Hindawi
Security and Communication Networks
Volume 2021, Article ID 6245306, 13 pages
https://doi.org/10.1155/2021/6245306

WILEY | Hindawi

Research Article

Security Analysis and Bypass User Authentication Bound to Device of Windows Hello in the Wild

Ejin Kim [1] **and Hyoung-Kee Choi** [2]

[1]*Department of Electrical and Computer Engineering, Sungkyunkwan University, Suwon 16419, Republic of Korea*
[2]*College of Software, Sungkyunkwan University, Suwon 16419, Republic of Korea*

Correspondence should be addressed to Hyoung-Kee Choi; meosery@skku.edu

Received 19 April 2021; Revised 16 June 2021; Accepted 30 June 2021; Published 23 July 2021

Academic Editor: Ahmad Samer Wazan

Windows Hello is a Fast IDentity Online- (FIDO-) based new login system for Windows 10, which provides a single sign-on (SSO) service to diverse online applications. Hardware protection is essential for Window Hello's security. This paper aims to examine the security of Windows Hello on a device where hardware protection is unavailable. We present the first detailed analysis of Windows Hello's security. The results show that, on a hardware-unsupported device, the authentication data for Windows Hello is not properly protected. We propose a migration attack to compromise Windows Hello's security. In the proposed attack, an attacker extracts authentication data from a device to impersonate a victim in his or her Microsoft online account. We consider the possibility of such an attack to be serious and harmful to our society and demand immediate attention for remediation.

1. Introduction

A new standard, Fast IDentity Online version two (FIDO2), provides an alternative to password-based authentication by accommodating high-level yet easy-to-use security for user validation [1]. While a password authenticates a user based on what the user knows, FIDO2 is based on who the user is and what the user has. FIDO2 provides strong, attested, asymmetric public key-based credentials for the authentication of users. FIDO2 authenticates a user by verifying a private key with a public key registered in the FIDO server at the time of initial login. The private key is encrypted with FIDO credentials and never leaves the user's device. FIDO credentials with an asymmetric key pair are higher entropy than text-based passwords and are only valid within the user's device.

Microsoft has adapted FIDO2 and implemented it as Windows Hello in Windows 10 [2]. Windows Hello not only provides single sign-on services for device login but also extends the service to web browsers and desktop applications. Microsoft recommends using Windows Hello on devices that are equipped with the onboard Trusted Platform Module (TPM) [3]. Windows Hello's security hinges quite heavily on device dependency, which means that even if login credentials are leaked, those login credentials are of no use on other devices and, hence, the accounts are safe. This is because a private key plays a main role in authentication and the login credentials are used to encrypt the private key. Simply transferring the login credentials between devices does not compromise a victim's account. An adversary is required to acquire both the login credentials and the private key.

The private key never leaves a device once it is set for an individual user, which is done at the time of account creation. Furthermore, the private key can be saved in the cryptographic hardware, the so-called TPM, and security data are unavailable outside the TPM. Windows Hello is most secure when authentication data is stored in a TPM. However, TPMs have only been actively deployed very recently [4]. Currently, many Windows devices, especially desktops and servers, run without missioning the TPM.

In this paper, we evaluate the security of Windows Hello on a hardware-unsupported device by examining how difficult it is to break the device dependency. We analyze, in

detail, how Windows Hello works to identify authentication data on a device and how that data is processed by the operating system for user authentication. Based on the detailed analysis, we propose a migration attack. The attack allows authentication data to be transferred between devices without being limited by device dependency. Anyone, including adversaries, can impersonate a victim in Microsoft's online services. In addition, we built a migration tool, which is available as an open-source solution. This tool extracts essential data used for authentication from a victim's device and, if necessary, disarms any protection applied to the data. The tool then reinstates the same protections on the data to fit it to an adversary's device. Despite Microsoft's claims to the contrary, the device dependency is not as secure as we would hope on hardware-unprotected devices.

The remainder of this paper is structured as follows. Section 2 contains background information relevant to FIDO2 security, including how Windows Hello adapts FIDO2. Section 3 introduces an overview of the proposed attack, the Windows Hello migration attack. Sections 4–6 give a step-by-step explanation of the proposed attack based on how Windows Hello works. Section 7 evaluates the impact and feasibility of our attack in the wild. Section 8 presents related works regarding the security of FIDO2 and Windows Hello. Section 9 concludes this paper.

2. Passwordless Authentication

The password is the most popular user authentication method; password protection is employed by many web services and operating systems, including Microsoft Windows. Password-based authentication does not achieve the extremely high level of security that we would like, because passwords can be stolen or hijacked [5]. Thus, passwordless authentication was proposed in an effort to compensate for the weaknesses of passwords. Fast IDentity Online version two (FIDO2) is an alternative to password-based authentication that provides high-level yet easy-to-use security for user validation [1].

2.1. Passwordless FIDO2 Standard Authentication. FIDO2 uses a suite of user credentials, such as biometrics, personal identification numbers (PINs), and external devices like mobile devices and Universal Serial Bus (USB) devices, to support passwordless login. It relieves users of the need to remember long passwords and to expose their credentials in public. FIDO2 was quickly adopted by many web services and operating systems. Windows, Android, and major browsers such as Edge and Chrome now officially support FIDO2 authentication.

FIDO2's security depends on separating the user's credentials in user devices from the FIDO server. The first time a user accesses a FIDO-enabled web service, the user generates an asymmetric key pair for authentication purposes. Once it is encrypted by the user's credentials, the private authentication key is saved in the user's device. The public authentication key is registered on the FIDO server. This key pair is used by the FIDO server to authenticate the user during each subsequent login attempt. The credentials and the private key never leave the user's device.

FIDO2 serves two main functions: (1) user authentication and (2) device attestation. Figure 1 illustrates the implementation of these main functions in two protocols, the web authentication (WebAuthn) protocol [6] and the Client to Authenticator Protocol (CTAP) [7]. The web authentication protocol defines an authentication procedure between a user and a web client, such as a browser, and an attestation procedure between a user device and a FIDO server. A FIDO authenticator is a cryptographic entity that authenticates users and protects the asymmetric key pair for attestation. Internal and external authenticators in a user device play a key role in authentication and attestation on behalf of users.

The internal authenticator is responsible for user authentication using PINs or biometrics readings, such as a fingerprint, iris scan, and face recognition. A user may use an external authenticator using Bluetooth low energy (BLE), near-field communication (NFC), and the security key. The CTAP is a protocol that carries messages between a user device and an external authenticator.

A user may have different authenticators in different devices and for different services, which means that the user may use the same credentials in multiple devices and services without sacrificing security. Simply acquiring the credentials does not help an adversary hijack a user's account. This is true because users are uniquely authenticated by a public key in a server and a matching private key that is stored in a device encrypted by the credentials. In order to hijack the user's account in the web service, the attacker must steal the user's device and credentials.

2.2. Microsoft's Adaptation of FIDO2, Windows Hello. We focus on a Microsoft adaptation of FIDO2, called Windows Hello; this is a new user authentication technology first implemented in Microsoft Windows 10. Windows Hello helps users access Microsoft applications and third-party applications without the need for passwords, which can be problematic. Windows Hello authenticates a user using a PIN or biometric readings, called the gesture. The gesture plays the same role as the credentials in FIDO2. Windows Hello uses asymmetric key pairs to authenticate the user and the device to the Microsoft server. The gesture never leaves the device, nor is it exposed to the Microsoft server. Microsoft claims that the user's gesture, even if it is a short PIN, is more secure than a traditional password [8].

Windows Hello officially supports three types of user login: (1) device login with a local account, (2) device login with a Microsoft account, and (3) application login with a Microsoft account. A user may choose either a local account or a Microsoft account. A user with a Microsoft account can access multiple devices with a single account, while with the other types of account a user must have an independent account for each device. Furthermore, the Microsoft account enables users to access a number of services offered by Microsoft or third parties running on a Microsoft service framework. This is a single sign-on (SSO) service. Another

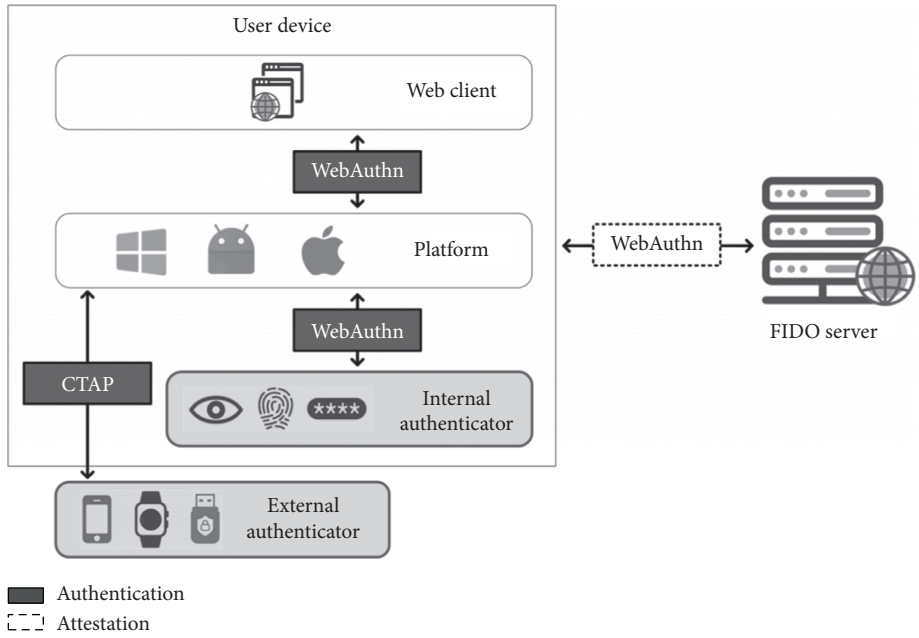

Figure 1: FIDO2 works with WebAuthn and CTAP. Through WebAuthn, an internal authenticator in the platform communicates with the FIDO server to authenticate a user. Optionally, the user activates an external authenticator with CTAP using other security devices.

difference between these two types of accounts is that they place information associated with authentication on either a remote server or a local device. In any case, Windows Hello is the default setting for user authentication in Windows 10. With either a local or a Microsoft account, authentication is initiated by a user providing his or her gestures.

When a user registers a Microsoft account on a local device, the use of FIDO's CTAP implementation for mobile applications can enhance the security of the account. If the user activates the two-factor authentication option for the Microsoft account, they log in to the account by entering a security code delivered to the Microsoft authenticator application on the Windows device [9]. After the initial login, the service is used through Windows Hello without additional CTAP authentication.

3. Windows Hello Migration Attack

3.1. Motivation and Goal. We show that an attacker who steals the victim's PIN can access Microsoft services from any device. When Windows Hello is used in combination with single sign-on services, the problem becomes more serious. When the authentication data for Windows Hello is not properly protected, all associated single sign-on services with Windows Hello will be exposed to danger if an attacker steals only the authentication data for Windows Hello.

We first explain how Windows Hello works on general devices. Based on the analysis, we propose a new attack, the Windows Hello migration attack. The attacker's goal is to access the victim's account without authorization. The attacker first obtains the gesture and the authentication data for Windows Hello on the victim's device. Having done so, the attacker can then access the Microsoft account, as well as any resources that are stored on the victim's device. Furthermore, the attacker can access cloud-based Microsoft applications, such as Office 365 and Microsoft Store and third-party applications. The proposed attack works even if the user has set up two-factor authentication for the Microsoft account. As a result, the proposed attack bypasses all FIDO authentication methods adopted by Microsoft.

3.2. Attack Model. A victim owns a Microsoft account running on a Windows device. The victim has enabled two-factor authentication using the Microsoft authenticator application. Our attack initiates on the device once the victim completes the registration at Microsoft for the use of Windows Hello.

Windows Hello's authentication data are stored in the hard disk and protected by Windows. The Access Control List (ACL) in Windows restricts users' and services' access to authentication data [10]. Only users and services with a valid security identifier (SID) are granted access to authentication data [11]. The SID is given to those who have an administrator's privilege. However, these multiple layers of security can be compromised by the migration attack.

The migration attack presumes the administrator's privilege available to the attack. A number of techniques and programs are available to us and attackers not to believe that our assumption is too optimistic [12, 13]. In one way, the attacker may entice victims to download malware through social engineering like e-mail phishing. In another way, the attacker may remotely exploit arbitrary code execution (ACE) vulnerabilities that existed in software running on the victim's device. The vulnerabilities can be found at the level of applications such as web browsers, services, kernels, and drivers in Windows.

The next step is to bypass a user consent requested by the user account control (UAC) [14]. The attacker attempts to escalate the privilege to gain the administrator's privileges without being noticed by the victim [15–17]. One of the most common ways is to find the credentials of an administrator account required when a normal user wants to use the administrator privilege. The attacker attempts to obtain the existing credentials of the administrator account through storage exploration, memory scraping, and guessing attacks, or add a new account with administrator privileges to the device. In other ways, the attacker seeks ways to set the attacker's process with the high privilege by manipulating permission tokens and policies managed by the operating system. This allows the attacker to modify ownership of the attacker's process and bypass access control. Another commonly used method for privilege escalation is exploiting vulnerabilities in software. The attacker abuses applications or services running with the administrator privilege. The attacker inserts malicious code or library into memory using programming errors at the kernel level.

Privilege escalation vulnerabilities are constantly found, and recently discovered vulnerabilities can be easily found in the Common Vulnerabilities and Exposures (CVE) list [18]. Our study assumes that the attacker already has infiltrated the victim's device and gained the administrator privilege in any way.

The attacker launches the migration attack in four steps, which will be discussed in the remainder of the paper. Figure 2 illustrates the flow of the migration attack. The attacker identifies the data that is used to authenticate the user and the device to the Microsoft server (① in Figure 2). The attacker extracts the authentication data for Windows Hello, which is stored in the victim's device (② in Figure 2). The attacker receives data through the network and calibrates the attacker's device with the victim's authentication data (③ in Figure 2). After the data is migrated, the attacker can access applications with the victim's account on the attacker's device (④ in Figure 2).

4. Authentication Protocol in Windows Hello

The attacker's ultimate goal is to log in to the application with the victim's account. In order to do so, the attacker examines the data that is exchanged between the Microsoft server and a device to validate the user, the account, and the device when the user attempts to log in. To examine the process that occurs when the device communicates with the Microsoft server for the first time, we begin our analysis at the point at which the user creates a local account on a Windows device.

Windows Hello users must perform the three following steps: (1) provision a Windows device and a Microsoft account, (2) set up the gesture, and (3) log in to an application with Windows Hello. After elaborating on all three steps, we will discuss our analysis of Windows Hello authentication security.

4.1. Step 1: Provision of a Windows Device and a Microsoft Account. When a Windows device and a Microsoft account are first provisioned, the device receives two tokens from the Microsoft server, as shown in Figure 3. The Microsoft server and the user device communicate using the Extensible Markup Language (XML) protocol [19] which encapsulates all data that is transmitted to the network. The server and the device use XML encryption [20] and XML signature [21] technology to ensure the integrity and confidentiality of sensitive information in XML. An XML key K_{XML} which encrypts and signs sensitive data is issued when the local account is created on the Windows device and registered to the server (① and ② in Figure 3). A device token, denoted by T_1, is an identifier of the XML key. The device transmits the device token with each communication to specify the XML key used for XML encryption and the XML signature.

The other token is an account identifier, denoted by T_2 (④ in Figure 3). As long as T_2 does not expire, the user can access Microsoft services, including registration of Windows Hello, without entering the account password. In this phase, the Microsoft server associates the device token T_1 with the account token T_2 so that the account token can be used together with the associated device token. T_1 plays the role of an identifier for the device and an XML key. Another parameter denoted as *AuthTicket* in ④ in Figure 3 enables the user to sign in to the device with the Microsoft account.

If the user has enabled the two-factor authentication option in the Microsoft account, they need to perform additional authentication with the Microsoft authenticator application when signing in with the Microsoft account (between ③ and ④ in Figure 3). T_2 is a token containing a state in which two-factor authentication has been completed. Subsequent requests using T_2 are not checked for two-factor authentication.

4.2. Step 2: Set Up a Gesture. Figure 4 illustrates the six messages that are exchanged to set up a gesture. For gesture setup, a user verifies themselves to the Microsoft server using T_2 in the second message. If T_1 and T_2 are not associated, the server requests the account password. In the third message, the server returns a renewal of T_2, denoted by T_2', along with a new token, T_3. The new token will be used later, in the fifth message, to confirm the user's account on the device. When the user generates the gesture, an asymmetric key pair is also generated. This key pair is unique to each device, which means that the same user's account in different devices will have different keys.

After the private key is registered, the association between the device token (T_1) and the account token (T_2) is broken up. Instead of the device token, the Microsoft server verifies the association between the account token and the asymmetric key pair. The device token is not involved in authentication; it only works as a key identifier for XML encryption and XML signature.

After being encrypted with the gesture, the private key K_{Pri} is encrypted and saved in a file accessible only by a user with an administrator privilege. The public key K_{Pub} is sent to the Microsoft server along with T_3 in the fifth message in Figure 4. The gesture never leaves the device where it was created.

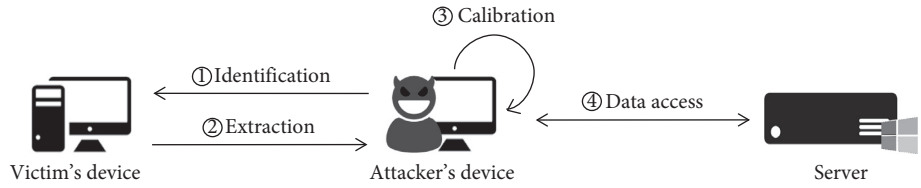

Figure 2: Flow of Windows Hello migration attack. An attacker identifies a victim's authentication data for Windows Hello and extracts it from the victim's device. The attacker then calibrates it to their device. Finally, the attacker accesses applications with the victim's account.

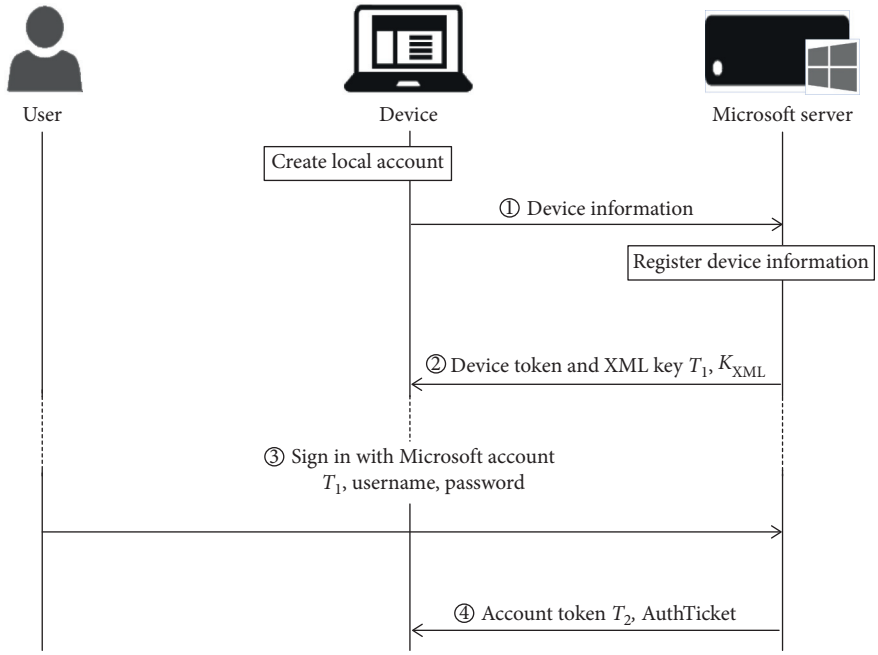

Figure 3: Detailed protocol flow: provisioning a Windows device and a Microsoft account. When a user creates a local account in a Windows device and logs in with the Microsoft account, the Microsoft server issues encrypted tokens to identify the device and the account.

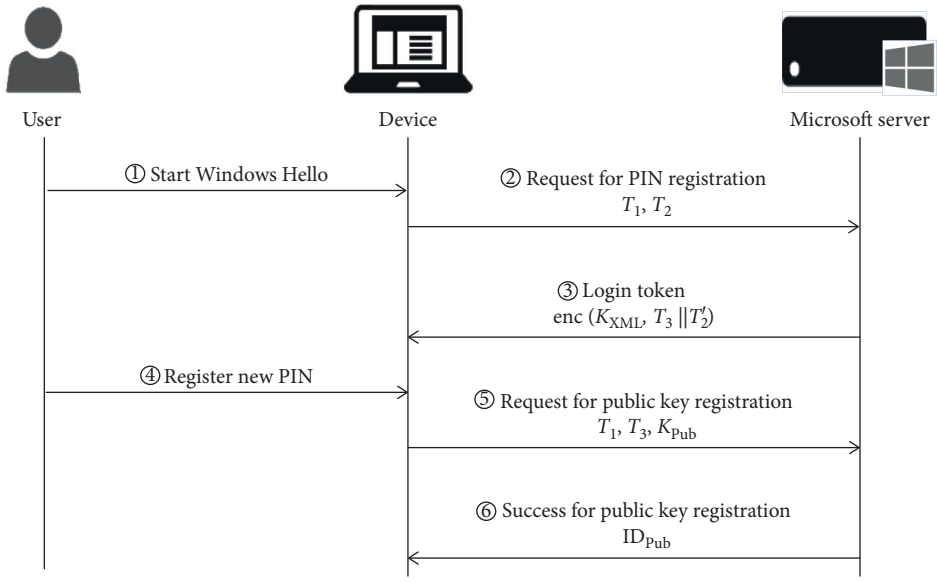

Figure 4: Detailed protocol flow: setting up the gesture. Mutual authentication is performed between the user and the server, which involves verifying the user with the device and authenticating the device with the Microsoft server. Through this process, the user's account and the device are bound.

4.3. Step 3: Log In to Application with Windows Hello. At this point, the user's account has been set up and the user is ready to partake in services provided by Microsoft and third parties. Figure 5 shows an application login protocol. Before the user accesses the services, the user must log in to the Microsoft server for authorization. The Microsoft server challenges the user with a random number R in the second message and expects a digital signature of the challenge. The private key is encrypted with the gesture. Only the user with the valid gesture has the private key, and only that person is able to generate the digital signature. The Microsoft server verifies the signature in the fourth message with the corresponding public key.

The server returns an authorization, which is composed of two tokens encrypted with a session key that is generated during the sign-in phase of the Microsoft account and shared between the device and the Microsoft server. The first token, T_4, is an access token that provides access to a service. The second token, T_2'', is the second renewal of an account identifier. Once the access token sent in the sixth message is authenticated by the service provider, the user can begin using the service.

This protocol analysis enables us to complete the first stage in the migration attack detailed in Figure 2. In the process of logging in to the application using Windows Hello, the Microsoft server authenticates the user and the device with an account token (T_2) and a private key (K_{Pri}). Those three pieces of information are saved in a device, more specifically in the file system. The attacker needs to identify T_2 and K_{Pri}, which contain information on the format and storage location of the device. The attacker knows that K_{Pri} is encrypted with the gesture. The gestures used to encrypt K_{Pri} include a registered PIN, biometrics, and security keys. We focus on how the PIN is used and how the attacker reveals the PIN in the victim's device.

5. Data Management in Windows Hello

In this section, we discuss the storage and use of the account token and private key and show how the attacker can extract them from the victim's device and calibrate them to fit the attacker's device. After the account setup described in ⑤ in Figure 4, Windows Hello ends up owning three additional pairs of the asymmetric key. These keys have the same structure but are saved in different files. A private key file consists of four elements: (1) RSA key identifier, (2) public key, (3) private key properties such as salt and an iteration count for encrypting or decrypting, and (4) encrypted binaries of the private key.

Windows 10 supports a set of interfaces for handling private keys in a secure manner. These interfaces are defined and declared under a subsystem called the Data Protection Application Programming Interface (DPAPI) [22]. DPAPI protects sensitive data in secure storage locations on the disk by encrypting that data with keys derived from a local password. The *(un)protect* function is the implementation of DPAPI operation. The *(un)protect* function allows only the user who has the password for device login to encrypt or decrypt the data and assess its integrity.

$$KEK^i = SHA512\,(PBKDF2\,(PIN\ or\ seed, salt, iteration)). \quad (1)$$

$$K^i_{\text{Pri}} = unprotect\left(KEK^i, EncPriKey\right). \quad (2)$$

Figure 6 presents the procedure used to decrypt a private key (K^i_{Pri}) from the private key file. A key encryption key KEK^i is computed from the PIN, as shown in equation (1). The *PBKDF2* function increases the difficulty of a brute-force attack by iterating over cryptographic secure hash functions. The values of salt and iteration used in equation (1) are extracted from private key properties that are decrypted from the private key file with a DPAPI key of a Windows account. The result of the *PBKDF*2 function is used to compute a key encryption key KEK^i via the *SHA*512 function. The key encryption key decrypts the private keys of the encrypted key file through the *unprotect* function in equation (2).

Figure 7 illustrates the procedure used to decrypt four private keys in the Windows Hello login. Table 1 shows the main functions used in the authentication process, while Table 2 shows where authentication-related data is stored.

5.1. ①, ② Enter the PIN and Decrypt the Private Keys. The user-supplied PIN is a secret key that is used for the decryption of the first private key (K^1_{Pri}), which further decrypts *seed* in a key metadata file. *seed* is randomly generated when the gesture is set up, and it is used as a secret key to decrypt the second, third, and fourth private keys in the private key files in equation (1).

5.2. ③-❶ Device Login with a Local Account. Local account users are authenticated with an *EncPwd* in the registry. This is an encrypted local password with a second public key. In a decryption procedure, the second private key (K^2_{Pri}) is derived from the *seed* and, thus, from the user's PIN. This is tantamount to saying that the *EncPwd* is decrypted with the PIN. The output of decryption is used as an input to an authentication process for a login in Windows called the New Technology LAN Manager (NTLM) authentication process [23], where a hash of the user's password is stored. If the two passwords match, authentication succeeds.

5.3. ③-❷ Device Login with a Microsoft Account. Authentication for Microsoft account users is quite similar to that for local account users, except that the third private key (K^3_{Pri}) is used to decrypt an *AuthTicket* in the *CacheData* file. The Microsoft authentication server issued the *AuthTicket* to the user when the Microsoft account was registered. *AuthTicket* is used for the user authentication process with the Microsoft server. The device sends *AuthTicket* to the Microsoft server; if *AuthTicket* is valid, authentication is successful.

5.4. ③-❸ Application Login with Windows Hello. The Windows Hello service signs in a random challenge value received from the Microsoft server with the fourth private

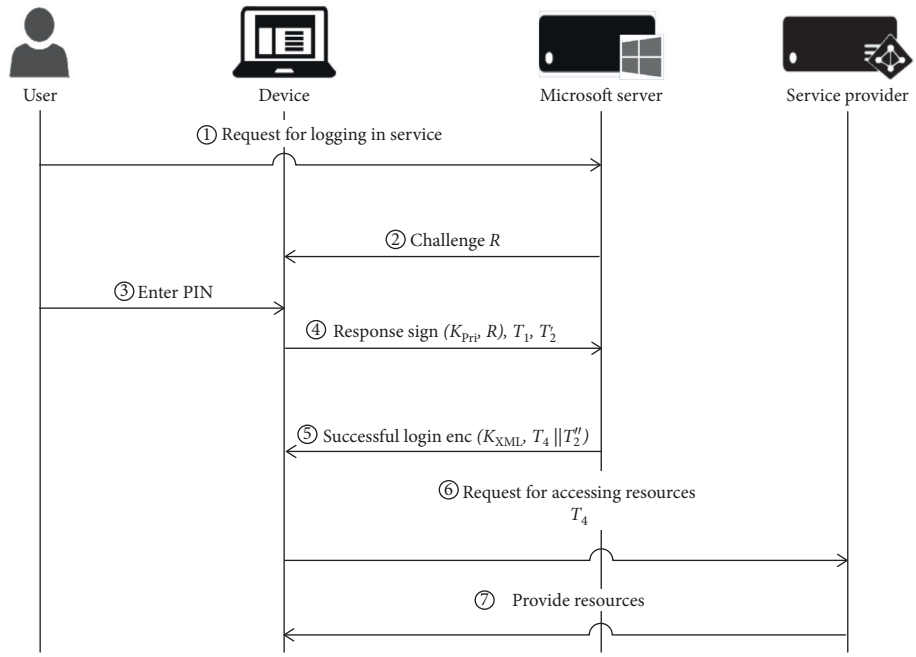

FIGURE 5: Detailed protocol flow: application login with Windows Hello. The user and the device are authenticated to obtain the access token (T_4) for use by the service provider.

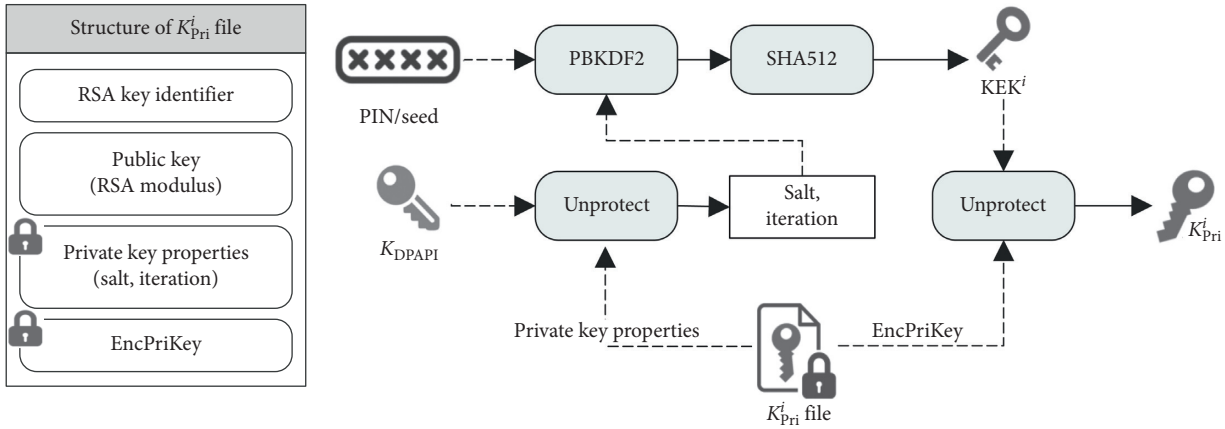

FIGURE 6: Structure and decryption procedure of private key K^i_{Pri} files. To decrypt K^i_{Pri}, the key encryption key KEK^i is computed in the PIN or seed. (a) Structure of key file. (b) Decrypting procedure of private keys.

key (K^4_{Pri}). As discussed in Section 3.1, the fourth private key is registered to the Microsoft server when the gesture is set up (⑤ in Figure 4). The Windows Hello service gets T_2 from the credential storage location and sends it to the Microsoft server. After the Microsoft server validates the signed challenge and the token, the Windows Hello service succeeds in logging in to the application and receives the renewed T_2' from the server.

In Windows 10, credential storage is the repositories that encrypt and store security essentials used in the user's local, network, and web accounts. T_2, which is issued by the Microsoft server, is stored in credential storage. DPAPI is used to protect the data in credential storage.

6. Details of the Migration Attack

As discussed in the above sections, Windows Hello works based on private keys that are encrypted with the user's PIN. To migrate the authentication data for Windows Hello, the attacker must first figure out the PIN and then extract the keys by decrypting them with the PIN. The victim's authentication data is transferred to the attacker's device, calibrated on the attacker's device, and used by the attacker to access the victim's services.

6.1. PIN Cracking. Malware that is injected into a local device can be used to access private key files on the disk. The attacker launches a brute-force guessing attack on these files

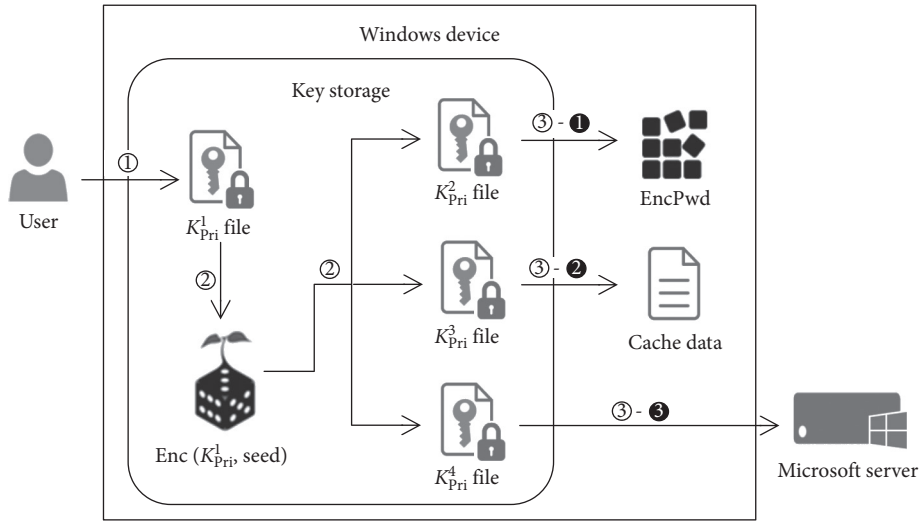

Figure 7: Windows Hello login phases showing the roles of the four private keys: ① enter the PIN, ② decrypt private keys, ③-❶ log in to the device with the local account, ③-❷ log in to the device with the Microsoft account, and ③-❸ log in to the application with Windows Hello.

Table 1: Main functions used in the Windows Hello login process. We focus on the role that each function plays in the login process.

Module	Function	Descriptions
BCrypt.dll	*BcryptDeriveKeyPBKDF2*	Derive a key encryption key through hashing for PIN or *seed*
Crypt32.dll	*CryptUnprotectDataNoUI*	Decrypt all private keys with key encryption keys derived from *BcryptDeriveKeyPBKDF2* function
NCrypt.dll	*NcryptOpenKey*	Generate key handler based on a key name
CryptSvc.dll	*NgcDecryptData*	Decrypt the authentication data for a device login
NgcCtnr.dll	*DecryptPkcs1*	Decrypt *seed* with the first private key managed by the software key storage provider
	SignHashPkcs1	Sign challenge *R* with the fourth private key for application login

Table 2: Storage of authentication data for Windows Hello.

Data	Path
Private keys	*%SystemRoot%\ServiceProfiles\LocalService\AppData\Roaming\Microsoft\Crypto\Keys*
Key metadata	*%SystemRoot%\ServiceProfiles\LocalService\AppData\Local\Microsoft\Ngc*
EncPwd	*\HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Authentication\LogonUI\NgcPin\Credentials\EncryptedPassword*
CacheData	*%SystemRoot%\System32\config\systemprofile\AppData\Local\Microsoft\Windows\CloudAPCache\MicrosoftAccount*
Credentials	*%LOCALAPPDATA%\Microsoft\Credentials*

to find the PIN. The PIN is an essential target of the attack as it is used to decrypt private keys. The guessing attack works with offline files, allowing the attack to launch repeatedly without any pauses between attempts and without any limits on the number of attempts. We have developed a tool for PIN cracking. Section 5.2 describes the tool in further detail.

6.2. Migration of Authentication Data. After decrypting the file with the PIN, the attacker extracts the first private key. The attacker also decrypts three other private keys with *seed*. The attacker extracts most authentication data based on the private keys. The extracted data includes the private key identifiers, *seed*, *CacheData*, registry, and credentials. The data is decrypted from the filesystem and registry on the victim's device and transmitted to the attacker's device.

Some data is protected by the DPAPI mechanism. DPAPI works under security identifiers (SID) [11]. An SID is an identifier that is assigned to identify the user in the system or to grant permissions to users in Windows 10. Data that is encrypted by DPAPI cannot be decrypted without the SID that was used for encryption. When the device needs to use protected data, that data is decrypted using the *unprotect* function executed by the process that is granted by a specific SID.

The Windows Hello service uses DPAPI to protect all private keys with the local authority, SID_{Local}. SID_{Local}

includes all users who have logged in locally. Data that is encrypted using SID_{Local} is accessible to all users who are currently logged in to the device, without any authentication process. T_2 is also protected by DPAPI and is stored in credential storage. Credentials are encrypted with the LocalSystem authority, SID_{System}, which has full access to the system. The data encrypted using SID_{System} is only accessible to a user with administrator privileges.

The malware running on the victim's device works only at the precise moment the authentication data is extracted. After the authentication data is extracted, the malware no longer needs to run on the victim's device. This makes the attack difficult to detect. Moreover, using DPAPI decryption tools such as Mimikatz [24], the attacker can obtain all authentication data even when the victim's device is not activated and only the disk image is acquired.

The attacker sends the extracted authentication data to the attacker's device. The attacker then calibrates it to be used by the Windows Hello service running on the attacker's device. At this point, the attacker's device is logged in to the attacker's Microsoft account and Windows Hello is enabled. The extracted authentication data replaces the attacker's data. These are also based on the PIN and the first private key. All private keys and credentials are protected by DPAPI using the proper SID on the attacker's device.

Finally, the attacker attempts to log in to the application with the victim's account. All authentication data contains the victim's information, and the Microsoft server recognizes the attacker's device as the victim's device. The attacker enters the victim's PIN and then accesses the application with the victim's account.

7. Evaluation

7.1. Effectiveness of the Migration Attack.

(1) *Similarities to Hijacking of a Microsoft Account*. A typical session hijacking attack is performed by obtaining session data, such as cookies or tokens, and deceiving the server using that data. The scope of a session hijacking attack is limited to the web services that use the hijacked session data; after the session expires, the attack will not be maintained.

On the other hand, in a Windows Hello migration attack, the victim's authentication data is transferred to the Microsoft server and used to impersonate the victim. Thus, this attack is similar to stealing a Microsoft account. This attack can lead to the hijacking of all web services that are accessible through Windows Hello. Several apps, such as Store, OneDrive, Office, Skype, and Outlook, deal with sensitive information that could be a problem if it is leaked. As the number of applications that are accessible through Windows Hello is expected to continue to increase in the future, the ripple effects of such an attack are expected to increase as well.

(2) *Stealth*. The victim may recognize that their authentication data for Windows Hello login has been leaked and may make changes or remove the PIN from the affected device. However, the attacker can still access Windows Hello applications using the victim's account. This is possible because the Windows Hello service does not communicate with the Microsoft server when the PIN is removed or changed.

(3) *Bypassing Two-Factor Authentication*. The migration attack appears valid even if the user has activated Microsoft authenticator. The tokens that are obtained through the migration attack were previously approved by Microsoft authenticator. If the attacker uses Windows Hello after migrating the victim's authentication data to the attacker's device, the victim's mobile device does not receive an approval notification or alert. As a result, it is possible to steal the victim's Microsoft account using only the Windows Hello authentication data by bypassing two-factor authentication.

(4) *Privilege Escalation for Changing Login Password*. The attacker cannot steal the login credentials that are used to log in to a Windows 10 device with the Microsoft account, even if the attacker has administrator privileges on the device. In Windows 10, a user with administrator privileges can change the passwords of other users who log in to the local account. However, the user cannot change the password of a device that is logged in to the Microsoft account. The password for the Microsoft account cannot be found on the victim's device. However, the attacker can access the PIN and the authentication data that are used in lieu of a password in Windows Hello.

7.2. Efficiency for PIN Cracking. Microsoft introduced a secure password-based key derivation function (PBKDF) to protect encrypted key files from brute-force attacks [25]. This function mitigates a vulnerability to the brute-force attack by increasing the computational cost of generating encryption and decryption keys. The PBKDF2 function in Windows Hello, which is used when generating a key encryption key KEK^1 for an encrypted key file, increases the time taken by a brute-force attack by repeating the hash function 10,000 times. This makes it difficult to find the PIN within a realistic time.

We measured the feasibility of using a PIN-cracking tool to find the PIN through brute-force attack. The tool takes about one second to decrypt an encrypted key file 100 times with a single-core processor. Table 3 shows the amount of time it takes a brute-force attack to crack a PIN of four to eight digits. For a six-digit PIN, the tool can try every possible combination within three hours. With a multicore processor or Graphics Processing Units (GPUs), it might be possible to hijack PINs with less than ten digits in a realistic timeframe.

One of the reasons that it is possible to crack a PIN is that it is a numeric password. It has a minimum of four digits to a maximum of 20 digits. In PIN-based authentication systems, about 99 percent of users use PINs within ten digits [26].

Table 3: Brute-force attack duration according to PIN length.

# of digits	4	5	6	7	8
Time	1.7 min	17 min	2.8 hours	1.1 days	11 days

Some PINs mix letters and special characters with digits, but this compromises convenience, which is one of the greatest advantages of a PIN. With a complex PIN, the user experiences the same usability issues as they do with password authentication.

If the attacker performs a brute-force attack on an online system, the attack is restricted to a certain period. This is because login attempts cannot be performed more than a predefined number of times. The proposed attack is performed by an attacker who holds an encrypted key file offline. Unlike an online brute-force attack, there is no limit to the number of attack attempts that can be performed. Therefore, this is a suitable environment for a brute-force attack.

7.3. Target Application. We performed case studies on applications for which a victim's account can be stolen through a migration attack. The target applications support a single sign-in service, which, after a user logs in with a Microsoft account on a Windows device, allows automatic logins without additional account credentials. An attacker can access most services used after the victim logs in for target applications:

(1) *Office 365*. Office 365 is a set of cloud-based document-sharing services provided by Microsoft. Office 365 contains popular document editors such as PowerPoint, Excel, Word, and OneNote. All documents created through Office 365 applications are shared to the cloud via OneDrive. OneDrive provides cloud storage for not only documents created through the Office 365 application but also user data. An attacker who accesses the Office 365 service through the migration attack obtains all data stored in the victim's OneDrive. The attacker can steal or modify data on OneDrive. Also, by uploading new files such as malware files, the attacker can take over any of the victim's other devices that are synchronized.

(2) *Microsoft Store*. Microsoft Store is an online market for purchasing applications that are available on Windows desktops. An attacker who accesses Microsoft Store through a migration attack can see the victim's wish list and library of previous purchases. The attacker can redownload the purchased applications. If payment information was added to the victim's account, the attacker can purchase other applications with that payment method.

(3) *Edge Browser*. Edge browser is a web browser that is provided by default in Windows. Edge browser provides a synchronization service that maintains browser settings across different devices and operating systems. An attacker who accesses the Edge browser through a migration attack can steal website passwords, history, and payment information.

(4) *Single Sign-On Website*. An attacker can access websites that allow single sign-on login using a victim's Microsoft account. When accessing such websites through a web browser, a user can sign in with Windows Hello by selecting the option to log in with a Microsoft account. The option for logging in to websites using Windows Hello is supported by all major browsers such as Google Chrome, Microsoft Edge, and Mozilla Firefox. The attacker can impersonate the victim to access the services provided by the websites.

8. Related Works

FIDO2 is a successor to FIDO1 [27]. FIDO2 extends FIDO1's coverage from mobile devices to web services and various operating systems. Along the same lines as FIDO2, FIDO1 uses an asymmetric key pair based on trusted platform computing to authenticate a user without exposing the user's password. FIDO1 provides two subprotocols: (1) the Universal Authentication Framework (UAF) protocol, which plays a similar role to that of FIDO2's web authentication protocol, and (2) the Universal Second Factor (U2F) protocol, which plays a role similar to that of FIDO2's CTAP. Most security threats in FIDO1, which operates under the same conceptual framework as FIDO2, can be applied equally to FIDO2. Along with focusing on the security analysis and possible attacks on FIDO2, we also investigated related works about FIDO1 which may be relevant to FIDO2 security.

Table 4 summarizes previous works that are relevant to our study. We were interested the protocol, analysis method, and attack model for security analysis on which related works are focused. Most studies focus on the formal or informal analysis of protocol design. A few studies on FIDO implementation were conducted for the Android platform and U2F-supported websites. Attack models can be divided into six categories: (1) malicious or curious server, (2) a network attacker who can intercept, forge, or send all messages, (3) a web attacker who intervenes in the protocol via the web as a third party not directly participating in the authentication, (4) a local attacker who has full control of a Trusted Execution Environment- (TEE-) unsupported user device, (5) a local attacker who has full control of a TEE-supported user device, and (6) a user who wrongly performs some actions or ignores security warnings. The remainder of this section summarizes what threats each study derives from which attack models for the analysis target.

First, informal analyses of the UAF protocol were conducted. Hu and Zhang [28] pointed out the problem of unauthenticated software communication with the FIDO authenticator, which allows the malware to impersonate benign software in order to hijack credentials within the FIDO authenticator. Similarly, Panos [29] used informal analysis to present attack vectors that exist in the UAF protocol. This study points to a major potential problem: a

Table 4: Comparison of security properties of related works. Our focus is on the threats and attack scenarios presented by the studies and whether the attacks are feasible. We investigated the protocol, analysis method, analysis target, and attack model targeted for previous works.

Item	Property	[28]	[29]	[30]	[31]	[32]	[33]	[34]	[35]	[36]	[37]	[38]	[39]
Protocol	UAF/WebAuthn	•	•	•	•					•		•	•
	U2F/CTAP					•	•	•	•	•	•		
Analysis target	Design	•	•	•	•	•	•	•	•	•			•
	Implementation										•	•	•
Analysis method	Formal			•	•				•	•			•
	Informal	•	•			•	•	•			•	•	
Attack model	Server attacker			•	•	•	•	•			•		
	Network attacker				•			•		•	•		
	Web attacker			•			•						
	Local attacker (without TEE)	•	•		•	•			•	•	•		
	Local attacker (with TEE)		•		•	•		•				•	•
	Human error								•				

trusted computing platform that manages FIDO credentials can be replicated or bypassed by an authorized attacker.

Several works use formal analysis to examine potential security threats in the design of the UAF or WebAuthn protocol. Guirant and Halpin [30] discussed the privacy issue involved with identifying a user in a different server when one authenticator is used in two web servers simultaneously. Xu [31] focused on the environment in which the UAF protocol is based, the Trusted Execution Environment (TEE), which is used, for example, in the TPM chip. Alaca and Oorschot [32] warned of possible threats in which authentication data is extracted from a disk or memory by malware on user devices.

FIDO U2F has also been analyzed in both informal and formal manners. Chong [33] presented an analysis of the U2F protocol, including a discussion on possible attacks. The U2F protocol uses the channel ID and origin site as challenge values to prevent password reuse, phishing, and Man-in-the-Middle (MITM) attacks. However, if the FIDO client is compromised, the protocol remains vulnerable to MITM and Denial of Service (DoS) attacks. An attack vector that compromises the FIDO client is theoretically similar to our attack vector, but previous works do not provide details of the attack scenarios. Pereira et al. [34] and Jacomme and Kremer [35] modeled the FIDO authentication process and formally analyzed the protocol phase by phase using the ProVerif tool. They validated security of various threat scenarios that may occur in practical environments. They discussed an attack model that is related to our work in which a victim's platform is compromised and an attacker can control FIDO1 authentication with the victim's permissions. Barbosa [36] performed a formal analysis of both WebAuthn and CTAP, where cryptographic components required for the use of FIDO2 securely through a provable security analysis were defined.

Few studies have investigated the implementations of FIDO. Patat and Sabt [37] targeted YubiKey [40], which is an implementation of the U2F protocol. They focused on a feature called "remember me," which allows a device to log in with only a password after initial U2F authentication to make use of U2F authentication less of a hassle. They demonstrated the feasibility of exploiting the feature by logging in to the Facebook website from a device that has not performed the U2F authentication. Li et al. [38] analyzed the UAF protocol implemented on the Android platform. Based on the analysis, they proposed a rebinding attack, which binds the victim's FIDO identifier to the attacker's FIDO authenticator and accesses the service with the victim's account. This study points out that the feasibility of the rebinding attack is due to the lack of proper authentication between the FIDO authenticator and the FIDO server. Feng [39] formalized the security model and protocol for various scenarios and then developed an automated verifier which performs security analysis and identifies design flaws according to the security assumptions and goals. Unlike the present paper, that work does not focus on how authentication data is managed on authenticators.

9. Conclusion

In this work, we presented a detailed analysis of Windows Hello implementation on Microsoft Windows 10. This work provided the first empirical study of the security of the Windows login system recommended by Microsoft for next-generation devices, the FIDO2 web authentication protocol. We explore the security issues, which are mainly due to the differences between the ideal environment, in which Windows Hello is completely secure, and the actual operating environment. Based on our analysis of Windows Hello authentication, we propose the Windows Hello migration attack to prove the feasibility of our attack scenario. We hope this paper will help researchers reconsider the implementation environment of the FIDO2 authentication protocol.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the Institute for Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korean Government (MSIT) (no. 2019-0-01343, Regional Strategic Industry Convergence Security Core Talent Training Business).

References

[1] FIDO Alliance, "FIDO2: WebAuthn & CTAP," 2018, [Online]. Available: https://fidoalliance.org/fido2/.

[2] M. Docs, "Windows hello for business overview," [Online]. Available: 2020, https://docs.microsoft.com/windows/security/identity-protection/hello-for-business/hello-overview.

[3] M. Docs, "Trusted platform module technology overview," 2018, [Online]. Available: https://docs.microsoft.com/windows/security/information-protection/tpm/trusted-platformmodule-overview.

[4] K. Mayes, "An introduction to smart cards," *Smart Cards, Tokens, Security and Applications*, Tokens, Security and Applications, London, UK, 2017.

[5] S. W. Shah and S. S. Kanhere, "Recent trends in user authentication - a survey," *IEEE Access*, vol. 7, no. 1, pp. 112505–112519, 2019.

[6] World Wide Web Consortium (W3C) and W. Authentication, "An API for accessing public key credentials level 2,"W3C Candidate Recommendation, 2020, [Online]. Available: https://www.w3.org/TR/webauthn/.

[7] FIDO Alliance, "Client to authenticator protocol (CTAP) proposed standard," 2019, [Online]. Available: https://fidoalliance.org/specs/fido-v2.0-ps-20190130/fido-client-toauthenticator-protocol-v2.0-ps-20190130.pdf.

[8] M. Docs, "Why a pin is better than a password," 2017, [Online]. Available: https://docs.microsoft.com/windows/security/identity-protection/hello-for-business/hello-why-pin-isbetter-than-password.

[9] M. Docs, "Enable passwordless sign-in with the microsoft authenticator app," 2020, [Online]. Available: https://docs.microsoft.com/azure/active-directory/authentication/howto-authentication-passwordless-phone.

[10] M. Docs, "Access control: understanding windows file and registry permissions," 2019, [Online]. Available: https://docs.microsoft.com/en-us/archive/msdn-magazine/2008/november/access-control-understanding-windows-file-andregistry-permissions.

[11] M. Docs, "Security identifiers," 2017, [Online]. Available: https://docs.microsoft.com/windows/security/identity-protection/access-control/security-identifiers.

[12] G. Hull, H. John, and B. Arief, "Ransomware deployment methods and analysis: views from a predictive model and human responses," *Crime Science*, vol. 8, no. 2, pp. 1–22, 2019.

[13] S. Sibi Chakkaravarthy, D. Sangeetha, and V. Vaidehi, "A survey on malware analysis and mitigation techniques," *Computer Science Review*, vol. 32, no. 1, pp. 1–23, 2019.

[14] M. Docs, "How user account control works," 2018, [Online]. Available: https://docs.microsoft.com/en-us/windows/security/identity-protection/user-account-control/how-useraccount-control-works.

[15] K. Oosthoek and C. Doerr, "SoK: ATT&CK techniques and trends in Windows malware," in *Proceedings of 15th International Conference On Security And Privacy In Communication Systems (SecureComm)*, Orlando, FL, USA, October 2019.

[16] M. J. Haber, "Privilege escalation," *Handbook of Privileged Attack Vectors: Building Effective Cyber-Defense Strategies To Protect Organizations*, 2020.

[17] A. T. T. Mitre, "C. K., "Privilege escalation" 2021, [Online]. Available: https://attack.mitre.org/tactics/TA0004/.

[18] CVE, "Common vulnerabilities and Exposures list," 2021, [Online]. Available: https://cve.mitre.org/cve/.

[19] Word Wide Web Consortium (W3C), "XML protocol activity", W3C's architecture domain," 2000, [Online]. Available: https://www.w3.org/2000/xp/.

[20] Word Wide Web Consortium (W3C), "XML encryption syntax and processing version 1.1,"W3C Recommendation, 2013, [Online]. Available: https://www.w3.org/TR/2013/REC-xmlenc-core1-20130411/.

[21] Word Wide Web Consortium (W3C), "XML signature syntax and processing version 1.1,"W3C Recommendation, [Online]. Available: 2013, https://www.w3.org/TR/xmldsig-core1/.

[22] E. Burzstein and J. Picod, "Recovering Windows secrets and EFS certificates offline," in *Proceedings of 4th USENIX Workshop on Offensive Technologies (WOOT)*, Washington, DC, USA, Augest 2010.

[23] M. Docs, "Microsoft NTLM," 2018, [Online]. Available: https://docs.microsoft.com/windows/win32/secauthn/microsoft-ntlm.

[24] B. Delpy, "Mimikatz," 2020, [Online]. Available: https://github.com/gentilkiwi/mimikatz.

[25] B. Kaliski, "PKCS #5: password-based cryptography specification version 2.0," *IETF RFC*, vol. 2898, 2000.

[26] DataGenetics, "Distribution of all-numeric passwords based on length," 2012, [Online]. Available: http://www.datagenetics.com/blog/september32012/.

[27] FIDO Alliance, "Specifications overview," 2014, [Online]. Available: https://fidoalliance.org/specifications/.

[28] K. Hu and Z. Zhang, "Security analysis of an attractive online authentication standard: FIDO UAF protocol," *China Communications*, vol. 13, no. 12, pp. 189–198, 2016.

[29] C. Panos et al., "A security evaluation of FIDO's UAF protocol in mobile and embedded devices," in *Proceedings of 28th International Tyrrhenian Workshop On Digital Communication*, Palermo, Italy, September 2017.

[30] I. Guirant and H. Halpin, "Formal verification of the web authentication protocol," in *Proceedings of 5th Annual Symposium And Bootcamp On Hot Topics In the Science Of Security (HoTSoS)*, Raleigh, NC, USA, April 2018.

[31] S. Xu et al., "A symbolic model for systematically analyzing TEE-based protocols," in *Proceedings of 22nd International Conference On Information And Communications Security (ICICS)*, Copenhagen, Denmark, Augest 2020.

[32] F. Alaca and P. Oorschot, "Comparative analysis and framework evaluating web single sign-on systems," *ACM Computing Surveys*, vol. 53, no. 5, p. 112, 2020.

[33] J. Chong, "Breaking FIDO: are exploits in there?" in *Proceedings of the 19th Blackhat*, Las Vegas, NV, USA, August 2016.

[34] O. Pereira, F. Rochet, and C. Wiedling, "Formal analysis of the FIDO 1. x protocol," in *Proceedings of 10th International Symposium On Foundations And Practice Of Security*, FPS), Nancy, France, October 2017.

[35] C. Jacomme and S. Kremer, "An extensive formal analysis of multi-factor Authentication protocols," *ACM Transactions on Privacy and Security*, vol. 24, no. 2, pp. 13–34, 2021.

[36] M. Barbosa et al., "Provable security analysis of FIDO2," Cryptology ePrint Archive Report 2020/756, 2020 [Online]. Available: https://eprint.iacr.org/2020/756.pdf.

[37] G. Patat and M. Sabt, "Please remember me: security analysis of U2F remember me implementations in the wild," in *Proceedings of 18ème Symposium sur la sécurité des technologies de l'information et des communications*, SSTIC), Rennes, France, June 2020.

[38] H. Li, X. Pan, X. Wang, H. Feng, and C. Shi, "Authenticator rebinding attack of the UAF protocol on mobile devices," *Wireless Communications and Mobile Computing*, vol. 2020, no. 1, 14 pages, Article ID 8819790, 2020.

[39] H. Feng et al., "A formal analysis of the FIDO UAF protocol," in *Proceedings of 28th Network And Distributed System Security Symposium (NDSS)*, San Diego, CA, USA, February 2021.

[40] Yubico, "YubiKey," 2021, [Online]. Available: https://www.yubico.com/products/.