Semantic Approach to Designing Applications with Passwordless Authentication According to the FIDO2 Specification

Anton Zhidovich and Alexei Lubenko and Iosif Vojteshenko and Alexey Andrushevich
Belarusian State University
Minsk, Belarus
{anton.zhidovich, alexeilubenko02}@gmail.com, {voit, andrushevich}@bsu.by

***Abstract*—In this paper, a semantic approach to designing applications with the FIDO2 specification-based passwordless authentication using OSTIS technology is proposed. Obtained results will improve the efficiency of the component approach to the development of applications with passwordless authentication, as well as provide the ability to automatically synchronize different versions of components, increasing their compatibility and consistency.**

***Keywords*—FIDO2 technology, passwordless authentication, OSTIS technology, biometrics**

I. INTRODUCTION

When building an intelligent system, it's necessary to accord special priority to the issue of access to system resources and the differentiation of user rights. The key concept here is authentication – a procedure of identity verification to ensure that the user is the subject whose identifier he uses. The issue becomes more complicated when designing different semantically compatible intelligent systems [1], requiring a unified authentication apparatus: easy to use and integrate, as well as the most secure.

The authentication system is only a component, and therefore the development of a unified approach to its design is required. There are various authentication standards, many of which may not provide a high level of security and, moreover, may be compatible only with a certain software class or be proprietary.

II. ANALYSIS OF AUTHENTICATION METHODS

Let's take a look at the most common authentication methods. These methods can be encountered both in everyday life, with the use of messengers, online banking or other online services, and within corporate systems where data is accessed by company employees and delineated according to their position.

A. Password-based authentication

Being the most common because of its ease of implementation, password-based authentication method is vulnerable to the most types of attacks: brute-force, range attacks, dictionary attacks, key-logging, social engineering such as phishing, man-in-the-middle and replay attacks.

B. Trusted third-party authentication

The method is based on the fact that the service (provider) that owns the user's data, with his permission, provides third-party applications with secure access to this data. The provider is usually a service such as Google, GitHub, Facebook or Twitter. The most common implementation is the OAuth 2.0 protocol.

The OAuth 2.0 specification defines a protocol for delegating user authentication to the service that hosts a user account and authorising third-party applications to access that user account [2].

In [2] the main participants of OAuth 2.0 authentication and their interaction are described. Although this method is one of the most user-friendly and, moreover, implemented in most online resources, it is still vulnerable to a man-in-the-middle attack, which is a common and effective way to gain unauthorised access to a system.

C. One-time password (OTP)

OTP, which is used in many systems as a second or first authentication factor, can be a number or some string that is generated for a single login process. When authenticating, the OTP can be sent to the user via SMS-message, push notification or in a special application. The most secure tool for generating one-time passwords is a token (software, such as Google Authenticator, or hardware).

OTP quickly becomes invalid, which provides resistance to replay attacks. However, most attacks on authentication systems with OTP target the way the user receives it. For example, OTP transmitted via SMS can be intercepted by software such as FlexiSPY or Reptilicus.

One-time passwords are protected against phishing in the classic sense: users cannot reveal long-term credentials. However, the man-in-the-middle attack can be used to retrieve a currently valid one-time password.

D. Passwordless authentication methods

Passwordless authentication allows a user to access an information system without entering a password or answering security questions. Instead, the user provides some other form of evidence such as a fingerprint, proximity badge, or hardware token code [3].

The essence of passwordless authentication is to never reveal any secrets. That is, everything about the user's identity and sensitive data remains protected.

There are several standards for passwordless authentication, depending on which factor is used and the available hardware, software and other features of the information system and its users. For example, biometric authentication involves comparing a user's unique biometric characteristics (facial features, retinal structure, etc.) with previously recorded samples of those characteristics. Biometric characteristics are inseparable from their owner, which ensures that it is impossible to refuse to log on and perform certain actions in the system. However, these authentication methods are also susceptible to hacking, such as face spoofing with a photo, 3D head model, etc. One way to combat such attacks on system security is liveness detection technology, which consists in checking the presented identifier for belonging to a "live" user and is designed to strengthen the identification procedure and protect against hacking in biometric authentication [4], [5].

The modern approach to passwordless authentication is the open standard FIDO2, jointly developed by the FIDO Alliance and the W3C consortium. The FIDO2 specification uses public-key cryptography and consists of two groups of standards. One of these is called the W3C WebAuthn standard. The second part is the Client to Authenticator Protocol (CTAP, CTAP1, CTAP2). The FIDO Alliance states that FIDO2 "reflects the industry's answer to the global password problem" by addressing legacy authentication's challenges as they pertain to security, usability, privacy, and scalability [6].

FIDO2-based authentication has advantages, primarily related to usability and security. FIDO2 can be used for passwordless login to the application or as an additional authentication factor, while ensuring a sufficient level of security for most tasks [7], [8]. The following advantages can be pointed out:

- The use of public-key cryptography provides resistance to phishing and man-in-the-middle attacks. Indeed, even by intercepting the public key or any data during registration, a fraudster cannot create a digital signature without access to the authenticator. The user authenticator itself is either built into the operating system, where the protection of the credentials (private key) is organised at the hardware level, or is an external device.
- Generating a special random byte buffer each time the server communicates with the authenticator prevents replay attacks.
- Users have a simple built-in mechanism like a fingerprint scanner to provide fast, secure, and convenient access to online services [9].
- Many operating systems and browsers have built-in support for WebAuthn API, which greatly simplifies the implementation of the technology in information systems.
- Cryptographic keys are unique for every website, providing users with enhanced privacy as sites cannot track the users across the web [9].

Thus, FIDO2-authentication is a modern, secure and convenient phishing-resistant method based on open standards and implemented in browsers and operating systems. The method provides ease of use by allowing users to register their device with a given online service through the selection of a local authentication mechanism. Speaking into the microphone, looking into the camera, inputting a PIN, or swiping a finger could all be valid local authentication mechanisms [10], [11].

III. SEMANTIC DEFINITION OF FIDO2 SPECIFICATIONS AND SECURITY KEYS

In order to increase the level of convergence and subsequent integration of a unified authentication system with next-generation intelligent computer systems, this paper proposes a semantic approach to their designing based on OSTIS technology. The OSTIS technology is a set of models, methods and tools permanently developed as part of an open project oriented on the ontological design, production, operation and re-engineering of semantically compatible hybrid intelligent computer systems that can independently interact with each other [12]. A number of languages are used to represent knowledge bases of ostis-systems (systems built on OSTIS Technology). Among them are external languages [12]:

- SCg (Semantic Code graphical) – a language whose texts represent a graph structures of a general type with precisely defined denotational semantics.
- SCn (Semantic Code natural) – a language for the structured external representation of SC-code texts.

A. Definition of FIDO2-authentication specifications using SC-code

W3C WebAuthn
:= [a specification developed by the FIDO alliance and W3C that allows an application to register and authenticate users using public-key cryptography instead of a password.]
:= [WebAPI built into platforms and browsers of all common operating systems to support passwordless authentication]

CTAP
⇒ *decoding**:
[Client to Authenticator Protocol]
:= [a specification that describes how the client (mobile app or web browser) and operating

system interact with cross-platform (physical) authenticators via USB, BLE, and NFC]

⇒ *levels**:

{• *Authenticator API*

:= [level that represents a specific set of authenticator functions used for generating new credentials, confirming authentication and cancelling current operations]

• *Message Encoding*

:= [level at which all requests to the Authenticator API level are generated and encrypted]

• *Transport-specific Binding*

:= [level at which requests and responses to the external authenticator are transmitted via USB, BLE, NFC]

}

⇒ *subdividing**:

{• *CTAP1*

:= [a protocol that describes client interaction with legacy authenticators as a second authentication factor]

• *CTAP2*

:= [a protocol that describes client interaction with new authenticators for passwordless access, two-factor authentication or multi-factor authentication]

}

The interaction of multiple parties is defined by the specifications included in FIDO2. Each party performs specific functions depending on its role in the authentication process:

FIDO2-authentication process participants

= {• *user*

• *client*

• *credentials*

:= [the pair of private and public keys associated with the user account]

• *security key*

• *relying party*

:= [a server that stores the public key associated with the user account, makes requests to the WebAuthn client, and verifies the authentication signature]

}

B. Definition of security keys in the WebAuthn specification using SC-code

In general, a security key (also called authenticator) is assumed to have only one user. If multiple natural persons share access to an authenticator, they are considered to represent the same user in the context of that authenticator. If an authenticator implementation supports multiple users in separated compartments, then each compartment is considered a separate authenticator with a single user with no access to other users' credentials [13].

Platform authenticators have a built-in Trusted Platform Module (TPM) used to secure any generated private keys and are often biometric in nature, although this is certainly not a requirement. When present however, the biometric element provides a mechanism for the platform to match against the device's identity profile of a user, and in turn, use the stored cryptographic credentials to authenticate against a relying party. When it is not present, the same outcome can be achieved with other methods such as PINs for instance, although potentially, may be less secure [14].

The following definition of security keys can be made in SC-code.

Security key

⇒ *synonyms**:

[authenticator, WebAuthn key]

:= [a software component built into the operating system or external device that supports FIDO2 authentication]

⇒ *subdividing**:

{• *cross-platform (roaming) security key*

:= [an external physical device, not tied to a specific platform (operating system), used for authentication on multiple devices]

⇒ *example**:

{• *YubiKey, developed by Yubico*

• *Titan Security Key, developed by Google*

• *OneKey, developed by CryptoTrust*

}

• *platform (internal) security key*

:= [a software module, implemented either as a separate application or at the operating system level, used for authentication on a single device]

⇒ *example**:

{• *Windows Hello for Windows 10/11*

• *Touch ID, Face ID for IOS/MacOS*

```
                    •    Graphic key, iris scanner,
                         PIN code, etc. for Android
                    }
        }
```

C. Definition of FIDO certification process using SC-code

At the moment, all common operating systems support passwordless FIDO2-authentication methods using internal and roaming authenticators, certified by FIDO. The authentication tool undergoes a multi-step FIDO certification, confirming compliance of the implementation with FIDO2 specifications and defining a security level (L1, L2, L3) depending on resistance to various types of cyberattacks and built-in software and hardware protection of credentials [15]. The FIDO certification process can be represented as follows in SC-code:

```
FIDO certification process
⇒     steps*:
        {•    Confirmation of compliance with FIDO
              specifications
         •    Functional compatibility testing
         •    Obtaining an authenticator security
              certificate of at least L1 level
         •    Certification submission
         •    Obtaining a trademark FIDO® Certified
         •    FIDO Metadata Service Registration
        }
```

The FIDO Authenticator Certification levels can be represented in the following SC-code:

```
Authenticator Certification Levels
∋     {•    L1
              ⇒    defence*:
                   [phishing and majority of scalable
                    attacks]
              ⇒    example*:
                   [YubiKey 5 developed by Yubico]
       •    L2
              ⇒    defence*:
                   [remote software attacks]
              ⇒    example*:
                   [vFido developed by SecureMetric
                    Technology]
       •    L3
              ⇒    defence*:
                   [remote software and local hard-
                    ware attacks]
              ⇒    example*:
                   [de.fac2 developed by German Fed-
                    eral Office for Information Secu-
                    rity]
      }
```

As shown in [16], authenticators with L1 certification level can be vulnerable to timing attacks on FIDO2-authentication, which can allow attackers to link user accounts on multiple resources. Such authenticators may not guarantee a high level of security and privacy. This vulnerability has now been patched for major browsers (Mozilla Firefox [17], Google Chrome [18]).

IV. REGISTERING A WEBAUTHN KEY AND AUTHENTICATING WITH IT

The WebAuthn API allows applications to create and use secure credentials based on public-key cryptography with limited scope for user authentication. These credentials are created and saved in protected memory by the authenticator.

There are two operations defined in the WebAuthn API: creation of new user credentials and authentication using existing ones.

A. Registering new user credentials

Registration of new credentials can be done either for an authorised user (using a password, other authenticator or any available factor) or when creating a new account. Figure 1 shows the sequence of actions of the participants in the process of creating and registering new credentials. Let's look at the implementation using a WEB application as an example.

The user initiates the creation of new credentials while being on the WEB application page. The relying party's server sends a request to the client. The request contains a challenge - a buffer of cryptographically random bytes generated on the server, as well as information about the user (the identifier) and information about itself. The request data goes to the JavaScript program executed by the client (browser). The protocol for communicating with the server, as well as creating the request in the required format, are beyond the scope of the WebAuthn specification and depend on the platform on which the server runs and the libraries used.

The JS program makes a request to the authenticator to create new credentials. The request must contain data received from the server, information about the authenticator being used (internal or roaming), information about the expected credentials (such as the type of cryptographic algorithm used by the authenticator to create the signature), and the name of the new credentials (displayed for the user). The request is passed as parameters to the authenticator via the WebAuthn API built-in to the browser.

After the authenticator receives the data, the user authentication phase begins. If the verification is successful, the authenticator creates a new asymmetric key pair, saves the private key in protected storage on the device and generates a message to the client. The message contains the unique identifier of the created credentials, the address of the original web application, the challenge buffer, and

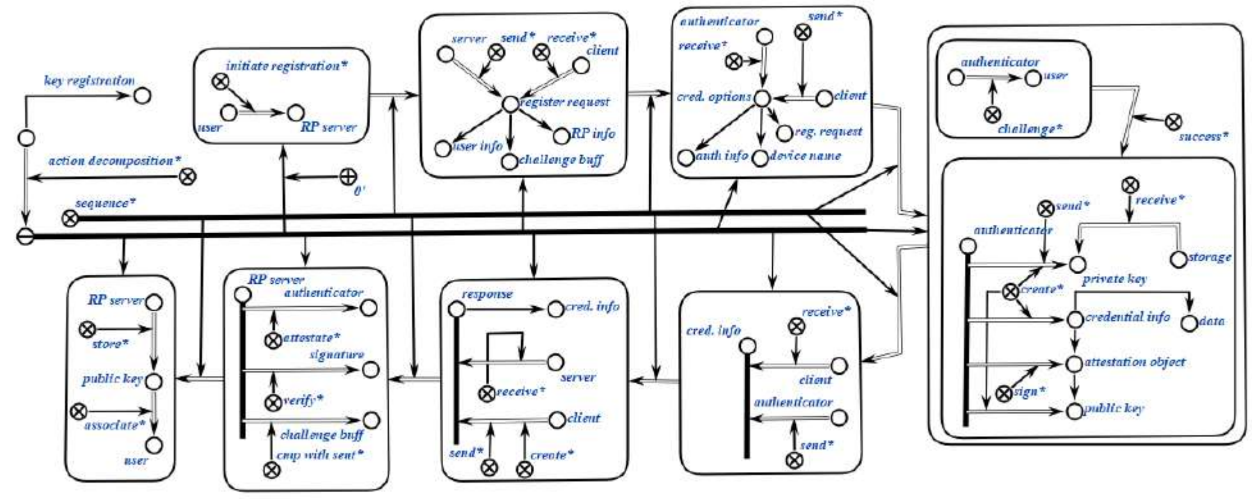
Figure 1. Key registration flow.

a special attestation object containing the public key and other information to be verified by the authenticator. The attestation object is signed with the private key.

The authenticator returns the data to the browser and the data is sent to the JavaScript program. The program creates a response to the server from the received data and then sends it to the server.

After receiving the data, the server must perform a series of checks to ensure that the registration process was passed and the data have not been modified: compare the received and sent values of the challenge buffer (they must be the same), verify the signature and perform attestation using the certificates set by the specific authenticator. If all the steps are successful, the server will store the received public key and associate it with the user account. The public key will be used later in user authentication .

B. Authentication using registered credentials

After a user has registered the WebAuthn key, it can be used in passwordless authentication. The authentication flow is much the same as the registration, but has its own peculiarities. The main difference is that the authenticator creates a specific authorization response signed with the private key instead of creating a new key pair. In a simplified form, the authentication process is shown in Figure 2.

V. ARCHITECTURE OF APPLICATION WITH FIDO2-AUTHENTICATION

Based on the analysis above, and using [19], we propose to formalise the architecture of an application with passwordless authentication based on the FIDO2 specification in the form of the following tree:

(0) Application architecture $S = A * B * C * D$

(1) Client platform $A = E * F * G$

(1.1) Client $E = H * I$

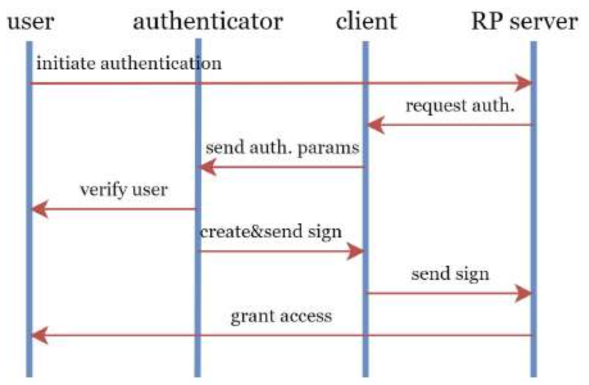

Figure 2. Authentication flow.

(1.1.1) Client-side JS-program H: H_1 (Processing of the RP request), H_2 (Accessing the WebAuthn API)

(1.1.2) WebAuthn API in the client I: I_1 (Authenticator registration), I_2 (Signature creation)

(1.2) Authentication API built into the platform F: F_1 (Locating the authenticator), F_2 (Connecting to the authenticator)

(1.3) Internal authenticator platform G: G_1 (Windows), G_2 (Android), G_3 (IOS), G_4 (MacOS)

(2) Roaming authenticator standard $B = P * T$

(2.1) Protocols P: P_1 (CTAP1), P_2 (CTAP2)

(2.2) Transports T: T_1 (Bluetooth Low Energy (BLE)), T_2 (NFC), T_3 (USB)

(3) Relying party $C = J * K * L$

(3.1) Server application J: J_1 (Relying party operations), J_2 (Basic logic)

(3.2) Library API interfaces K: K_1 (RSK FIDO2 Lib for ASP.NET), K_2 (Yubico libfido2)

(3.3) FIDO2 server $L = M * N$

(3.3.1) User & key storage M: M_1 (internal), M_2 (external)

(3.3.2) Attestation trust store N: N_1 (Full basic), N_2 (Surrogate basic), N_3 (With a privacy certification centre)

(4) External metadata service D: D_1 (Metadata Service v2.0 (MDS2)), D_2 (MDS3)

VI. Conclusion

In this paper a semantic approach to designing an application with FIDO2 authentication is described. The most common authentication methods have been analyzed and their common vulnerabilities identified. Using the capabilities of the OSTIS technology – SCn- and SCg-languages – the main components of FIDO2, as well as the WebAuthn key registration and authentication flows were described. The proposed approach enhances the efficiency of integration of FIDO2-authentication components in multi-agent intelligent systems.

Most modern operating systems and browsers have built-in support for FIDO2-authentication, in particular, have support for WebAuthn API, which is user-friendly, accessible, and protected from many types of attacks. In combination with the proposed approach, the technology can be easily integrated into OSTIS-systems, providing a single authentication apparatus.

In the context of authentication, an ontology that defines concepts — user, client, authenticator, certificates and levels of security, credentials, relying party, and its actions — and the relationships between them are presented. This will help to standardise the authentication process and ensure that the different devices and systems interact correctly. Standardisation of the authentication process improves functional compatibility and interoperability of systems and helps in the development and integration of new components of the OSTIS ecosystem.

Acknowledgment

The authors thank the teams of the Department of Software Engineering of the Belarusian State University and the Department of Intellectual Information Technologies of the Belarusian State University of Informatics and Radioelectronics for their help and valuable comments.

References

[1] V. Chertkov, "Information security in intelligent semantic systems," *Otkrytye semanticheskie tehnologii proektirovanija intellektual'nyh sistem [Open semantic technologies for intelligent systems]*, pp. 417–420, 2022.

[2] (2023, Feb) An Introduction to OAuth 2. [Online]. Available: https://www.digitalocean.com/community/tutorials/an-introduction-to-oauth-2

[3] (2022, Dec) Passwordless Authentication. [Online]. Available: https://www.cyberark.com/what-is/passwordless-authentication/

[4] S. Policepatil and S. M. Hatture, "Face liveness detection: An overview," *International Journal of Scientific Research in Science and Technology*, vol. 8, no. 4, pp. 22–29, Jul. 2021. [Online]. Available: https://doi.org/10.32628/IJSRST21843

[5] V. Zolotarev, A. Povazhnyuk, and E. Maro, "Metody usileniya procedury identifikacii pol'zovatelej na osnove tekhnologii liveness detection [Methods of strengthening the user identification procedure based on liveness detection technology]," *Izvestiya SFedU. Engineering Sciences*, vol. 14, no. 2, pp. 212–225, May 2022. [Online]. Available: https://doi.org/10.18522/2311-3103-2022-2-212-225

[6] (2023, Jan) FIDO2 Web Authentication. [Online]. Available: https://www.hypr.com/security-encyclopedia/fido2-web-authentication

[7] A. M. Kadan and E. R. Kirichonok, "Authentication module based on the protocol of zero-knowledge proof," *Selected Papers of the V International Scientific and Practical Conference "Distance Learning Technologies"*, pp. 365–373, 2020. [Online]. Available: https://CEUR-WS.org/Vol-2914/paper34.pdf

[8] A. Pathak, T. Patil, S. Pawar, P. Raut, and S. Khairnar, "Secure authentication using zero knowledge proof," in *2021 Asian Conference on Innovation in Technology (ASIANCON)*, 2021, pp. 1–8.

[9] (2022, Nov) What is FIDO2 and Its Benefits. [Online]. Available: https://www.kensington.com/news/security-blog/what-is-fido2-and-its-benefits//

[10] (2023, Jan) FIDO Authentication with WebAuthn. [Online]. Available: https://auth0.com/docs/secure/multi-factor-authentication/fido-authentication-with-webauthn

[11] (2023, Dec) Pros and Cons of FIDO authentication. [Online]. Available: https://www.securemetric.com/2019/05/17/pros-and-cons-of-fido-authentication/

[12] V. Golenkov, N. Guliakina, and D. Shunkevich, *Open technology of ontological design, production and operation of semantically compatible hybrid intelligent computer systems.* BSUIR, Minsk, 2021.

[13] (2022, Jan) Webauthn Specification. [Online]. Available: https://w3c.github.io/webauthn/

[14] (2023, Feb) Platform and Roaming Authenticators. [Online]. Available: https://developers.yubico.com/Developer_Program/WebAuthn_Starter_Kit/Platform_and_Roaming_Authenticators.html

[15] (2022, Dec) Functional Certification. [Online]. Available: https://fidoalliance.org/certification/functional-certification/

[16] M. Kepkowski, "How not to handle keys: Timing attacks on fido authenticator privacy," *22nd Privacy Enhancing Technologies Symposium*, pp. 705–726, 2022.

[17] (2023, Jan) FIDO2/WebAuthn privacy leak through a timing attack using silent authentications. [Online]. Available: https://bugzilla.mozilla.org/show_bug.cgi?id=1730434

[18] (2023, Jan) Stable Channel Update for Desktop. [Online]. Available: https://chromereleases.googleblog.com/2021/11/stable-channel-update-for-desktop.html

[19] (2023, Feb) WebAuthn Introduction. [Online]. Available: https://developers.yubico.com/WebAuthn/

Семантический подход к проектированию приложений с беспарольной аутентификацией по спецификации FIDO2

Жидович А. А., Лубенько А. А.,
Войтешенко И. С., Андрушевич А. А.

В работе предложен семантический подход к проектированию приложений с беспарольной аутентификацией по спецификации FIDO2 на основе использования компонентов и средств технологии OSTIS. Приведено формальное описание спецификаций и ключей безопасности FIDO2.

Полученные результаты позволят повысить эффективность компонентного подхода к разработке приложений с беспарольной аутентификацией, а также обеспечить возможность автоматической синхронизации различных версий компонентов, повышая совместимость и согласованность.

Received 14.03.2023