network authentication scenarios. Marquest et al. [17] designed a custom EAP method named EAP for Secure Hotspots (EAP-SH) that adapts the web-based authentication of captive portals to be used in IEEE 802.1X access controls. The work of Marquest et al. [17] can be adapted to use the work of this paper with EAP technology.

2.2. Captive Portal Authentication

Despite the numerous security problems of captive portals [18], there have been improvements in this field. The RFC 8910 [19] published in 2020 adds a DHCP code to improve the captive-portal detection. In wireless networks, the appearance of WPA3, a new Wi-Fi Alliance standard, adds encryption to open Wi-Fi networks with the Opportunistic Wireless Encryption (OWE) protocol [20]. This feature mitigates eavesdropping attacks on open wireless networks and, therefore, makes them more secure to use a captive portal on them.

Captive-portal authentication systems rely on web authentication technology. Apart from the well-known passwords, there are other identity verification factors that can be used in the web. Multi-factor authentication (MFA) implies the use of more than one method to verify the identity of a user. Online systems typically use a knowledge-based factor such as a password or a PIN. Different strategies include possession-based authentication [21]. For example, authentication based on digital certificates stored on smart cards or USB keys can verify the identity of users in online accounts. Other methods, usually used in MFA, are One-Time Password (OTP) codes. These can be generated by a specific device, generated via a software application, sent via SMS or via push notifications to the user.

In this work, we integrate captive portals with a trending technology for web authentication, based on asymmetric cryptography: FIDO2 credentials. This method can be classified as possession-based authentication, as FIDO2 credentials are stored within a device that the user owns. This can be a personal computer, a smartphone or a FIDO2 security token.

3. Materials and Methods

The main objective of this work was to design, develop and validate the integration of FIDO2 authentication with a captive portal, providing a usable system for network authentication. After the FIDO2CAP protocol design, we developed a prototype. Firstly, we developed an authentication web server and, secondly, we integrated it with the OpenNDS captive-portal software. Also, for improving usability, our work involved a study of the exceptions issued by different operating systems, together with a user interface design. Finally, the result was validated with a compatibility test and a usability experiment with real users. This section describes the materials and methods used during all these phases of this work.

3.1. Required Materials

Considering all the involved parties in this development authentication and authorisation system, the environment used during this development tasks is based in Virtual Machines (VMs). Also, we describe the networking hardware used during validation. Additionally, for working with WebAuthn authentication, we also need security keys and a compatible technology stack, namely, the following:

- Virtualisation software. We used *VirtualBox 6.1* for setting up the virtual environment for development.
- Three security keys. For the development of the tool and the validation phase, we used the Yubico Yubikey Security Key, which is compatible with discoverable credentials.
- Compatible technology stack. That is, compatible operating systems and browsers. We used Linux (Manjaro and Ubuntu), macOS Monterey, Windows (10 and 11), Android (v8, v12 and v13), and iOS 16.
- Hardware wireless router compatible with OpenWRT. We used ASUS RT-AC1200.

3.2. Work Methodology and Planning

This research, development and validation work was planned in six main phases, namely, (PH1) initial research, for analysing captive-portal implementations, WebAuthn technology, and configuring the environment; (PH2) protocol design, for grouping the main requirements; (PH3) authentication server development, to develop the web application with WebAuthn authentication; (PH4) system integration, where the developed web application is integrated with the captive-portal authorisation; (PH5) usability improvement, where the error handling is designed and user interface is improved; and (PH6) validation, where the result is validated through compatibility tests and a usability experiment with real users.

3.2.1. Initial Research

The initial research includes two main technology stacks: (1) FIDO2: WebAuthn and FIDO authentication; and (2) network authentication with captive portals. Apart from the scientific literature review, this research also included informal tutorials published in technology blogs, open-source projects, and market-related technology solutions. All these information sources together enrich the research in these technologies, which is continually evolving.

3.2.2. Development and Integration

When designing the proposed solution, the authors considered two main phases for the development: first, a web authentication server which implements WebAuthn registration and the authentication of security keys and passkeys; and second, the integration of WebAuthn authentication with the captive-portal authorisation mechanism. These steps correspond to the phases PH3 and PH4, respectively.

3.2.3. Usability Improvement

For improving usability, the user interface was redesigned after a first test with a few users. Also, we noticed that messages displayed originated from different use errors were not user friendly. Therefore, we identified many possible errors during authentication and described the exceptions issued by web browsers in each operating system to improve error messages accordingly.

3.2.4. Validation

Finally, we measured system usability by (1) studying the compatibility across the most common operating systems and browsers, (2) a usability experiment with real users. The compatibility test was tested using the hardware deployment and two already registered security keys, one with discoverable and other with non-discoverable credentials. In relation with the usability experiment, we designed an experiment where the subjects were given the task to connect and authenticate in the developed, Wi-Fi protected captive portal. Users were observed to identify all use and technology errors, while measuring the time and reactions. After the task, they were asked some questions to measure their satisfaction. Finally, users were asked to repeat the task and to complete a final questionnaire. We considered usability measures defined by ISO 9241: completeness, efficiency and satisfaction.

4. Fido2cap: FIDO2 Captive-Portal Authentication Protocol

In this paper, we propose the FIDO2 Captive-portal Authentication Protocol (FIDO2CAP), a captive-portal network authentication protocol using FIDO2. This section includes the theoretical design of the protocol. First is the architecture design based on the captive-portal architecture defined by RFC 8952 [8]. Then, we have the message flow of the FIDO2CAP protocol.

4.1. Architecture Design

RFC 8952 [8] describes the general architecture of a captive portal in a network. In this work, we adapt and extend this architecture for enabling network authentication using FIDO2 Authenticators. This solution can be implemented to provide access control for users in an access-level network, such a Wi-Fi network or a cabled network of end devices.

Figure 2 shows the final architecture diagram, highlighting our proposed changes. In our architecture, we include several elements.

- **FIDO2 Authenticator**. The user FIDO2 Authenticator can be a hardware security key, a platform authenticator, or a FIDO2 multi-device credential (passkey). The user interacts with the FIDO2 Authenticator to prove their identity when accessing a network. A platform authenticator may be included in the User Equipment. For describing the general architecture, we consider it a separate device communicating via the FIDO CTAP protocol.
- **Web browser with WebAuthn API**. The User Equipment must have a compatible client for performing authentication through the W3C WebAuthn API.
- **WebAuthn Authentication Web Application (WAWA)**. A WebAuthn Relying Party is implemented as a web-based application, which works as the User Portal displayed by the captive-portal system. This includes a web interface displayed at the compatible web browser, and the authentication server.
- **User Database**. A database that stores the FIDO2 credentials linked to a user identity.
- **Session database (optional)**. A database of opened sessions, which can allow the user or the administrator to list all open sessions linked with a user identity.
- **Registration portal**. The registration system allows a registrar to register the FIDO2 Authenticator of the user. This portal may also show the active sessions linked with a user and an authenticator. Alternatively, the registration portal may provide a self-registration flow for a user to register themselves. For instance, this self-registration can be controlled by a temporal access token, shared as a QR code.
- **Registration Equipment**. The Registration Equipment and the User Equipment may be different, depending on the registration flow, for example, if the registration process is performed by an administrator, or there is a setup equipment for self-registration.

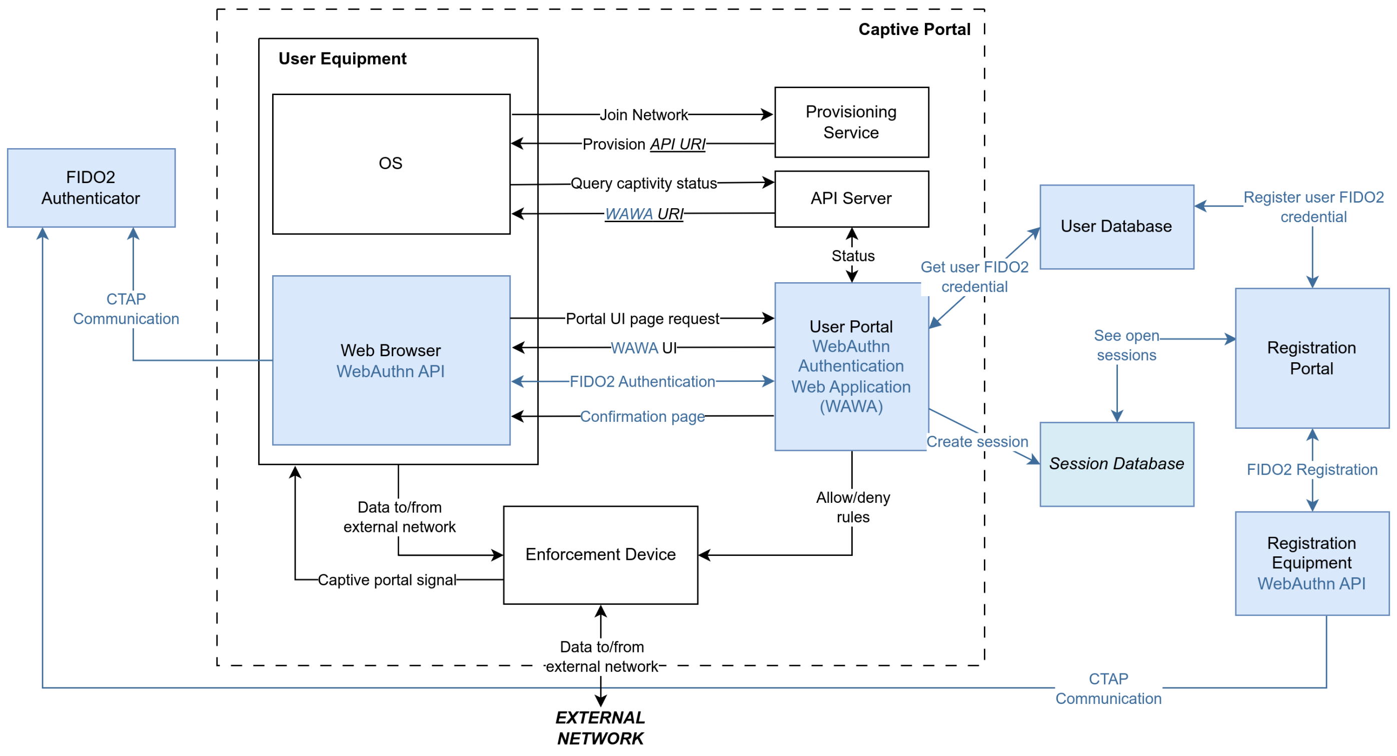


Figure 2. Architecture of the FIDO2 Captive-portal Authentication (FIDO2CAP) Protocol.

The proposed system architecture works similarly to the RFC 8952 [8] for network authentication. After joining the network, the captive portal intercepts the traffic and provisions the WAWA URI. Then, the operating system of the User Equipment displays the WAWA user interface to the user. The WebAuthn Authentication Web Application now starts the FIDO2 authentication process, involving the user and the FIDO2 Authenticator via the WebAuthn API and the FIDO CTAP protocol. If the authentication is successful, the Enforcement Device is instructed to allow access to the External Network for the User Equipment. This authorises the user to access the protected network.

4.2. Protocol Overview

The FIDO2CAP (FIDO2 Captive-portal Authentication Protocol) describes two main ceremonies, following the FIDO2 standards. Firstly, we describe the user authentication protocol, based on the proposed architecture. Then, we propose a FIDO2 registration flow via the registration portal.

4.2.1. Registration Scenarios: Administrator or Self-Registration

We consider two main actors in the protocol: a user and a registrar. Also, we consider a different equipment for authentication (User Equipment) and for registration (Registration Equipment). This provides a flexible range of environments. For instance, the system may be enabled for user self-registration. The user and the registrar may be the same person. In this case, the Registration Portal is accessed by a temporal access token for user self-registration, which may be shared as a QR code to the user.This would allow the user to use the User Equipment for registration, which enables the registration of user FIDO2 platform authenticators. For example, a user may scan a QR code provided by an administrator, and use their Android smartphone to register a passkey in the system.

4.2.2. FIDO2 Credentials: Discoverable and Non-Discoverable

As mentioned in Section 1.6, FIDO2 credentials can be discoverable (resident) or non-discoverable (non-resident). This changes how the authentication protocol works. With non-discoverable credentials, the user must be identified in the first place, while discoverable credentials do not need this step. When the user is identified (for instance, using a username), the list of allowed credentials is sent to the FIDO2 Authenticator. As non-discoverable credentials are not resident (stored) at the authenticator, they need to be provided in the allowed credentials list during the authentication protocol. Our definition of the FIDO2CAP protocol is compatible with discoverable and non-discoverable credentials, as it includes the identification of the user. If discoverable credentials are used, then this step during authentication can be omitted.

4.3. Authentication Protocol

Figure 3 shows the FIDO2CAP authentication protocol message flow. The parts highlighted in blue are modified or added to the RFC [8]. The FIDO2CAP protocol starts with the captivity signalling from the router. When the User Equipment requests the captivity status, the captive-portal API Server replies with the WAWA URI. This opens a web browser in the User Equipment, visiting the WAWA user interface. Here, the user sends the username for identification (this is the optional step with discoverable credentials, see Section 4.2.2).

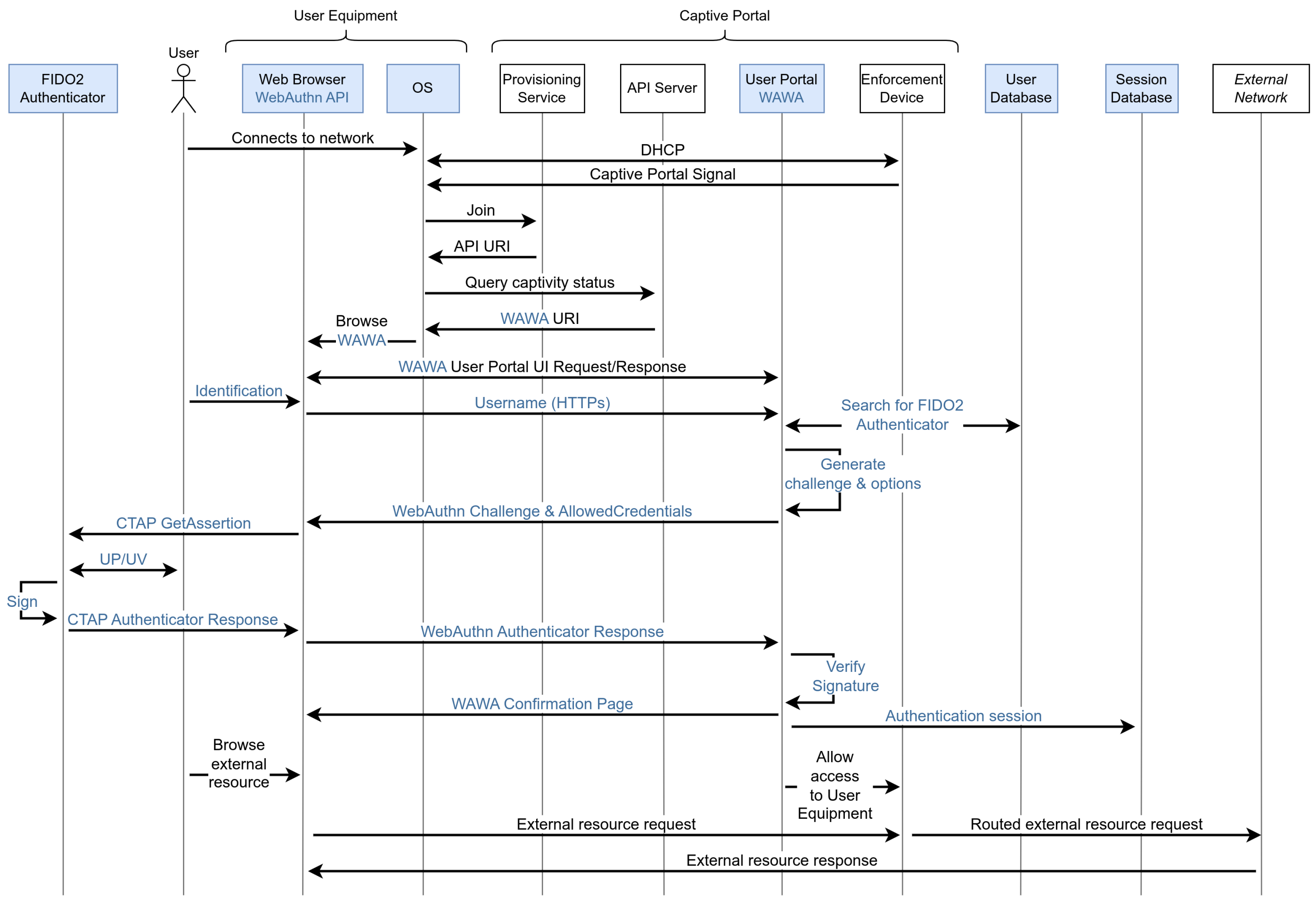


Figure 3. Authentication message flow in FIDO2CAP. The User Equipment is connected to the captive portal, getting redirected to the WAWA server URI. The user performs FIDO2 authentication at the WAWA User Portal UI. After verifying the authentication, the Enforcement Device allows access to the User Equipment, ending captivity.

The username allows the WAWA application to search for registered FIDO2 Authenticators linked with the user in the User Database. With this information, the WAWA server generates the Allowed Credentials list, and the WebAuthn challenge, which is sent to the Web Browser. In the user end, the browser initiates the WebAuthn API, which interacts with the authenticator via the OS CTAP GetCredentials call. After the user interacts with the authenticator via User Presence (UP) and/or User Verification (UV), the FIDO2 Authenticator signs the challenge and sends its response, which is forwarded to the WAWA server. Finally, the response is verified, and the WAWA server notifies the Enforcement Device, allowing the User Equipment to access the external resources, ending the captivity.

4.4. Registration Protocol

The FIDO2CAP registration protocol can be instantiated with different setups. Figure 4 shows the general registration process, which involves the WAWA Registration Portal, a different user interface of the WAWA server. The registrar, which may be the user, sends a request to the Registration Portal. After verifying that the registrar has the required registration rights (see Section 4.2.1), the registrar sends the user username. Then, the username is used to search for registered FIDO2 Authenticators in the User Database. These are be included in the Excluded Credentials list, together with the registration challenge generated by the WAWA server. This triggers the WebAuthn API that interacts with the user FIDO2 Authenticator via the CTAP MakeCredential command. Once requested, the registrar authorises the registration in the authenticator, which generates the response. This response is forwarded to the Registration Portal, which verifies it. Then, if the user is already registered, the credential is added to the allowed FIDO2 Authenticators. If the user is not present in the database, it is registered with the new credential.

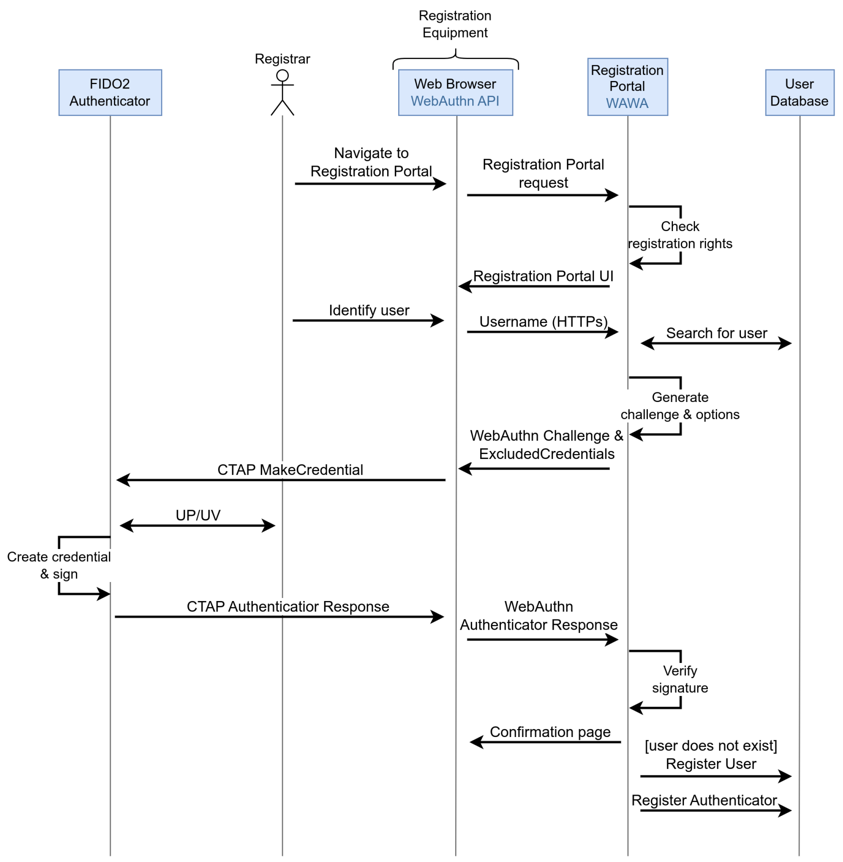


Figure 4. Registration message flow in FIDO2CAP. The Registrar has the required permissions to access the Registration Portal UI. After specifying the username, FIDO2 registration starts. Finally, the FIDO2 credentials (referred as Authenticator) are registered and linked with the user account.

4.5. Threat Model and Security Assumptions

In FIDO2CAP, the architecture of a captive portal is adapted to use FIDO2 authentication. It is important to highlight that we do not introduce any new security element not present in a captive portal or a FIDO2 authentication service. The main components of the authentication system are the User Equipment, the User Portal and the FIDO2 Authenticator. Therefore, the security assumptions and threat analysis are the same as in FIDO2, described in the FIDO Security Reference [22]. Further, in FIDO2CAP, we assume that the communication between the different captive-portal elements of the architecture is trustworthy, such as between the User Portal and the Enforcement Device, or between the User Portal and the User Database.

In this paper, we focus on preventing attacks during authentication by adapting captive portals to use FIDO2 authentication. Attacker capabilities include (1) impersonating the Access Point (AP) of the network, (2) listening to the channel between the client and the User Portal, and (3) performing online brute-forcing attacks towards the User Portal. A captive-portal authentication system relying on a password is vulnerable to all these attacks. With FIDO2CAP, the authentication relies in asymmetric cryptography to proof the possession of the FIDO2 credentials, without sending them. Also, in FIDO2, the web browser includes security checks on the authenticity of the User Portal during authentication. FIDO2 credentials only work with the WAWA server in which they have been registered. Finally, online brute-forcing attacks towards public-key cryptography authentication currently not proved to be a threat.

In this threat model, we should take into account other types of attacks [18]. First, other network-based attacks may take advantage of vulnerabilities of the Enforcement Device. These may differ depending on the nature of the network in which FIDO2CAP is implemented. For instance, an attacker could impersonate authenticated User Equipment to bypass the Enforcement Device by a spoofing attack. On the other hand, an attacker may gain access to the Registration Portal, allowing them to register their own FIDO2 Authenticator within the network. Therefore, registration should be protected by temporal authorisation tokens or by administrator credentials.

5. Prototype Development and Validation

As a Proof of Concept (PoC), we developed and validated a prototype of the proposed FIDO2CAP protocol. In this section, we present the developed prototype: a captive-portal authentication system using FIDO2CAP. First, we explain the WAWA server implementation (Section 5.1), and then the integration with the OpenNDS captive-portal software (Section 5.2).

Our prototype follows the FIDO2CAP architecture (see Figure 2). We implemented the WAWA server, using a single database as the User Database and Session Database. This web application includes both a User Portal and a Registration Portal, which implement the WebAuthn functionality. Finally, we integrated our WAWA implementation with OpenNDS. OpenNDS implements a captive portal: the Enforcement Device, the Provisioning Service and the API Server.

5.1. WebAuthn Authentication Web Application (WAWA)

In this section, we describe the WebAuthn Authentication Web Application (WAWA) that allows (1) the registration of WebAuthn credentials by the administrator and (2) the authentication with a WebAuthn credential by the user. The following sections include relevant details on the development, which was based in NodeJS and the SimpleWebAuthn library [23].

5.1.1. WebAuthn Standard and Credentials

As explained in Section 1.6, the WebAuthn standard considers the registration and authentication of credentials. There are two types of WebAuthn credentials related to authenticators (like security keys): discoverable and non-discoverable. For this purpose, we have to consider two different authentication flows.

With discoverable credentials, the user does not need to identify themselves with a `username` during the process. The `userHandle` included in the authenticator response is used as the `userId` to find the registered device for verifying the authenticator response.

Hence, using discoverable WebAuthn credentials implies that the user is identified *after* the authentication of the credential.

On the contrary, with non-discoverable credentials, the authentication response from the authenticator will not contain a `userHandle` that identifies the user. Therefore, the `username` should be specified at the authentication form. When the server generates the WebAuthn authentication options, a list of the registered credentials for the corresponding user must be provided.

5.1.2. Administration Interface

The registration of credentials in the web application is restricted to administration. Using a RBAC, whether the users are authorised to register new devices or see restricted information depends on their role. For this purpose, we created a separate administrator interface. The interface is restricted to users with the admin (or registrar) role, which authorises a user to (1) check registered users, their active sessions and registered credentials; (2) register a WebAuthn credential; and (3) assign administration roles to users.

5.2. Captive Portal Integration with OpenNDS

Once we developed a first working version of the WAWA, it was integrated with captive-portal software: OpenNDS (v9.10.0). This software implements the Provisioning Service, the API Server and the Enforcement Device. First, we explain why we have chosen OpenNDS for the prototype development. Secondly, we explain the integration of the WAWA with the API Server and the Enforcement Device of OpenNDS.

OpenNDS feature for externalising the User Portal is called Forwarding Authentication Service (FAS) [11], which forwards the redirected requests to a captive portal to the selected external web server. Finally, the OpenNDS Enforcement Device should be integrated with the external web server to identify the authenticated client and authorise its access to the network. Therefore, during this section, the developed WebAuthn Authentication Web Application (WAWA) server was modified to implement an external FAS server compatible with OpenNDS.

During this section, the captive portal is commonly referenced with the chosen solution name: OpenNDS. On the other hand, our WAWA server is commonly referenced as the FAS server (Forwarding Authentication Service), nomenclature used in the OpenNDS configuration for the User Portal server.

5.2.1. OpenNDS Authmon: API Server and Enforcement Device

OpenNDS has a module named Authmon that implements an API Server and an Enforcement Device. As implemented in OpenNDS, the Authmon module sends periodic requests to the FAS server. In this prototype, the FAS server is our WAWA server, which we have made compatible by extending the WAWA API, accepting the requests from Authmon.

The Authmon module of OpenNDS sends periodic requests to the FAS server to poll for new authenticated clients. Figure 5 shows a flow diagram that represents the final integration of the Authmon operation to authorise clients with the developed FAS server. It is worth mentioning that this integration was possible after the reverse engineering process due to the lack of detailed documentation.

Following Figure 5, (1–3) the captive portal run by OpenNDS redirects unauthorised clients to the WAWA server URL, together with a hashed id (`hid`), encoded and encrypted as explained in Section 5.2.3; (4–5) when handling User Equipment requests, the WAWA server must decrypt these details and calculate the authorisation token `rhid` as explained in Section 5.2.3; (6–9) when the FIDO2 authentication is successful, the WAWA server marks the client as authenticated; (11) all authorisation codes (the `rhid`) are requested periodically to the WAWA (FAS) server, who returns a list of authorisation tokens is sent to the OpenNDS router; and (12–14) using this token, OpenNDS can authorise the corresponding client for their access to network resources, notifying WAWA about this.

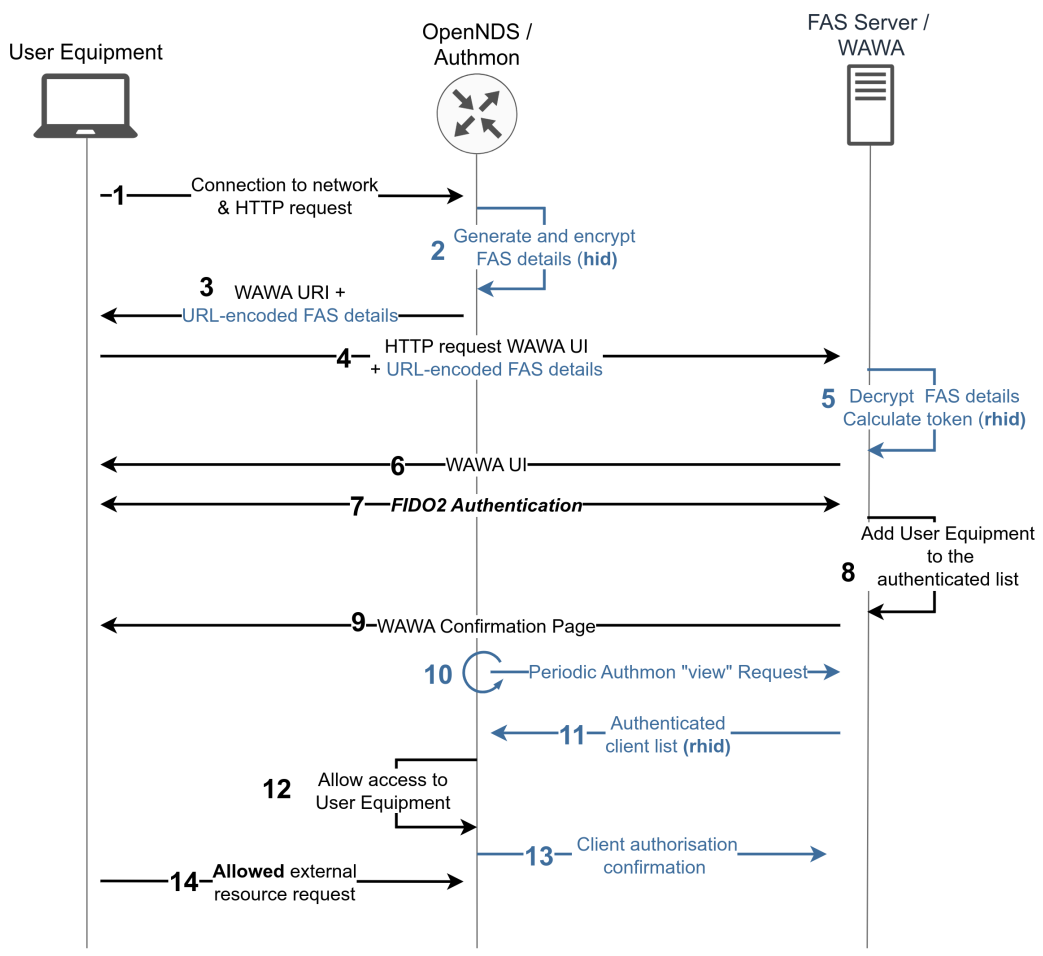

User Equipment
OpenNDS / Authmon
FAS Server / WAWA
1 Connection to network & HTTP request
2 Generate and encrypt FAS details (hid)
3 WAWA URI + URL-encoded FAS details
4 HTTP request WAWA UI + URL-encoded FAS details
5 Decrypt FAS details Calculate token (rhid)
6 WAWA UI
7 FIDO2 Authentication
8 Add User Equipment to the authenticated list
9 WAWA Confirmation Page
10 Periodic Authmon "view" Request
11 Authenticated client list (rhid)
12 Allow access to User Equipment
13 Client authorisation confirmation
14 Allowed external resource request

Figure 5. Communication between OpenNDS Authmon module and the developed WAWA server (FAS server). The User Equipment is redirected to the WAWA UI hosted at the FAS server and, once authenticated, it is authorised by OpenNDS. In blue, we have the particular operation of OpenNDS we considered for the integration of the developed WAWA server.

5.2.2. New API Endpoints For Authmon

Thanks to the reverse engineering, the WAWA REST interface was adapted to handle the Authmon periodic requests and the new User Equipment (client) requests. According to the reverse engineering process, the OpenNDS Authmon module can issue three HTTP POST requests to the authentication WAWA (FAS) server. These are distinguished by the `auth_get` parameter of the HTTP request body.

- **"clear"**: All authenticated clients should be clear from the list. That is, the list is reset. This is used by OpenNDS when booting.
- **"list"**: The list should be sent and cleared. This type of request is not frequent and is kept for backwards compatibility.
- **"view"**: The most frequent request depends on its payload:
  - *** or none**: The complete list of authenticated clients is required by Authmon. The corresponding authorisation tokens (or `rhid`) should be sent in a list, according to the compatible format: `* <rhid>`.
  - *** < rhid >**: Authmon is confirming the authorisation of a client.

To implement the list of authenticated clients, the session management feature developed in the WAWA server can be used. The server incorporates a separate database collection for storing authenticated user sessions. Then, once OpenNDS confirms the client authorisation, it is marked as authorised. Finally, the session expiration time is synchronised with the OpenNDS expiration time.

5.2.3. Cryptographic Notes on FAS in OpenNDS

For the implementation of the integration, there are two cryptographic particular notes that should be considered: (1) the initial encrypted details, and (2) the authorisation token of the OpenNDS captive portal.

Steps 3 to 6 of Figure 5 show how the client traffic is redirected to the captive portal at the FAS Server. OpenNDS adds some parameters necessary for the later client authorisation, encoded in base64 and encrypted with AES-256-CBC. Therefore, the WebAuthn authentication server installed at the FAS Server should decode and decrypt these parameters. Specifically, the AES block cipher in the Cipher-Block-Chaining (CBC) mode used by OpenNDS has a 256-bit block length. In the WebAuthn authentication server in NodeJS, the required key length is 32 bytes, which is shared with OpenNDS.

On the other hand, step 5 of Figure 5 shows the calculation of the authenticated hash performed by OpenNDS. This authenticated hash has to be used by the FAS server to send an authorisation token that allows a client to be authorised by OpenNDS. The hash used by OpenNDS is SHA256. However, in order to authenticate the hash, OpenNDS developers have chosen to include the symmetric key at the end of the payload to hash. In this case, the FAS server should return a re-hashed version of the `hid` parameter when the client is authenticated. Letting k be the 32 byte shared key, the operation is shown in Equation (1):

$$rhid = sha256\,(\,hid\,||\,k\,) \tag{1}$$

5.3. Test Deployment and Improvements

This section depicts the environment setup of the developed prototype: the WAWA server integrated with OpenNDS. The environment presented here is the one used in this work to perform development and validation of the prototype. Our setup provisions a Wi-Fi connection via a WPA3 open network. This network is protected with our prototype captive-portal solution. It only provides Internet connectivity after successful FIDO2 user authentication.

5.3.1. Environment Configuration of a Mock Use Case: Wi-Fi Connectivity in a Hotel

Our environment setup can be summarised in four pieces of equipment, mocking a use case of a hotel providing connectivity to the hotel network for their hosts. The FIDO2 security keys are registered beforehand, and they are provided to the clients. As shown in Figures 6 and 7, the setup consists in the following:

1. The **User Equipment** and the **FIDO2 security keys**. They are the personal devices of the hotel clients, used to connect to the Wi-Fi for Internet connectivity.
2. The **Registration Equipment** for the administrator. Using this device, the hotel personnel register security keys for their clients.
3. The developed **WAWA server**. Running in a physical server, the WAWA application is configured properly to work with the OpenNDS instance.
4. The **OpenWRT router running OpenNDS**. For validating our solution with real hardware, we used the ASUS RT-AC1200 router, which is compatible with OpenWRT. OpenNDS should be properly configured to work with the WAWA.

The environment was configured by installing the OpenWRT firmware in the ASUS router, using the firmware restoration tool. We configured a WAN interface as a DHCP client for providing Internet connection to the router through another network. Also, the client network was configured using Wi-Fi with the WPA3 open network.

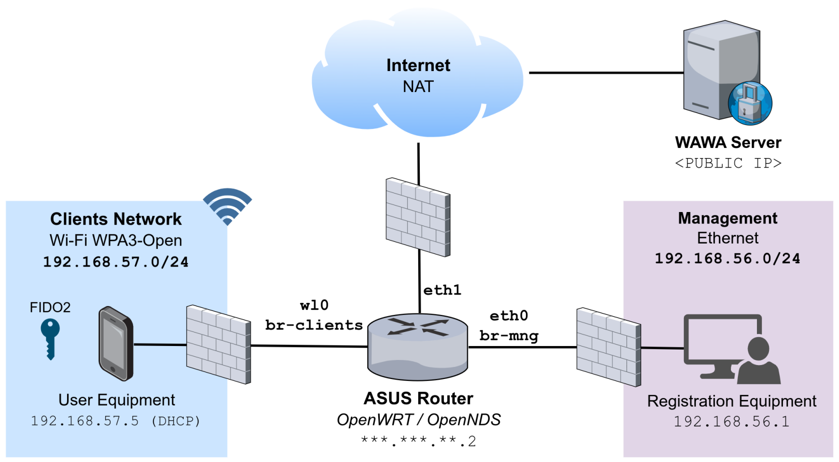


Figure 6. Example environment diagram: Wi-Fi connectivity in a hotel using the developed prototype. The network is divided in three zones: (1) a client wireless network; (2) a management network; and (3) Internet through NAT. The WAWA server is placed in a public server for simplicity. Clients have restricted access to Internet until successful FIDO2CAP authentication.

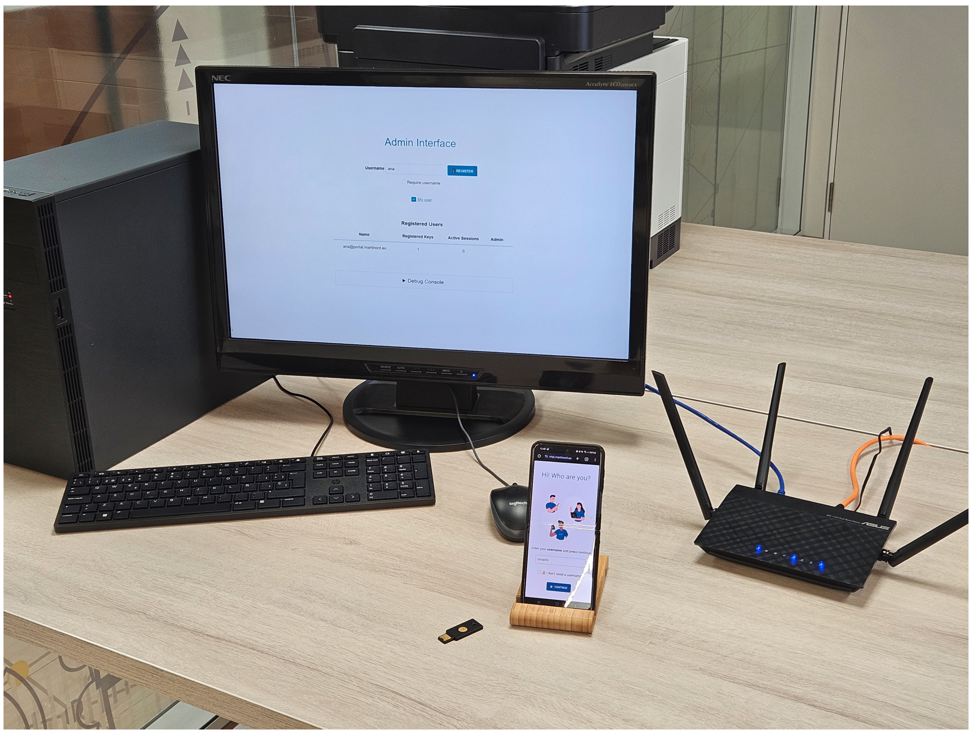


Figure 7. Physical deployment of the example environment. From left to right: the Registration Equipment (Windows 10), the FIDO2 Authenticator (Yubikey Security Key), the User Equipment (Android 13), and the ASUS Router (OpenWRT). The WAWA server is publicly hosted at our servers (running the prototype [24]).

Moreover, as shown in Figure 6, the example topology divides the corporate networking into three networks attached to the OpenWRT router, namely, (1) Internet, (2) management, and (3) clients network. For simplicity, our WAWA server is placed in Internet, although it could be in a DMZ within the corporate network. Following our mock example of a hotel network, the clients are restricted to access Internet and some hotel services until successful connection to the Wi-Fi network. This provides the proper segmentation of the network, and access control to clients via FIDO2CAP authentication.

Finally, as mentioned before, configuring the captive portal in the router relies in OpenNDS configuration with the WAWA server. As said, in our example environment, the

WAWA server is deployed in a public domain with a specific unique IP address. A unique IP address is necessary so OpenNDS can configure the Enforcement Device (or firewall) properly to allow the User Equipment initial request for the WAWA UI. Also, we apply compatible configuration of the server and the router, so the router can serve as a gateway of the authentication server. The configuration steps are included in Github [24].

5.3.2. Improvements in Usability and Compatibility of the Prototype

During the deployment and initial testing of the prototype, we found some usability and compatibility issues. Here, we explain some of them, and in the following section, we include the final system validation and usability experiment.

During FIDO2 authentication, there are several errors that can occur, so the user should be notified. However, we found that the user is not properly notified of all errors through their web browser. This decreases the usability of the authentication flow UI. In fact, the browsers raise different browser exceptions upon the same error. For instance, Firefox and Chrome in Android differ when the user disconnects a FIDO2 security key during authentication. At the time of writing, Linux raises the "Unknown Error" exception, while the Chrome browser raises "Not Readable Error".

For this reason, we are conducting a study of all these differences, and finding some guidance, which will be published in the near future. In the prototype proposed in this paper, we addressed some of these issues. Some of the errors display a non-intrusive message, like a "Try again" button, without identifying the error and allowing the user to repeat the action. Other errors that occur when the operation is cancelled include this information to the user, allowing them to have another try.

5.4. Prototype Validation and Usability

A captive portal using FIDO2 authentication is a novel approach. For this reason, we have measured both software compatibility with real devices and system usability with real users of the developed prototype. This section describes both studies.

5.4.1. Captive Portal with WebAuthn Compatibility Test

Our captive-portal prototype was tested in several desktop and mobile operating systems. Taking into account the most used operating systems nowadays [25,26], we conducted some specific compatibility tests for validating the solution. We tested Windows, macOS and Linux desktop OSs, and Android, iOS and Samsung One UI mobile OSs. Namely, the following tests were designed:

- NET-1 (CPD): The captive portal is detected by the OS and a browser is launched.
- NET-2 (INTERNET): There is internet connection after successful captive-portal authentication.
- WA-1 (WEBAUTHN): The browser launched by the OS is compatible with WebAuthn, and the API exists.
- WA-2 (NDC): The browser was able to successfully use non-discoverable credentials.
- WA-3 (DC): The browser was able to successfully use discoverable credentials.
- WB-1 (REDIRECT): In the case of incompatibility of the default browser for captive-portal detection, the user is correctly redirected to the other browser.
- WB-2 (EXCEPTION): Exception is issued when the default browser has no practical WebAuthn compatibility.

Table 1 shows the test results. The tests were performed with the ASUS router running the captive portal. We registered two FIDO2 security keys, one with discoverable credentials and other with non-discoverable credentials. As shown in the table, there are different operating systems that detect the captive portal and launch a specific mini-browser that usually is not compatible with WebAuthn. As analysed in [27], these captive-portal mini-browsers have limited capabilities, and even security issues. These browsers are used in smartphone OSs and Linux-based desktop OSs, which affect to our system compatibility.

Table 1. Compatibility of the captive-portal solution across different operating systems.

Test / OS	**Windows**		**macOS**	**Manjaro**		**Ubuntu**	**Android**			**iOS**
	10	**11**	**Monterey 12.6.7**	**KDE 23.0.1**	**GNOME 23.0.1**	**22.04.3 LTS**	**8**	**12**	**13**	**16**
NET-1	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES
NET-2	YES	YES		YES	YES	YES	YES	YES	YES	
WA-1	YES	YES	NO	YES	NO	NO	NO	NO	NO	NO
WA-2	YES	YES		YES						
WA-2	YES	YES		YES						
WB-1			NO [4]		NO [3]	NO [3]	NO [1]	YES [2]	YES [2]	YES
WB-2			NotAllowedError							NotAllowedError

[1] Android web browser menu has the option to "open in the web browser", which allows the user to choose the alternative. [2] Before opening the link in an external app, it asks for confirmation by the user with a warning message. [3] The user should manually copy the URL or open another browser manually. [4] There is no option for the user to open the URL in another browser manually.

5.4.2. Usability of the Captive Portal with Security Keys in Mobile Devices

As part of the validation of the solution, we designed and conducted a usability experiment with 15 subjects. For this experiment, subjects used their own smartphone devices, which they are used to, as clients of the Wi-Fi network running our captive portal. This follows the mock environment described in Section 5.3.

Users were asked to connect to the Wi-Fi network and authenticate with the security key and a username, after a brief explanation. The security key and username were already registered in the system by an administrator. After the first try, the subjects were asked about their satisfaction and completeness of the process. Then, they were asked to perform the task two more times and finally complete a small questionnaire.

For measuring the usability of the solution, we considered three different measures based on ISO 9241 [28]: effectiveness, completeness and efficiency:

- Effectiveness "is the accuracy and completeness with which users achieve specified goals", that is, that complete the connection to the Wi-Fi network with the security keys. Also, we measure the error rate while performing the task by observing the difficulties or technical issues that the user encounters. The error rate measures the average number of errors per performed task, calculated as the division of total error occurrences by the number of tasks.
- Efficiency "is the resources used in relation to the results achieved" that we measure as Time-Based Efficiency and Overall Relative Efficiency.
- Satisfaction "is the extent to which the user's physical, cognitive and emotional responses that result from use of a system, product or service meet user's needs and expectations". We measured satisfaction with a custom questionnaire after performing the task.

5.4.3. Usability Experiment Subjects and Devices

In this experiment, 15 users whose ages vary from 18 to 64 participated, including men and women. All of them use technology on a daily basis, and half of them have high or expert knowledge about computer science. Also, although 93% use second-factor authentication, only 20% have tried security keys before. For this reason, one of the hypotheses of the experiment is that the first contact with the proposed system will delay the completion of the task.

The users have used their own smartphone devices, 13 of them Android and 2 iOS. We should highlight that iOS devices are not compatible with the system, as the captive portal is opened in a non-compatible web browser. Most of Android devices had the Android 12 or 13 version, which allow the user to open an alternative compatible browser for authenticating in the network. As all users use technology on a daily basis, they should be able to manipulate their own devices correctly, which will help them to perform tasks reliably.

6. Results

This work resulted in a novel network authentication protocol based on FIDO2 authentication: FIDO2CAP. The protocol includes a new architecture based on RFC 8952 [8], and a message flow design for authentication and registration. Our proposed architecture adds a WebAuthn Authentication Web Application (WAWA), a FIDO2 Authenticator, and other elements to a captive portal. Also, we present a prototype implementation of FIDO2CAP in an environment mocking a hotel Wi-Fi network.

Finally, we conducted compatibility tests and a usability experiment with 15 real users. We found that more than 69% of them successfully completed the connection, but they had different problems: almost two errors on average per task. They performed much better on the second try. Most of them found the system easy or very easy to use.

In this section, we summarise the specific results, namely, (1) the FIDO2CAP protocol; (2) a prototype of the FIDO2CAP in a mock scenario; and (3) the results of the compatibility tests and the usability experiment.

6.1. FIDO2CAP: A Novel Protocol for Captive-Portal FIDO2 Authentication

In Section 4, we define FIDO2CAP: FIDO2 Captive-portal Authentication Protocol. This is the main result of this paper. The proposed novel protocol is based on captive portals defined by RFC 8952 [8], and adapted to support FIDO2 user authentication. The main advantage of this approach is that any captive portal compatible with the RFC could be modified with the architecture of FIDO2CAP to support this new authentication protocol.

The FIDO2CAP architecture is shown in Figure 2, where we have included some elements, like the User Database and the WAWA. On the other hand, we have considered the two FIDO2 ceremonies: authentication and registration. The message flows are presented in Figure 3 for authentication, and Figure 4 for registration. In both flows, we take into account discoverable and non-discoverable credentials, which ensures compatibility with different FIDO2 authenticators.

6.2. Functional Prototype of FIDO2CAP

We have developed a prototype of FIDO2CAP in the scenario described in Section 5.3. This scenario describes a hotel Wi-Fi network protected by a FIDO2CAP captive-portal system, which controls access to the Internet and other network resources by the hotel clients.

The developed prototype has been published as open-source in Github [24], available to download and deploy in a compatible OpenWRT router. It provides documentation of the configuration options and some instructions for its deployment. In this section, we describe the specific results.

6.2.1. WebAuthn Authentication Web Application (WAWA)

WAWA is a web application that allows a user to authenticate using FIDO2 Authenticators compatible with the WebAuthn standard, using a web browser. The registration of users, and their corresponding security keys, is performed by a registrar, which is a privileged user, implemented using RBAC roles.

The developed server is compatible with the last W3C Recommendation of the WebAuthn standard L3 [29], implementing both discoverable and non-discoverable WebAuthn credentials. This feature makes all types of FIDO2 security keys or software “passkeys” compatible with the server registration and authentication procedures. In the developed system, modern “passkeys” or discoverable credentials are supported, so the user does not need to introduce a username in the authentication form. If the user has software or hardware not compatible with discoverable credentials, they can use the system with non-discoverable credentials by introducing their username (see Figure 8).

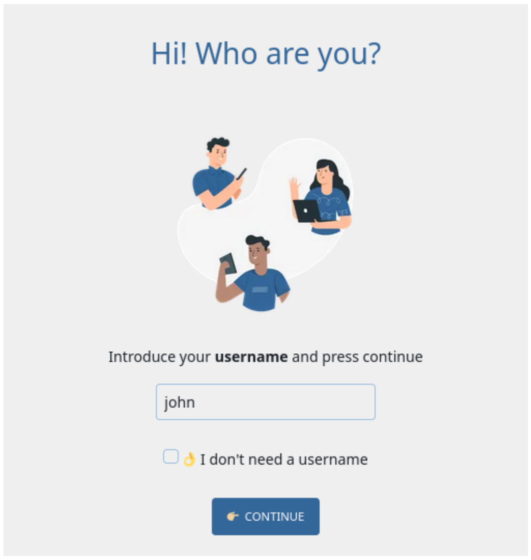
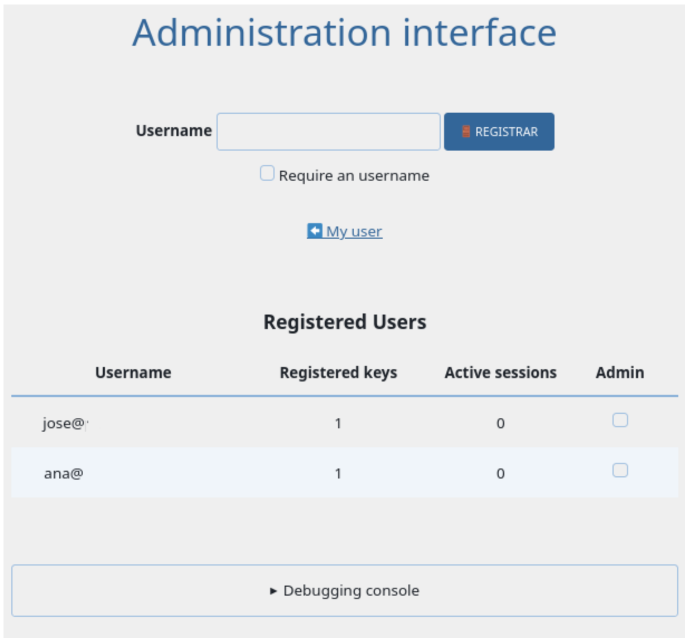


Figure 8. Developed captive portal UI. The WebAuthn login form uses discoverable credentials when the "I don't need a username" option is selected.

Additionally, an administrator user in the server can list authenticated users in real time (see Figure 9). The session database allows users to open more than one authenticated web session, which are listed in the administration panel. Although the user can logout at any time during the session, the expiration time will force the end of the web session automatically.

Finally, an administrator can register security keys. These can be associated with an existing user by specifying the username, or can be registered and associated to a new user. Therefore, a user can have multiple registered security keys. This feature allows users to have a backup security key, which can be used in case of device loss to securely gain access to the network.



Registered Users

Username	Registered keys	Active sessions	Admin
jose@	1	0	☐
ana@	1	0	☐

Figure 9. The administration interface of the developed authentication server. It allows the registration of security keys. The table of registered users lists the registered device of each user and the active sessions in real time. Admin column is used to configure admin privileges to users.

6.2.2. Integration of WAWA in the OpenNDS Captive Portal

The FIDO2CAP prototype includes the integration of the WAWA server with the OpenNDS captive-portal network authentication system. This integration adapts the WAWA server to communicate with the OpenNDS Enforcement Device and API Server as defined by the FIDO2CAP protocol. The integration allows the authorisation of the User Equipment traffic after the FIDO2 authentication with the WAWA server. The following are some specific results achieved during the development of the FIDO2CAP prototype.

Firstly, with this integration, our prototype benefits from the compatibility of OpenNDS. For instance, the integrated solution triggers the Captive Portal Detection (CPD) technology embedded in the most used operating systems, showing the captive-portal page once connected to the network.

Other relevant results are that the same WAWA server can be configured to work with different OpenWRT routers (or Access Points) to control access in different networks. This way, the resulting prototype provides a central point of authentication compatible with a multi-gateway scenario. The same user can be authenticated in different networks, creating different sessions that can be viewed from the administration panel, while all gateway requests are managed independently.

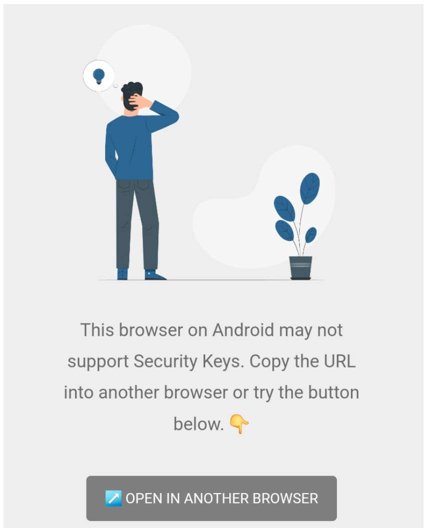
Finally, thanks to the integration with OpenNDS, the solution is compatible with a real deployment scenario that we used to validate the system (see Section 6.3). OpenNDS can be installed on OpenWRT router firmware, which can form part of a final production environment easy to deliver with multiple hardware routers. These devices can run the OpenNDS captive-portal software, configured to be used with the integrated developed WAWA server. This way, when a client connects to the network via OpenWRT, it gets redirected to the WebAuthn authentication server, who manages the request accordingly. After the client authentication, the authentication server waits for the periodic requests and confirms the client authorisation, redirecting them to the original HTTP request.

6.3. FIDO2CAP Compatibility and Usability

With the described FIDO2CAP prototype deployment, we tested different operating systems and browsers. Finally, we conducted a usability experiment with real users. These results provide evidence of the feasibility of the protocol and its usability. In this section, we include the compatibility results and then the usability experiment results.

6.3.1. Compatibility with Operating Systems and Web Browsers

Applying FIDO2 authentication to a captive portal is a novel approach. For this reason, there are some captive-portal-specific browsers that operating systems launch that do not support the WebAuthn API. We tested our prototype solution in the most common operating systems and browsers. Table 1 shows the compatibility validation results. As it can be seen, the system is fully compatible with operating systems using the user-defined web browser, which usually supports WebAuthn. There are other operating systems that use a custom web browser with limited functionality to open captive-portal web pages, known as mini-browsers [27], which mostly are not compatible with WebAuthn. When there is no compatibility in a captive-portal mini-browser, the developed web interface shows a redirection button to the user, which opens a compatible web browser in Android and iOS (see Figure 10).



This browser on Android may not support Security Keys. Copy the URL into another browser or try the button below. 👇

Figure 10. Prototype UI warning the user that the OS or the browser are not compatible. The button opens a new browser. The target browser depends on the specific OS running the request. This screenshot was taken on an Android 13 phone running the default mini-browser for captive portals, and the button should open the Google Chrome app with the same URL.

6.3.2. Usability Experiment with Users

As explained in Section 5.4.2, we designed and conducted a usability experiment with 15 subjects with their own smartphone devices. Users were asked to connect to the Wi-Fi network running the developed captive-portal authentication service, and authenticate with a registered security key.

From the 13 users with compatible smartphones, 69.23% completed the task of connecting to the Wi-Fi network and successfully authenticating in the captive portal with security keys. Table 2 shows the completeness of every step of the task. As we can see, 84.62% reached a compatible web browser to start authentication. For those who had a compatible web browser, 81.82% completed authentication.

Table 2. Completeness of every step of the proposed task. Sample of 13 users with compatible smartphones.

Step	Total Completeness	Step Completeness
Connect to Wi-Fi network	100.00%	100.00%
Follow OS instructions	92.31%	92.31%
Browser redirection	84.62%	91.67%
Authentication	69.23%	81.82%

On the other hand, we detected several errors of different categories that constituted different types of obstacles during the execution of the task. Namely, we found (1) use errors, produced by users; (2) OS errors, caused by the software of the user device; (3) connectivity issues, while connecting the security key via USB or NFC; and (4) compatibility errors, as incompatibility with captive portal on iPhone devices.

Table 3 shows the error rate of 1.93. Most of the detected errors were operating system-related errors. Specifically, the most common error was the Android credential manager menu that delayed to appear, which confused some users. Even further, in some cases, we detected that the Android credential manager menu blocks the completion of the operation by becoming frozen, causing the user to abandon the task. On the other hand, there were errors caused by the use. We observed that some users tried to connect the key directly to the unlocked smartphone, expecting the Wi-Fi network to be automatically connected and authenticated without any additional step. Finally, some of the errors were caused by connectivity of the USB using an adaptor.

Table 3. Error rate of the different types of errors.

Type of Error	Error Rate
Use error	0.67
OS error	0.73
Connectivity error	0.40
Compatibility error	0.13
Total error rate	1.93

Most of the observed errors did not cause the task to fail, although they delayed its completion. For measuring the efficiency, we calculated Time-Based Efficiency (TBE) and Overall Relative Efficiency (ORE), shown in Figures 11 and 12. In the first try, users completed 6.18×10^{-4} objectives per second, while in the second try, where the users had learnt from the errors, users completed 1.571×10^{-3} objectives per second, more than the double. Also, we measured the Overall Relative Efficiency as the ratio of time taken by the users who successfully completed the task in relation to the total time taken by all users. The ORE of the first try is 54.38%, and 35.95% in the second try. In the second try, the users who failed the first time spent more time than the ones that completed the task the first time.

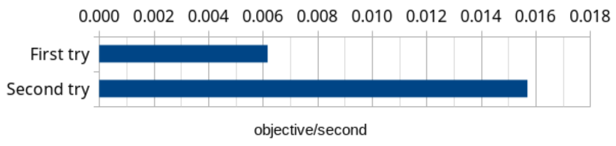


Figure 11. Time-Based Efficiency (TBE). Average objectives per second in each try. Users had learnt from errors by the second try.

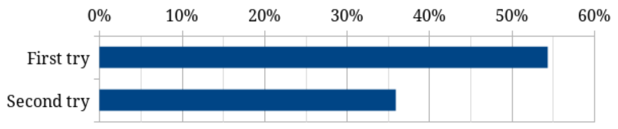


Figure 12. Overall Relative Efficiency (ORE). Ratio of time taken by the users who successfully completed the task in relation to the total time taken by all users.

Finally, we measured satisfaction through different questions during the experiment. According to Figure 13, 53.33% of users found it easy to complete the process of connecting to the Wi-Fi with security keys, and 57.14% found it easy to try again when some error occurred. At the end of the experiment, as shown in Figure 14, 66.67% of users declared that they would have no problem in using security keys for connecting to a Wi-Fi network, but none of them preferred using keys. That means that users do not directly find security keys more usable than other authentication methods, but they find the solution acceptable to use.

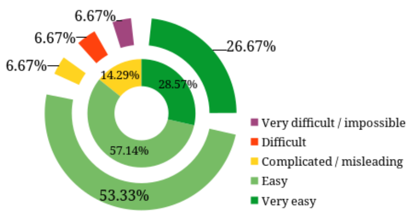


Figure 13. Difficulty as a measure of satisfaction: overall satisfaction (outer) and satisfaction from recovering from errors (inner). Answers from users after performing the task.

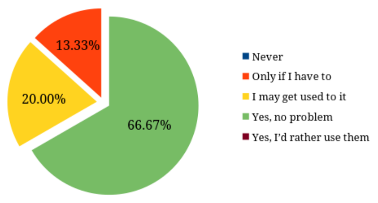


Figure 14. User answers to "Would you use security keys for connecting to a Wi-Fi network?" after finishing the experiment.

7. Discussion and Conclusions

We propose a novel authentication protocol based on FIDO2 authentication integrated with a captive portal. Moreover, we have developed a functional prototype on the OpenNDS captive portal. The prototype can be installed on a network to authenticate users with FIDO2 security keys. In addition, we have validated the prototype usability, finding use errors that can drive future research on FIDO2 usability with a more extensive experiment.

Our FIDO2CAP protocol and the implemented prototype are proof of how FIDO2 authentication can be integrated with other systems and serve as a replacement of passwords and their vulnerabilities. Therefore, security keys and passkeys can now be used with captive portals that authenticate users in real network authentication environments. However, there is some work left in relation to the usability and producing mature software with full compatibility across devices and web browsers. Our implemented prototype depends on custom web browsers that operating systems open when detecting a captive portal, impacting the user experience.

In conclusion, FIDO2 authentication represents an opportunity to improve security by migrating the existing authentication services to a new authentication paradigm. Considering that central authentication involves different systems, the migration to FIDO2 should be performed in all authentication scenarios of a business. They include not only web authentication but also other systems like network access control.

Our solution directly involves FIDO2 authentication within an access control system in the network, a novel approach to the best of our knowledge. This differs from previous works such as that of Chifor et al. [13]. In this work, they provision temporary password-based credentials after FIDO UAF authentication for a WPA2-Enterprise Wi-Fi network. Then, these provisioned credentials are sent to the device via a TLS channel and added to a RADIUS server. However, their solution requires the user to install a separate Android application to configure the temporary Wi-Fi credentials and the authentication with FIDO UAF. In FIDO2CAP, authentication is directly performed in the web browser, compatible with the new FIDO2 standards. The user is not required to install any new software. Further, the protocol adapts already existing technology, integrating the new authentication, avoiding the need for temporary password-based credentials. Once the captive portal is detected, the user only needs to perform FIDO2 authentication.

For validation, we tested both compatibility with the technology and its usability with users. We conducted a pilot usability study with 15 users using external FIDO2 security keys. The results showed that 69,23% of them completed the connection with these hardware tokens and had a high level of user satisfaction. During the execution of the experiment, some concerns were raised when asking about user acceptance. As some previous works discovered [30], users are concerned about losing a physical token. FIDO2 platform authenticators, like the embedded in smartphones, may be a more accepted approach in the following years. FIDO2CAP compatibility with both scenarios ensures the continuity of the solution.

However, it is important to highlight that WebAuthn and FIDO CTAP are recent standards that are still under development and under adoption. Although it has already been implemented in web authentication in different operating systems, browsers and devices, their applicability to further scenarios has not yet been studied. This paper proves that there are other applications, like network authentication. Although the underlining technology is web-based, the integration with a real captive-portal Enforcement Device demonstrates the potential of security keys in network authentication.

8. Future Work

In this paper, we proposed FIDO2CAP, and validated it with a real prototype and users. Here, we include some of our current research lines that continue the work we presented in this paper:

1. **Validate the usability of the solution with more users and compare different samples**. Design new environments and validate user acceptance with different users.
2. **Extend the prototype for auto-registration and validate its usability**. Test the usability of users registering their own credentials in the captive portal for authentication.
3. **Compare the usability of the developed captive portal and a captive portal based on passwords**. Evaluate how passwordless authentication with security keys is more usable than passwords.
4. **Study the usability issues caused by the error management of web browsers with WebAuthn**. We detected some differences when managing exceptions during a failure in WebAuthn authentication. We are working on a study of the current browser implementations and proposing a solution, which can help to improve the usability and adoption of this authentication method.

Author Contributions: Conceptualisation, M.R.-D.; methodology, M.R.-D. and J.V.-N.; software, M.R.-D.; validation, M.R.-D.; formal analysis, M.R.-D.; investigation, M.R.-D.; resources, M.R.-D., A.P. and J.V.-N.; data curation, M.R.-D.; writing—original draft preparation, M.R.-D.; writing—review and editing, M.R.-D. and J.V.-N.; visualisation, M.R.-D., M.G., A.P. and J.V.-N.; supervision, M.G., A.P. and J.V.-N.; project administration, M.R.-D. and J.V.-N.; funding acquisition, A.P. All authors have read and agreed to the published version of the manuscript.

Funding: This work was founded by EU and “Xunta de Galicia” (Spain), grant ED431C 2022/46–Competitive Reference Groups GRC. This work was also supported by CITIC, funded by “Xunta de Galicia” through the collaboration agreement between the “Consellería de Cultura, Educación, Formación Profesional e Universidades” and the Galician universities to strengthen the research centres of the “Sistema Universitario de Galicia” (CIGUS). Also, the work is founded by the “Formación de Profesorado Universitario” (FPU) grant from the Spanish Ministry of Universities to Martiño Rivera Dourado (Grant FPU21/04519).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The developed prototype can be found in Github, published as Open Source: https://github.com/martinord/fido2cap-server (accessed on 23 April 2024).

Conflicts of Interest: Authors Marcos Gestal and Alejandro Pazos are partners of the company IKERDATA S.L. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

1. Thomas, K.; Li, F.; Zand, A.; Barrett, J.; Ranieri, J.; Invernizzi, L.; Markov, Y.; Comanescu, O.; Eranti, V.; Moscicki, A.; et al. Data Breaches, Phishing, or Malware? Understanding the Risks of Stolen Credentials. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, Dallas, TX, USA, 30 October–3 November 2017. [CrossRef]
2. Fido Alliance—Open Authentication Standards More Secure Than Passwords. Available online: https://fidoalliance.org/ (accessed on 1 April 2024).
3. Web Authentication: An API for Accessing Public Key Credentials Level 1. Available online: https://www.w3.org/TR/webauthn-1/ (accessed on 1 April 2024).
4. Web Authentication API—Web APIs | MDN. Available online: https://developer.mozilla.org/en-US/docs/Web/API/Web_Authentication_API (accessed on 1 April 2024).

5. Rivera-Dourado, M.; Gestal, M.; Pazos, A.; Vázquez-Naya, J.M. An Analysis of the Current Implementations Based on the WebAuthn and FIDO Authentication Standards. *Eng. Proc.* **2021**, *7*, 56. [CrossRef]
6. Cisco Systems, 2022. The 2022 Duo Trusted Access Report. Available online: https://duo.com/assets/ebooks/the-2022-duo-trusted-access-report.pdf (accessed on 1 April 2024).
7. 'Passkey—The Simplest Way to Sign Into Your Google Account'. Available online: https://safety.google/authentication/passkey/ (accessed on 1 April 2024).
8. Larose, K.; Dolson, D.; Liu, H. RFC 8952: Captive Portal Architecture. RFC Editor. 2020. Available online: https://www.rfc-editor.org/info/rfc8952 (accessed on 1 April 2024).
9. Captive Portal | pfSense Documentation. Available online: https://docs.netgate.com/pfsense/en/latest/captiveportal/index.html (accessed on 1 April 2024).
10. Brown, R. Welcome to the OpenWrt Project . OpenWrt Wiki. 2016. Available online: https://openwrt.org/start (accessed on 1 April 2024).
11. Forwarding Authentication Service (FAS)—OpenNDS v9.10.0 . Available online: https://opennds.readthedocs.io/en/v9.10.0/fas.html (accessed on 1 April 2024).
12. Rivera-Dourado , M.; Gestal, M.; Pazos, A.; Vázquez-Naya, J. Adapting a Captive Portal for Phishing-Resistant Network Authentication Using Security Keys. In Proceedings of the 2023 JNIC Cybersecurity Conference (JNIC), Vigo, Spain, 21–23 June 2023; pp. 1–8. [CrossRef]
13. Chifor, B.-C.; Teican, S.; Togan, M.; Gugulea, G. A Flexible Authorization Mechanism for Enterprise Networks Using Smart-Phone Devices. In Proceedings of the 2018 International Conference on Communications (COMM), Bucharest, Romania, 14–16 June 2018; pp. 437–440. [CrossRef]
14. Chifor, B.-C.; Bica, I.; Patriciu, V.-V.; Pop, F. A security authorization scheme for smart home Internet of Things devices. *Future Gener. Comput. Syst.* **2018**, *86*, 740–749. [CrossRef]
15. Luo, H.; Wang, C.; Luo, H.; Zhang, F.; Lin, F.; Xu, G. G2F: A Secure User Authentication for Rapid Smart Home IoT Management. *IEEE Internet Things J.* **2021**, *8*, 10884–10895. [CrossRef]
16. Huseynov, E. Passwordless VPN using FIDO2 Security Keys: Modern authentication security for legacy VPN systems. In Proceedings of the 2022 4th International Conference on Data Intelligence and Security (ICDIS), Shenzhen, China, 24–26 August 2022; pp. 1–3. [CrossRef]
17. Marques, N.; Zúquete, A.; Barraca, J.P. EAP-SH: An EAP Authentication Protocol to Integrate Captive Portals in the 802.1X Security Architecture. *Wirel. Pers. Commun.* **2020**, *113*, 1891–1915. [CrossRef]
18. Anderberg, P.; Thorselius, E. How to Circumvent a Captive Portal 5. Available online: https://www.ida.liu.se/~TDDD17/oldprojects/2010/projects/003-2.pdf (accessed on 1 April 2024).
19. Kumari, W.; Kline, E. RFC 8910: Captive-Portal Identification in DHCP and Router Advertisements (RAs). Internet Engineering Task Force. 2020. Available online: https://www.rfc-editor.org/info/rfc8910 (accessed on 1 April 2024).
20. Security | Wi-Fi Alliance. Available online: https://www.wi-fi.org/discover-wi-fi/security#Wi-FiEnhancedOpen (accessed on 1 April 2024).
21. Karim, N.; Kanaker, H.; Abdulraheem, W.; Ghaith, M.; Alhroob, E.; Alali, A. Choosing the Right MFA Method for Online Systems: A Comparative Analysis. *Int. J. Data Netw. Sci.* **2024**, *8*, 201–212. Available online: http://m.growingscience.com/beta/ijds/6474-choosing-the-right-mfa-method-for-online-systems-a-comparative-analysis.html (accessed on 1 April 2024). [CrossRef]
22. Baghdasaryan, D.; Hill, B.; Hill, J.; Biggs, D. FIDO Security Reference. 2022. Available online: https://fidoalliance.org/specs/common-specs/fido-security-ref-v2.1-ps-20220523.html (accessed on 1 April 2024).
23. Miller, M. SimpleWebAuthn Project. Github. 2022. Available online: https://github.com/MasterKale/SimpleWebAuthn (accessed on 1 April 2024).
24. martinord/fido2cap-server. WebAuthn Authentication Web Application Compatible with OpenNDS Captive Portal. GitHub. Available online: https://github.com/martinord/fido2cap-server (accessed on 1 April 2024).
25. Market Share Held by the Leading Computer (Desktop/Tablet/Console) Operating Systems Worldwide from January 2012 to June 2023 [Graph], StatCounter, 21 July 2023. Available online: https://www.statista.com/statistics/268237/global-market-share-held-by-operating-systems-since-2009/ (accessed on 1 April 2024).
26. Global Market Share Held by Mobile Operating Systems from 1st Quarter 2009 to 2nd Quarter 2023 [Graph], StatCounter, 1 July 2023. Available online: https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009/ (accessed on 9 March 2023).
27. Wang, P.L.; Chou, K.H.; Hsiao, S.C.; Low, A.T.; Kim, T.H.J.; Hsiao, H.C. Capturing Antique Browsers in Modern Devices: A Security Analysis of Captive Portal Mini-Browsers. In *Applied Cryptography and Network Security*; Tibouchi, M., Wang, X., Eds.; ACNS 2023: Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2023; Volume 13905. [CrossRef]
28. *ISO 9241-11:2018*; Ergonomics of Human-System Interaction—Part 11: Usability: Definitions and Concepts. ISO: Geneva, Switzerland, 2018. Available online: https://www.iso.org/standard/63500.html (accessed on 1 April 2024).

29. Web Authentication: An API for Accessing Public Key Credentials Level 3. Available online: https://www.w3.org/TR/webauthn-3/ (accessed on 1 April 2024).
30. Ruoti, S.; Roberts, B.; Seamons, K. Authentication Melee: A Usability Analysis of Seven Web Authentication Systems. In Proceedings of the 24th International Conference on World Wide Web, Florence, Italy, 18–22 May 2015. [CrossRef]