Please Remember Me: Security Analysis of U2F Remember Me Implementations in The Wild

Gwendal Patat and Mohamed Sabt
gwendal.patat@irisa.fr
mohamed.sabt@irisa.fr

Univ Rennes, CNRS, IRISA

Abstract. Users and service providers are increasingly aware of the security issues that arise because of password breaches. Recent studies show that password authentication can be made more secure by relying on second-factor authentication (2FA). Supported by leading web service providers, the FIDO Alliance defines the Universal 2nd Factor (U2F) protocols, an industrial standard that proposes a challenge-response 2FA solution. The U2F protocols have been thoughtfully designed to ensure high security. In particular, U2F solutions using dedicated hardware tokens fare well in term of security compared to other 2FA authentication systems. Thus, numerous service providers propose U2F in their authentication settings.

Although much attention was paid to make U2F easy to use, many users express inconvenience because of the repeated extra step that it would take to log in. In order to address this, several service providers offer a *remember me* feature that removes the need for 2FA login on trusted devices. In this paper, we present the first systematic analysis of this undocumented feature, and we show that its security implications are not well understood. After introducing the corresponding threat models, we provide an experimental study of existing implementations of *remember me*. Here, we consider all the supporting websites considered by Yubico.

The findings are worrisome: our analyses indicate how bad implementations can make U2F solutions vulnerable to multiple attacks. Moreover, we show that existing implementations do not correspond to the initial security analysis provided by U2F. We also implement two attacks using the identified design flaws. Finally, we discuss several countermeasures that make the *remember me* feature more secure.

We end this work by disclosing a practical attack against Facebook in which an attacker can permanently deactivate the enabled 2FA options of a targeted victim without knowing their authentication credentials.

1 Introduction

Passwords are by far the most popular method of end-user authentication on the web. Password breaches, due for instance to sophisticated phishing attacks, seriously multiply the risk of authentication compromise.

Worse, these risks are compounded by poor user practices, as illustrated by reusing passwords across multiple accounts [26]. Aware of such security issues, a growing number of service providers couple passwords with another authentication factor: this is known as second factor authentication (2FA). The use of 2FA is not new. However, recent studies show that their adoption rate is stagnant over the last five years because of a fragmented ecosystem [5]. Furthermore, most users still use their mobile phone as second factor which can lead to critical security issues like what happened with Twitter CEO Jack Dorsey [17].

In this context, Google and Yubico join the effort within the FIDO Alliance to standardize a set of 2FA protocols that they call the Universal Second Factor (U2F) authentication. Their design ambition is to guarantee a strong security level (through hardware tokens and cryptographic operations) while maintaining a pleasant user experience. Several formal analyses theoretically show that the U2F authentication protocol actually achieves its security goals [15, 21]. Google describe U2F as an opportunity to "*improve the state of the art for practical authentication for real consumers*" [16].

The starting point of our work is to underline a practical aspect within the U2F implementations that is often overlooked. Indeed, some U2F solutions provide a *remember me* feature that, once enabled, allows users to authenticate to a given website without presenting their U2F token on a given device. According to [7], this feature is important for better user acceptability. However, to the best of our knowledge, no document specifies how U2F and *remember me* shall be combined without negatively affecting the security of the resulted protocol.

We notice that the lack of specifications of the *remember me* feature brings confusion and contributes to the misconception of U2F solutions. Thus, motivated by Bonneau's framework [4], we start our work by defining a set of security properties upon which we construct our analysis. We also recall the threat models that should be considered when performing security evaluations of U2F.

In our work, we achieve the first study about the impact of the *remember me* feature on the overall security. In particular, we inspect the related implementations of all the websites proposing *remember me* that are included in the Yubico inventory [30] (67 websites in total). Interestingly, we have found that the analyzed U2F solutions use very diverse implementation choices. Then, we uncover several design aspects that can lead to several attacks. For instance, we show that U2F solutions become vulnerable to Man-in-the-Middle attacks, contrary to what is

suggested by FIDO [18]. This is unfortunate, since the U2F protocols are designed to protect against such powerful attackers. We verify the soundness of our analyses by conducting several attacks using the identified flaws. In this regard, our paper invalidates the results of [21] and [15] that formally prove the security of U2F protocols.

Our analysis has also pointed out problems related to the recovery methods defined for lost FIDO authenticators. In particular, we disclose a practical attack against Facebook in which attackers are able to permanently deactivate the enabled U2F option in the victims accounts, thereby removing any interest of using U2F. To our surprise, attackers do not need users credentials in order to achieve their attacks; they solely need to get a code received by SMS on the victim's mobile phone. In summary, this paper makes the following contributions:

— we define a general threat model for U2F implementations;
— we analyze all *remember me* solutions, we report undocumented details about their underlying implementations and we evaluate them using our secure design rules;
— we identify attacks and weaknesses in the analyzed websites. For each attack, we provide a proper attacker model and point out the corresponding violations regarding our model;
— we implement effective scenarios in which attackers can successfully bypass the second factor using a device that has never been remembered before;
— we expand our analysis to include the recovery mechanisms. Here, we show how U2F fails its security goals to replace other 2FA solutions. In particular, we reveal unpublished effective attack against Facebook and show its actual practicality. Impatient readers can refer to our demo video.[1]

2 Related Work

2.1 2FA Solutions

For a long time the community has been looking beyond passwords to ensure the security of users accounts against credentials breaches [1, 29]. The main trend is to employ two-factor authentication, namely to ask users to prove the possession of "something they have" in addition to passwords "something they know".

1. The demo is available as an unlisted YouTube video: `https://youtu.be/OrQJ7JSyhsI`

One-time-password (OTP) codes through SMS are one of the 2FA first methods. Despite their popularity, their use is being recently discouraged by the NIST (National Institute of Standards and Technology) [20]. GoogleAuthenticator [22], installed more than 10,000,000 from Google Play, is an Android app that locally generates OTP codes relying on specific algorithms [12, 14]. However, these OTP schemes fail to protect against phishing attacks.

Hardware tokens [13, 24] are deployed to secure accounts, especially for online banking. Those tokens periodically generate an OTP that is valid for a short period of time. Unfortunately, these OTP tokens do not counter phishing attacks neither. More advanced solutions implement challenge-response 2FA protocols, such as FIDO U2F [25] and PhoneAuth [8], and thus offer protection against phishing attacks. In spite of their security benefits, hardware-backed 2FA solutions suffer from stagnant adoption, as illustrated by Elie Bursztein in his blog [5]. This is because such solutions may be expensive to deploy, and there might be usability issues if the user authenticates in a machine with a USB-C port while she owns a USB-A token, or vice versa.

2.2 2FA Usability

Since the seminal work of Bonneau et al. [4], much two-factor authentication research has recognized the need for user-friendly implementations to promote adoption. Indeed, Bonneau et al.'s high-level evaluation rated existing two-factor systems as more secure, but generally less usable than passwords. In response to such concern, subsequent 2FA solutions have strived for better usability, hence FIDO U2F protocols. Lang et al. [16] fervently promote the usability of YubiKeys in enterprise environments. However, later results show that the Google experiments were somehow biased, and complement them with more qualitative evaluation.

Indeed, Das et al. [9] distinguish usability from acceptability. Based on users feedback, authors find out that improving usability does not necessarily lead to greater acceptability. Reynolds et al. [23] yield more insights into the usability of YubiKeys in conducting a laboratory and longitudinal experiments in a non-enterprise environment. They noticed, on one hand, that participants have struggled to set their YubiKeys as a second factor of authentication. On the other hand, they were positive regarding YubiKeys and 2FA in general. For wide adoption, authors in [7] state that almost all participants in their usability survey highlight that using *remember me* made using 2FA solutions more pleasant, especially when regularly accessing the same accounts from the same devices.

2.3 U2F Security Analysis

Bonneau et al. [4] study the security of different web authentication protocols: they provide a set of security *benefits* that they informally define in order to capture the most common security threats for authentication services. However, its informal nature raises some issues in the security community.

In 2017, Pereira et al. presented the first formal analysis of FIDO U2F authentication protocol. They modeled the protocol using applied pi-calculus and prove their model using ProVerif. Their analysis showed that the protocol is secure under their threats model if the FIDO client does not miss the optional step of verifying the *appID*. Nevertheless, the defined model starts after the establishment of the TLS channel between the FIDO client and the relying party. In order to address this limitation, Jacomme and Kremer [15] performed another formal analysis while taking into account computers malware and communication through TLS. The ProVerif tool was also used to accomplish the automated protocol analysis.

The formal analysis of [15] includes the *remember me* functionality. The model assumes that this functionality is only implemented by some secret value that is stored locally in the FIDO client, and that the relying party verifies alongside the fingerprint of users devices (e.g. IP addresses, browsers version, etc.). We can see that their model suffers from a main shortcoming; it does not faithfully consider various aspects of *remember me*. Indeed, it does not cover any technical detail related to HTTP cookies. In addition, it does not regard the fact that the relying parties actually store the *remember me* cookies. Moreover, our experiments show that the fingerprint of users device has little or no impact: *remember me* cookies remain valid even if used from different browsers running on different computers that are connected from different countries. Therefore, the results of [15] have a limited scope, and many of our identified attacks are not relevant to their model.

3 FIDO U2F Protocol

In this section, we provide an overview of the Universal 2nd Factor (U2F) protocol functionality. This overview covers its two main operations, including the process of registering and authenticating a user to a service provider.

3.1 Industrial Context

The FIDO (Fast IDentity Online) Alliance [2] is an industrial working group that proposes technical standards for online authentication systems reducing or discarding the reliance on passwords. This alliance was formed by PayPal, Lenovo and NokNok Labs in 2013 to define an interoperable set of authentication mechanisms that reinforce a fragmented ecosystem. Two protocols are being developed, namely the Universal Authentication Framework (UAF) [19] and the Universal Second Factor authentication (U2F) [25]. Both protocols might work either together, or independently. Here, we focus solely on the U2F protocols.

In contrast to UAF, the U2F protocol does not supplant the use of passwords. Indeed, a user logs on using a username and a password, then the protocol prompts the user to present a second factor device for authentication. U2F often requires carrying a distinct hardware token. A popular U2F-compliant token is Yubico Security Key called YubiKey [31]. This device comes mostly as a hardware USB token and can support NFC communications. To use this token, users need only to insert it in the USB port of the computer and to press a button during authentication. An important feature is that it is independent of the operating system.

Since recently, the FIDO U2F gains increasing support from online services and web browsers. Today, 67 service providers achieve U2F compliance [30], including Google Account (such as Gmail and YouTube), Dropbox and Facebook. In addition, major browsers add native support for the U2F protocol: Google Chrome (since version 38), Opera (since version 40) and Firefox (enabled by default since version 67). We include Figure 1 that shows the current U2F adoption status in the most popular browsers. Thus, U2F knows increasing adoption from both websites and hardware token manufacturers.

3.2 U2F Terminology

FIDO details the U2F specifications by employing a particular set of terminology that we describe. Henceforth, we will restrict ourselves to use the FIDO jargon. The U2F protocol defines three main actors: FIDO Authenticator, FIDO Client and Relying Party.

The **FIDO Authenticator** is responsible for generating asymmetric key pair and signing authentication challenges. For better security, it shall be a special-purpose hardware device that no entity has a direct access to its

2. `https://caniuse.com/#feat=u2f`

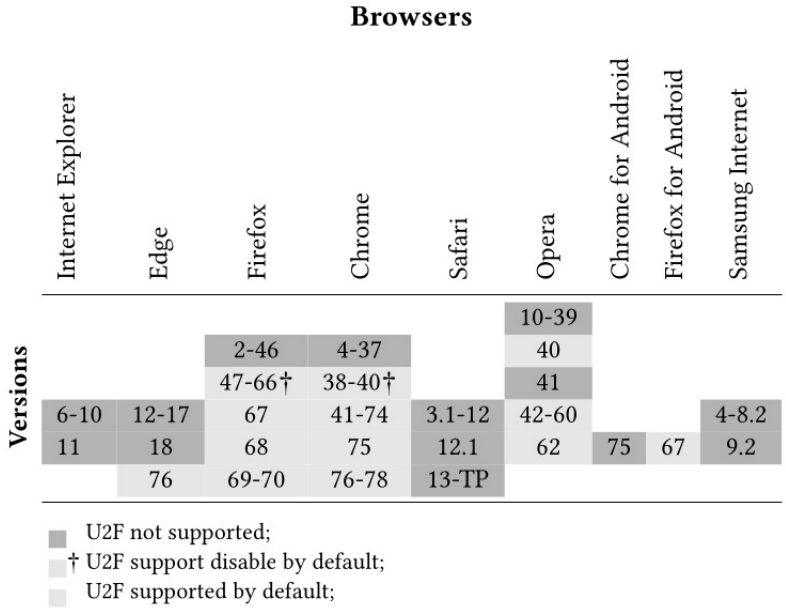

Fig. 1. Browsers' U2F compliance [2]

inner memory. The Authenticator might connect to the users' computer via different interfaces. For example, YubiKeys implement USB HID (Human Interface Device) connection.

The **FIDO Client** is typically a web browser that relays the messages between the FIDO Authenticator and the Relying Party. Moreover, it processes some FIDO messages by performing further verification or collecting more information.

The **Relying Party (RP)** includes two entities: (1) the web server to which the user authenticates, and (2) the server that can verify the authenticity of the used FIDO Authenticator. The RP communicates with the FIDO Client through some JavaScript API [3].

3.3 Protocol Outline

U2F is a challenge-response protocol. Its core idea is standard public key cryptography in which the FIDO authenticator generates a new key pair and shares the associated public key to the registering relying party. For an ongoing authentication, the RP will send a request to the user's authenticator to be signed. Several cryptographic algorithms can be used. For instance, the Yubikey NEO uses RSA-1024 and RSA-2048. The U2F

protocol supports two operations: *registration* and *authentication*. The two operations are illustrated in Figures 2 and 3. Below, we provide more details.

Registration The goal is to register a FIDO authenticator by binding it with a user account. As shown in Figure 2, it begins by the relying party issuing a random challenge. Upon reception, the FIDO client creates a *client data structure* that includes the type of the request (registration or authentication) and the received challenge. The client sends the hash of this structure alongside some RP related values, called the origin, to the authenticator. After the device is touched by the user, the authenticator creates a new credential in the form of a public key K_{pub}, a private key K_{priv} and a key handle. The key handle is used to refer to the K_{priv}. The handle and the K_{pub} are returned back to the client with the associated signature to ensure that the client data was not tampered with. The final step of registration concludes with the relying party storing the handle and the K_{pub} that it received from the client.

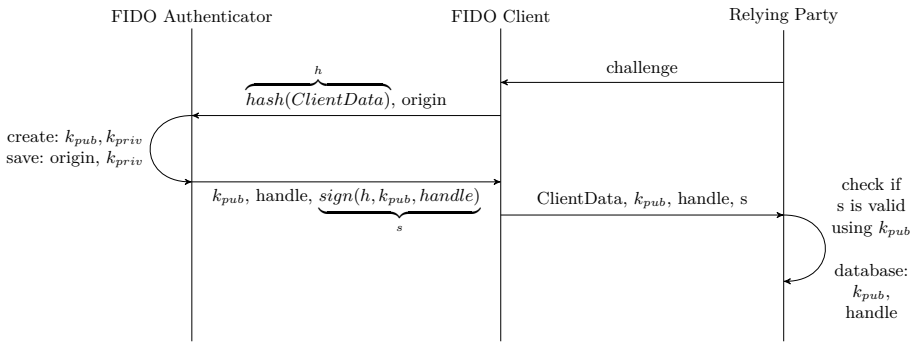

Fig. 2. FIDO U2F Registration

Authentication The aim of this process, illustrated in Figure 3, is to prove the possession of a registered key pair. After verifying the user's credentials, the relying party retrieves the associated K_{pub} and key handle from the registration process. The RP sends the key handle with a new challenge to the FIDO client. In turn, the client builds a new *client data structure* (including the challenge) to the authenticator that signs it using the K_{priv} retrieved from the key handle. The FIDO client passes this

signature back to the RP that verifies the signature and authenticates the user.

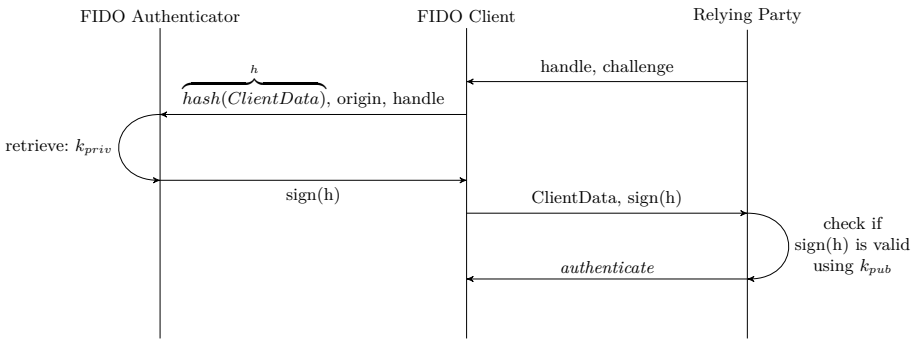

Fig. 3. FIDO U2F Authentication

4 Threat Models and Initial Security Analysis

The primary goal of any authentication system is to prevent attackers from impersonating other users. In 2FA systems, attackers should break the two factors to breach the solution security. In this paper, we focus on the concrete security offered by U2F solutions when coupled with hardware-backed tokens, such as YubiKeys. Thus, we always assume that attackers could guess users passwords. However, we suppose that attackers cannot break into FIDO authenticators, neither compromise cryptographic primitives. In particular, they cannot extract K_{priv} from the related key handle.

Now, we introduce the threat model that we use to capture our security analyses. We first define the different attacker capabilities that the U2F specifications take into account in their risk analysis [18]. Second, we recall the definitions of the relevant security benefits from Bonneau et al. [4] in order to account for maximum coherence in our study. Third, we show a brief security analysis of the U2F authentication protocol, as it is presented in [25]. This will play the role of a reference analysis when we examine the effective security of U2F solutions with *remember me* in Section 6.4.

4.1 Attacker Capabilities

We provide a list of attacker capabilities that are relevant to the four entities participating in the U2F protocol: FIDO authenticator, FIDO client, the communication channel, and the relying party. The capabilities are presented in this respective order.
The **Physical Proximity** ***[PP]*** capability consists in having closer access to users belongings, including the FIDO authenticator and even to their mobile phones. This capability allows attackers to steal objects or observe the authentication process.
The **Man-in-the-Browser** ***[MitB]*** capability allows attackers to compromise the FIDO client. Thus, for instance, attackers might install malicious plugins or read browsers files, such as cookies.
A **Man-In-The-Middle** ***[MitM]*** capability refers to scenarios in which attackers could sneak into the secure communication channels between the FIDO client and the relying party.
The **Intrusion** capability consists in breaking into the relying party, allowing attackers to read, write or execute arbitrary code inside the authentication context of the RP. This capability also concerns curious or malicious relying parties attempting to authenticate to other users' service providers.

4.2 Security Benefits

We consider the following security benefits that are originally defined by Bonneau et al. [4]. We adapt their definitions to U2F:

SB1 *Resilient-to-Physical-Observation:* Means that attackers cannot impersonate users by observing the authentication process one or several times. This includes shoulder surfing and recording users keyboard.

SB2 *Resilient-to-Input-Observation:* Intercepting users input by a malicious software (e.g. keylogger) is not sufficient to achieve illegitimate authentication. Our definition covers intercepting communications with the U2F authenticator.

SB3 *Resilient-to-Communication-Observation:* Eavesdropping all the web communications offers no advantage to attackers.

SB4 *Resilient-to-Leaks:* Breaking into a relying party does not make impersonations on the same relying party any smoother.

SB5 *Resilient-to-Leaks-from-Other-Verifiers:* Intrusion into one relying party is of no help when compromising another relying party. This prevents "hack once, hack everywhere".

SB6 *Resilient-to-Phishing:* Spoofing relying parties shall not leverage any attempt of impersonation against the real RP.

SB7 *Resilient-to-Theft:* Stealing the FIDO authenticator is not enough to be able to use it.

SB8 *Long-Term-Resilient-to-Theft:* This refers to the ability by users of effortlessly revoking registered authenticators. This is critical, for instance, in case of theft or during the process of deprecating vulnerable hardware tokens.

SB9 *Unlinkable:* This privacy-preserving benefit implies that two or more colluding relying parties cannot find out by combining their information whether the same user is authenticating to both using the same authenticator.

Security Benefits

	Resilient-to-Physical-Observation	Resilient-to-Input-Observation	Resilient-to-Communication-Observation	Resilient-to-Leaks	Resilient-to-Leaks-from-Other-Verifiers	Resilient-to-Phishing	Resilient-to-Theft	Long-Term-Resilient-to-Theft	Unlinkable
YubiKey	✓	✓	✓	✓	✓	✓	✗	✓	✓

✗ does not satisfy our definition of "Security Benefit";
✓ implements the "Security Benefit".

Fig. 4. Security Benefits of YubiKey-Backed U2F

4.3 Yubikey's Security Analysis

Here, we will use the attacker capabilities and the security benefits defined above in order to evaluate the U2F protocol based on the hardware FIDO authenticator from Yubico, namely Yubikey. As already mentioned, we solely focus on the second factor; the first factor, usually based on passwords, is not regarded. Figure 4 shows a summary of our security analysis. We notice that our findings are consistent with Bonneau et al.'s [4] (except for the *Resilient-to-Theft*). Simply put, YubiKey-backed U2F solutions, as defined in the standard and without considering actual implementations, fare well in security when compared to other authentication solutions.

The U2F protocol instructs users consent in both registration and authentication. This is achieved by requesting users to press a button on the YubiKey. In fact, the only physical and observable interaction with the YubiKey is this button pressing. Thus, attackers leveraging the *Physical Proximity* capability can only observe this action that reveals nothing about the used YubiKey. Consequently, this allows U2F solutions to be *Resilient-to-Physical-Observation*.

Attackers can, by the *Physical Proximity* capability, steal YubiKeys and effortlessly use them to sign U2F challenges. Therefore, YubiKeys are not *Resilient-to-Theft*. However, users can limit the damage of theft by revoking their registered FIDO authenticator, hence granting the *Long-Term-Resilient-to-Theft* benefit.

Attackers have no access to the internal memory of YubiKeys. Indeed, the U2F interface is the only way to communicate with this hardware token. Thus, by one of the *MitB*, the *MitM* and the *Intrusion* capabilities, attackers are able to observe the exchanged data between the authenticator and the relying party: users' K_{pub}, their key handle, fresh session data (e.g. challenges) and valid signatures. In this setting, the security of the U2F protocol is formally proven in [10] and [15] even if the attackers have access to these data. This corresponds to the benefits: *Resilient-to-Input-Observation*, *Resilient-to-Communication-Observation* and *Resilient-to-Phishing*.

More on the *Intrusion* capability, according to [18], the security of the U2F authentication should also be guaranteed under the hypothesis of corrupted or malicious relying parties. The U2F protocol specifies that the RP only stores the different K_{pub} and key handles of users. As previously assumed, key handles cryptographically protect K_{priv}, and no attacker can break into this protection. Pereira et al. in [21] prove that such information leak from relying parties is not enough to impersonate users on the same or on another RP. This concerns the *Resilient-to-Leaks* and *Resilient-*

to-Leaks-from-Other-Verifiers benefits. Moreover, if we do not consider side-channel attacks, colluding relying parties cannot conclude whether the same YubiKey was behind different public and key handles, thereby leading to the *Unlinkable* benefit.

5 U2F Remember Me

As shown in the previous section, authentication solutions based on FIDO U2F achieve a decent level of security when associated with a hardware-backed authenticator. Nevertheless, such solutions require from users to constantly carry around yet another object. For Yubikeys, the used authenticator must be inserted into a USB-port to complete their authentication. Very often, users tend to keep their authenticators connected to their personal computers in order to avoid continually repeating this extra step.

Regardless of all the efforts made to improve usability, users still express inconvenience, especially when they regularly log in the same services from the same computers. In [7], authors show the general displeasure of users experimenting the 2FA solution of the Carnegie Mellon University. The study shows that users found it annoying despite its ease of use.

Therefore, some relying parties propose a *remember me* option making the 2FA login necessary only once in a while for the device or browser on which it is enabled. Once this feature activated, users will solely be asked for their credentials. This option knows wide approval; in [7] study, almost all participants (95%) stated that using *remember me* made the 2FA authentication more pleasant.

Obviously, enabling the *remember me* functionality vastly improves the user experience. However, this does not come without any security consequence. Indeed, it removes the added security of the U2F protocol if the attacker gains physical access to the victim's computer. Still, the attacker would need the victim's account credentials. If strictly implemented, only authentication attempts from a *remembered* computer shall succeed without the second factor. In other words, the computer (or the browser) plays the role of the authenticator, except for the step implying consent. Thus, the security analysis of the Section 4.3 should be still valid.

Yet, the FIDO U2F standard does not say anything about this feature. As a result, websites did not wait and provided their own solution. Although U2F specifications are public, we could not find any documentation that describes the *remember me* design, or even assesses its security guarantees. We think that it becomes important to have good understanding

of it, since its wide adoption has a profound implication on the security of U2F-compliant authentication services. This is even more important for websites, such as Facebook, in which the *remember me* option is ticked by default.

In this section, we first briefly present the overall architecture of *remember me* solutions. Second, we distill the attributes that should be followed to achieve a good security level.

5.1 Exhaustive Remember Me

There are 67 services supporting U2F [30]. We do not consider OS-based tools, such as computer login with Linux PAM. Instead, we focus on web service providers. We have done exhaustive research and we only found five relying parties proposing remember me: Gmail, Dropbox, Facebook, FastMail and `login.gov`. Note that we mention Gmail as an example of the Google Suite (Youtube, Google Apps, etc) as they all provide the same authentication mechanisms.

5.2 Remember Me Overview

We notice that all the examined *remember me* solutions (refer to the next section for more details) rely on browser cookies. The principle is simple: the cookies are stored by the browser that sends them back with the next authentication request to the related relying party. Despite their disparities, all the solutions of *remember me* share almost the same user experience. Below, we introduce the U2F protocol when we take cookies into consideration.

Activate Remember Me while Authentication When not yet enabled, each time users sign in, they can choose to activate *remember me* by ticking the corresponding box. For Facebook and Google accounts, the checkbox is ticked by default. As for deactivation, there is no trivial way to make the service provider forget the remembered device. Simply put, it is straightforward to remember a device, but not trivial to forget a remembered one.

The U2F authentication workflow remains almost unchanged from Figure 3: users send their credentials and receive a new challenge from the relying party. The connected authenticator signs the challenge with the private key extracted from the key handle. At this stage, a checkbox proposes to remember the device. If ticked, the relying party generates a

cookie after a successful authentication (except for Facebook that generates the cookie before any authentication attempt). The RP stores the generated cookie in its database and sends it to the FIDO client. The browser associates this cookie to the RP domain. The cookie has special enabled flags:

— `secure`: this means that the cookie is transmitted only over a secure channel, typically HTTP over Transport Layer Security (TLS), so that it cannot be eavesdropped.
— `httpOnly`: this implies that JavaScript cannot read this cookie, thereby mitigating Cross-Site Scripting (XSS) attacks.

These flags improve upon the security of these U2F cookies: resist XSS exploitation and are sent only over HTTPS connection. Attackers still can bypass such a protection by the Cross-Site Tracing (XST) attack - a combination of XSS and HTTP TRACE method [11]. It is worth noting that this method is mostly deprecated today, as modern browsers prevent TRACE methods from being made.

Use Remember Me While Authentication Once the option is activated, the U2F cookies are sent alongside users credentials to the corresponding relying party. The RP verifies that at least one of these cookies is both valid and bound to the authenticating users. Otherwise, the RP starts the U2F second factor mechanism. In contrast to a complete U2F authentication, no explicit consent is provided, since no action is required from users.

5.3 Remember Me Evaluation Criteria

Note that each relying party has its own solution for *remember me*. For our comparative study, four properties are distinguished:

EC1 *Inner-Structure:* For each service provider, we look at the content of the generated *remember me* cookies. In particular, we note several aspects: size, fixed patterns and their different fields when we succeed in decoding them.

EC2 *Ephemerality: remember me* cookies shall be persistent with short lifetime. The goal is to set a temporal limit to cookie exploits. We consider that any expiring duration greater than one month does not satisfy *Ephemerality*.

EC3 *Effectiveness-of-Revocation:* In real life, every once in a while, users might like to revoke some remembered computers because

of, for instance, planned replacement of obsolete equipment. Another reason might also be the loss or the theft of personal devices, so that attackers would have shorter time to finish their exploit. Furthemore, relying parties might deprecate some versions of authenticators (because of recently discovered vulnerabilities [6]), hence revoking anything in relation with these authenticators. Here, we do not consider the trivial solution consisting of deleting the U2F cookies on the local storage device, since cookies can be easily copied and stored elsewhere. Surprisingly, no relying party proposes an easy-to-use interface to clear the *remember me* option. Thus, we explore two less-trivial solutions: (1) revocation by changing the associated FIDO authenticator, and (2) revocation by changing users credentials (i.e. passwords).

EC4 *SameSite:* Cookies have two security related flags: `secure` and `httpOnly`. In addition to these flags, cookies are now using an attribute named `SameSite` defined by Google and Mozilla in [28]. It is used to prevent Cross-Site Request Forgery (CSRF) and it takes the values `None`, `Lax` or `Strict`. In `Strict` mode, the cookie is not sent on cross-site queries, but only on resources that have the cookie domain as the origin. In `Lax` mode, the browser will send the cookie with a very limited number of cross-site queries, such as GET queries. As its name indicates, the `None` mode allows cookies to deactivate the `SameSite` feature. Major browsers, including Chrome, Firefox, Edge, Safari, Opera and their mobile versions, already support this feature [27].

6 U2F Remember Me in the Wild

In this section, we shed light on the *remember me* solutions in the wild. We start by describing the settings and the framework of our experiments. Then, we examine the different websites proposing this feature: Google, Dropbox, Facebook, FastMail and `login.gov`. We end this section by revisiting the security analysis of U2F when *remember me* is taken into account.

6.1 Experimentation Settings

In our experiments, our U2F environment is as follows:

— **FIDO Authenticators:** the Yubikey NEO with the firmware version 3.4.9 and the Security Key by Yubico with the firmware version 5.1.2.

— **FIDO Clients:** Mozilla Firefox 67.0.4 for x86_64-pc-linux-gnu and Google Chrome 75.0.3770.100 Official Build 64-bit.
— **Relying Parties:** Gmail, Dropbox, Facebook, FastMail and login.gov.

6.2 Testing Methodology

All our observations have been obtained through various experiments. For each relying party, we have created two users with different passwords and different FIDO Authenticators. We used two geographically separate networks with two different computers. Thus, each experiment has been repeated several times with different settings: user, relying party, Authenticator, FIDO Client. We did not observe any different behavior when the settings change for a particular experiment. For our evaluation, we created several cookies for different users on the same device or for different devices of the same user. Then, we test our criteria as follows:

— *Inner-Structure:* First, we generate three cookies for each user by varying these settings: the user's password, the FIDO Authenticator, and the trusted computer. In total, we obtain six cookies for each service provider. Then, we study whether a fixed pattern exists and relates to a specific setting. Finally, we try to decode their value by using the following encoding schemes: Base64, Base32 and Base16. We also attempt to forge a new valid cookie by modifying the values of the existing cookies. Of course, this says nothing about their (un)forgeability.
— *Ephemerality:* We generate several cookies and look at their expiration date. We are aware that a cookie might be only a pointer to a server database entry, and therefore their lifetime might not be equal to their browser expiration date. Thus, once generated, we waited for one month and also for six months without connection and then tested the cookies again. Of course, our experiments do not prove that a 10-year expiration duration actually means such a lifetime. However, we do at least show that they have an unnecessarily long lifetime. Moreover, to support our results, we have a 3-year-old remember me cookie for a Gmail account and we still can use it to bypass the U2F authentication.
— *Effectiveness-of-revocation:* The authentication page for any relying party does not offer the possibility to "forget" a remembered computer. Here, we test other revocation means that may make relying parties clear their database and refuse a previously-generated cookie. Our methodology is: we first generate a *remember me* cookie

on a given device for a given user, then we perform one of our revocation methods, and finally we attempt to log in using the same cookie. If the relying party asks to insert the U2F token, we conclude that the revocation method is effective. As mentioned above, we evaluate two revocation means: (1) changing the associated FIDO authenticator to a new one, or (2) modifying the account password. For each test, we properly log out and clear all session cookies (except for the *remember me* one) before evaluating the effectiveness of any revocation method.

— *SameSite:* Here, we only look at the corresponding cookie flag and note its value: none, lax or strict.

6.3 Remember Me Implementations

Throughout this section, for brevity and consistency, each "Evaluation Criteria" defined above will be referred to with its short title. Now, we provide our observations regarding widely-deployed *remember me* implementations. Figure 5 summarizes our findings.

Google Google proposes the *remember me* option across its different services: Gmail, YouTube and G Suites. During authentication, when a second factor is registered, the *remember me* checkbox is selected by default. After a successful authentication, if no further action taken by the user, a new cookie named *SMSV* is created and associated to the domain `accounts.google.com`. The cookie expires after ten years of its creation, thus not satisfying our definition of *Ephemerality*. Experimentally, we test several *remember me* cookies and verify that they were still valid after one month, six months and even three years of their generation. Google sets the `SameSite` attribute to `None`.

Google U2F cookies always begin with the same first six characters that are equal to *"ADHTe-"*. Its length is 119 characters. We did not succeed in decoding this structure using different encoding schemes, including Base64, Base32 and Base16. Cookie values *seem* random, but we have no clue how the cookie values are generated.

Regarding cookies revocation, the effective method is to revoke the associated FIDO authenticator (*Effectiveness-of-Revocation (1)*). Indeed, a user that changes her password still bypasses the U2F authentication in remembered devices. We also verify the possibility of devices revocation using the Google dashboard.[3] We found that even if users force the

3. `https://myaccount.google.com/device-activity`

remembered devices to sign out or indicate that they do not recognize them, the *remember me* cookies do not only remain valid, but also browsers do not clear them out.

Google maintains only one cookie on the same device even if several users select the *remember me* option. We noticed that the cookie length grows with respect to the number of remembered users. This shared cookie is not revoked when a new cookie is created. A peculiar observation is that the cookies keep their size despite users revocation. This makes us assume that actual *remember me* values are stored on the RP side.

Dropbox Dropbox allows users to remember their FIDO authenticators. The cookie is associated to the `.www.dropbox.com` domain. Its name is *trusted_$i*, where $i is the user's account numerical ID. Its expiration date is set after ten years, thereby breaking the *Ephemerality* property. In addition, our 1-month and 6-month old cookies were still valid to bypass U2F. The `SameSite` attribute is set to `None`. The cookie value, encoded in Base64, has the structure:

```
{
    "value": {"h": $0, "tkey": $1},
    "signature": $2
}
```

where $0, $1 and $2 are respectively 46, 15 and 46 characters long and are bound together: switching values from different valid tuples is not enough to forge a cookie even for the same user. Despite its decodable structure, we cannot tell exactly how the cookie values are generated, or what cryptographic algorithms are used.

During our evaluation, we noticed that the values of `tkey` and `signature` systematically change for each newly created cookie. However, `h` remains constant for a specific account and will only differ when the user modifies their password. In addition, we notice that if users enter an old password again, they do not find previously generated `h`. Thus, modifying users' passwords is an effective method of cookies revocation (*Effectiveness-of-Revocation(2)*). However, users who revoke their FIDO authenticator and associate a new one to Dropbox still can log into their account without presenting the recently added authenticator on remembered devices. The consequence is that the RP cannot safely revoke vulnerable authenticators, since new ones (possibly more secure) can be bypassed with cookies generated for the old ones.

The cookie values are not bound to the remembered device: a valid tuple (`h`, `tkey`, `signature`) remains valid on other devices and browsers.

Unlike Google, Dropbox generates a different cookie for each *remember me* even on the same computer. If different users share the same computer, the browser will transmit all the *remember me* cookies of all users each time one user attempts to authenticate.

Facebook Facebook, with its 2.41 billion monthly active users [32], implements *remember me* with two cookies, called *datr* and *sb*, linked to the `.facebook.com` domain. To our surprise, these cookies work independently: users need to transmit only one of them to bypass the U2F authentication. Moreover, erasing only one of them has no apparent impact. Unlike all other relying parties, the *remember me* cookies are set before users authentication; just when users arrive at Facebook homepage. They expire two years after their creation, hence not satisfying the *Ephemerality* property. In addition, experimentally, cookies keep their validity even after six months. The `SameSite` attribute is set to `None` for both cookies. We could not decode the cookie values.

The default behavior is to remember authenticating users unless otherwise expressed. If a user does nothing, and so accepts to remember their device, the *datr* and *sb* cookies, which are already sent to users, will be linked to their account and stored in the authentication service database. Similar to Google, this makes us think that some entries must be stored in the RP side. This process implies that multiple users can have the same cookie when using the same browser. Thus, dangerously, two users sharing the same computer but using two different FIDO authenticators get the same *remember me* cookie. This is because the cookies are produced independently of users or their devices (generated before logging in).

As for revocation, we have not succeeded in revoking valid cookies. Indeed, users who modify their passwords and replace their authenticator by a new one still can bypass U2F when one of the *datr* and *sb* cookies is presented. Thus, we think that *Effectiveness-of-Revocation* is not implemented at all. We argue that this property greatly exacerbates several attacks, since one leaked cookie can never be dissociated from their linked accounts. The scenario becomes worse if multiple users share the same cookie.

FastMail FastMail supports U2F into its authentication system. It provides a *remember me* implementation similar to the one from Dropbox. After a successful authentication, if the users tick the *remember me* checkbox, a U2F cookie is set into the FIDO client with the `.www.fastmail.com` domain. The cookie is called *f_$i*, where $i is a string of eight hexadecimal

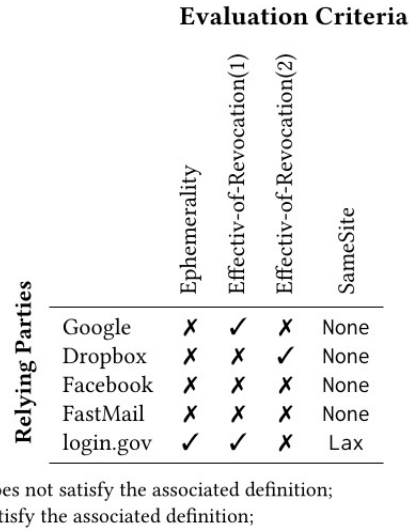

Fig. 5. Experimental Analysis of Remember Me Solutions

numbers representing the user's account ID. The cookie expires ten years after its creation, thus falling *Ephemerality* property. Similar to the other solutions, a 1-month and a 6-month-old cookies are still valid. As for the `SameSite` attribute, it is set to `None`.

When we decode the cookies, we observe the following structure:

```
1;$0;1;$1
```

where $0 is the epoch of creation and $1 is the variable size. This string seems random, but we do not know how it was produced.

Furthermore, similar to Facebook, there is no way to forget remembered computers: the cookies continue bypassing U2F despite modifying users' FIDO authenticator or setting new password. We can say that *Effectiveness-of-Revocation* is not available.

login.gov `login.gov` is an official web service of the United States government providing a single sign-on (SSO) solution for multiple participating government agencies. The registration of a 2AF or U2F authenticator is mandatory to sign up in order to increase the authentication security.

Similar to most relying parties that we study, `login.gov` creates the *remember me* cookie after a successful authentication. The option checkbox is not ticked by default. The cookie is named *remember_device*, and it is linked to the `secure.login.gov` domain. Its expiration date

is set after one month of its creation, which makes it the only site to satisfy our definition of *Ephemerality*. The implementation of `login.gov` is distinguished from other relying parties by two characteristics: (1) a cookie cannot be used after one or six months (even if we manually change the browser expiration date), and (2) the `SameSite` attribute takes the `Lax` value.

After decoding, we observe that cookie values are of size 366 characters. There is no discernible structure that can attest their forgeability. In addition, all valid cookies can be revoked by removing the registered FIDO authenticator and adding a new one (*Effectiveness-of-Revocation(1)*). Changing users password does not clear the entries of the associated *remember me* devices.

Discussion. Our findings highlight four points. First, the structure of *remember me* cookies are often opaque, which hinders our understanding about their values generation. Second, the implemented solutions set a lifetime unnecessarily long for these cookies. It is true that *remember me* makes U2F more pleasant, however, there is no study showing that enforcing the second factor on trusted devices every now and then would decrease the acceptability of such solutions. Third, it is peculiar that revocation is not easier. Worse still, two service providers, namely Facebook and FastMail, do not offer such a possibility to invalidate previously generated cookies. This can be partially explained by the fact that cookies are not bound to both the authenticator and the user account (including password). We believe that it is important to be able to forget devices that were remembered in the past. Fourth, the `SameSite` is still widely overlooked even for major service providers.

6.4 Security Analysis of U2F with *Remember Me*

Here, we provide more insights by revisiting the security analysis using the security benefits of Section 4. We will show the actual security resulted from a bad *remember me* implementation when combined with U2F authentication protocols.

As usual, we consider that attackers can easily defeat the security offered by passwords. We highlight our findings by evaluating the U2F solutions of the examined relying parties through the security benefits that we defined in Section 4.2 and that are widely used across the literature [4]. Our results are summarized in the Figure 6.

First of all, the resulted security analysis cannot be better than the one given in Section 4.3. Therefore, current *remember me* solutions are not

Resilient-to-Theft. However, the use of *remember me* worsens this scenario: the attackers impersonate the victim and *remember* their computers. Then, the attackers can discreetly return the FIDO authenticator. The generated *remember me* cookie will allow the attackers to bypass the U2F verification. Thus, the victim might never be aware of the theft. Second, Facebook and FastMail do not allow cookies revocations, and therefore the compromise will remain valid even if the victim replaces their authenticator. Furthermore, the lack of revocations in some relying parties makes the underlying solutions not *Long-Term-Resilient-to-Theft*.

Given their nature, cookies are vulnerable to any form of observation: physical, internal and communication. No ephemerality exacerbates the situation: stolen U2F cookies can be used as a long-term master key that bypasses U2F authentication. This is due to the fact that a cookie is independent from its *remember me* environment, since it is only bound to a user account and a given relying party within a static setting. Furthermore, the current implementations of *remember me* rely on web cookies that are locally stored with long validity duration. Malicious software inside the browser can achieve their goals by several means. First, the easiest way is to get the cookies and transmit their values to the attackers. Second, because no integrity protection is guaranteed, they can modify the cookies domain or set off their attributes: `secure` and `http-only`. Thus, attackers can easily steal the cookies through XSS vulnerabilities or phishing websites. To recapitulate, the physical, *Man-in-the-Browser* and *Man-in-the-Middle* capabilities allow attackers to compromise the *Resilient-to-Physical-Observation*, *Resilient-to-Input-Observation* and *Resilient-to-Communication-Observation* security benefits. We note that each compromise leads to a large number of impersonation before the expiration date.

Protection against servers intrusion is ensured by the U2F protocols, since no exploitable data are stored on the RP. Nevertheless, our experiments show that relying parties tend to store the *remember me* cookies in order to be able to revoke them in case, for instance, the U2F authenticator is replaced by another one. To the best of our knowledge, there is no document specifying how these cookies shall be securely stored. Assuming no secure storage, any intrusion into the RP database allows retrieving all users cookies. Here, we can claim that Facebook implementation is the most vulnerable one, since the cookies are created even before any successful authentication. Simply put, bad *remember me* implementations are evaluated not to be *Resilient-to-Leaks*. However, we observe that U2F cookies are implemented, so that they are strongly bound to the generating

relying parties. Therefore, we suggest that the examined solutions are *Resilient-to-Leaks-from-Other-Verifiers*.

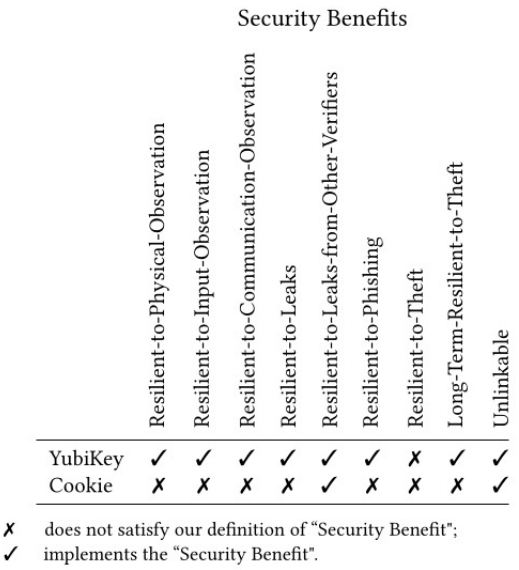

Fig. 6. Comparative Analysis of Authentication Mechanisms Regarding Security Benefits

Concerning the *Resilient-to-Phishing* property, we note that the direct damage is limited, since the U2F cookies will be only transmitted to the origin domain. However, a malicious website can still set some traps to the user by including some HTML elements to perform CSRF attacks. We consider a successful impersonation via cross-site authentication queries as a kind of phishing attacks.

As for the *Unlinkable* property, it is hard to assess anything about it from the structures that we obtained while decoding the cookie values. No technical detail plausibly stipulates any deducible relationship with the associated users account.

7 Attacks and Weaknesses

In the previous section, we have shown that the analyzed U2F solutions use very diverse *remember me* implementations that do contain bad design choices, such as long validity duration of more than six months. We now

exploit these design flaws to bypass U2F without compromising the trusted remembered devices.

Two attacks are defined and implemented. For each attack scenario, we provide a detailed description discussing the attacker settings, and how the *remember me* cookies are exploited. Unless mentioned otherwise, our attacks work successfully against all the previously examined relying parties. We begin this section by recalling the required threats model.

7.1 Threats Model

In order to perform our attacks, we consider a weaker version of the capabilities defined in Section 4.1. Indeed, we only consider two capabilities: (1) an attacker who can include a malicious plugin into a browser, and (2) an attacker executing some phishing attacks.

One may argue that our threats model are not plausible. At first glance, one may say that such attacks will trivially defeat any authentication mechanism. Nevertheless, we argue that many U2F design choices are motivated to mitigate these powerful attackers (access to passwords, Man-in-the-Browser, etc.). For instance, the FIDO Security Reference [18] claims to resist "online attacks by adversaries able to actively manipulate network traffic". All parties supporting U2F adoption (e.g. Google, W3C) think that such a threat model is reasonable enough for wide adoption. We have just considered the created threat model by FIDO and shown that several attacks are still possible because of the remember me feature.

Furthermore, we suppose that the remembered devices are trustworthy, and no malicious software run on them.

7.2 Remember Me on Untrusted Device

The *remember me* feature, as its currently implemented, shifts the security of the U2F authentication to a small file saved into the browser file system. Recovering the value of this file allows attackers to bypass the U2F verification step from other devices for a long time. The U2F cookies have well-identified names and format for each relying party. Thus, attackers would have no problem pinpointing them if they get access to the remembered devices. However, we suppose that attackers cannot compromise these devices, since we assume that users only remember trusted devices.

Our attack starts from this observation: *remember me* cookies are not bound to their remembered devices. We validate this observation through various experiments. In particular, for a given user on a given relying

party, we generate the *remember me* cookie on computer "A". Then, the cookie is copied into computer "B" that was never remembered before. Computers "A" and "B" use different browsers and are connected to separate networks. Finally, we test if U2F is bypassed on computer "B". Our experiments confirm that all the analyzed relying parties have their cookies independent from the *remember me* environment. We also perform the experiment in which the same cookie is used to establish two parallel connections. We notice that no RP finds this behavior suspicious.

We implemented the following scenario. The attacker succeeds in installing some malicious browser plugin in an untrusted device. The malicious plugin can copy cookies and modify queries content before transmission. Somehow, the victim would like to access to their account using this untrusted device. Of course, the victim will not tick the *remember me* option while authenticating. Now, the malicious plugin alters the authentication query by selecting the *remember me* checkbox, and thereby, unbeknownst to the victim, creating a U2F cookie. This is true because we can remember devices without explicit consent from users. Finally, the plugin copies the cookie and sends it to the attacker who can exploit it. These hijacked cookies in the wild make hopeless some relying parties that do not offer the possibility to revoke them.

7.3 Cross-Site Request Forgery

As we explained, the *remember me* cookies are valuable; getting their values is enough for successful impersonation in all the relying parties that we inspected. The previous scenario implies that attackers somehow would put their hands on these values and use them from other computers. Nevertheless, there is another attack vector against cookies: attackers can just ask victims to transmit their cookies to accomplish their malicious queries. At first sight, this assumption does not sound plausible, but this is how browsers deal with cookies by default. Indeed, cookies are sent by the browser to the relying party when an HTTP request starts, and they are sent back from the relying party when their content is edited. This default behavior has caused the well-known Cross-Site Request Forgery (CSRF) attack.

Unfortunately, none of the studied websites, except for `login.gov`, leverage the `SameSite` attribute: Google, Dropbox, Facebook and FastMail use the `None` mode, while `login.gov` uses the `Lax` mode. Therefore, attackers can take advantage from this, since any cross-site authentication request from a remembered device would make the browser include the *remember me* cookie, and thereby bypassing the U2F authentication.

We implement CSRF by including some HTML elements that force the browser into sending HTTP(s) queries to a remembered relying party. This does not require any user intervention, e.g. invisible forms submitted via JavaScript. The vulnerability comes from the fact that the browser will gladly include all the associated cookies. Therefore, assuming these requests contain valid credentials, U2F is bypassed unless the `SameSite` flag is properly set.

8 Countermeasures

Yubico describes U2F as "*a protocol that offers protection from phishing and Man-in-the-Middle attacks*". U2F gets many of its security properties from its challenge-response nature, and the fact that the cryptographic keys never leave the trusted hardware token. This comes in contradiction with how *remember me* is implemented through cookies. Indeed, these cookies play the role of a master key that allows users to bypass their second factor. However, we show that these cookies are not just stored on the trusted devices, but also are sent over the Internet and are stored on the RP side. Therefore, phishing and MitM attacks become possible, thereby raising security issues as we demonstrated in the previous sections.

Thus, we believe that U2F and *remember me* cookies do not go well together. As a result, we propose to keep the challenge-response logic even when *remember me* is activated. Indeed, we suggest that the *remember me* process triggers the generation of a soft U2F token. Here, the FIDO Client, namely the browser, would generate a random key and a key pair. Then, the latter is encrypted using the former to compute a key handler that is sent to the RP alongside with the associated public key. The RP registers this soft token and associates it to the authenticating user. This will not raise a compatibility issue, since most relying parties already allow users to register more than one authenticator. During authentication on a trusted device, the RP should detect the presence of a soft token, and automatically switches to it. The U2F authentication protocol runs normally except for the step that requires users consent. Indeed, we think that it would be better for acceptability that the RP sets the attribute `isFreshUserVerificationRequired` from the U2F standard as `false` in order to carry out the complete authentication without any action from the user.

Assuming that remembered devices are trustworthy, we can easily show that (refer to Section 4) such a solution is *Resilient-to-Communication-Observation*, *Resilient-to-Leaks*, *Resilient-to-Leaks-from-Other-Verifiers*

and *Resilient-to-Phishing*. Obviously, the *remember me* threat model is not compatible with the *Resilient-to-Physical-Observation* and *Resilient-to-Input-Observation* benefits, since this would break the assumption about trusted devices.

We did not provide any reference implementation, because we believe that each RP would adapt it to suit its own flow. Given our study in Section 6, we recommend the following properties:

— *Short-Lifetime*: *remember me* tokens shall be persistent with short lifetime. A secure implementation shall automatically clear out users soft tokens once in a while in order to force the use of the authenticator.
— *Ease-of-Revocation*: users should be able to forget remembered devices in more intuitive way. Moreover, any authenticator revocation or password modification shall result in revoking all the remembered devices by the user.
— *Notification*: users should be informed about newly remembered devices. Enforcing this property limits attacks leveraging the lack of integrity protection of *remember me* queries.

There is still one last property that we discuss: *Hard-to-use-Elsewhere*. As its name indicates, the *remember me* feature is intended to recognize some computers as personal or trustworthy. Therefore, the RP should accept the soft tokens if, and only if, they are used from the remembered device in which they were generated. In other words, if an attacker succeeds in retrieving the secret keys of U2F soft tokens, they shall be unable to bypass the U2F second factor authentication by merely using them from another device. This property is hard to accomplish. We leave the question of strong binding with the remembered device for future work.

9 Practical Attack Against Facebook

Similarly to passwords, the relying parties supporting U2F define a couple of policies in case of losing or forgetting the FIDO authenticator. The most used recovery method is to support alternative authentication methods, so that users can regain access to the account to delete (de-associate) the lost or stolen authenticator from their account. We stress that the recovery method should guarantee a decent level of security, since it allows users to deactivate their U2F authentication. Therefore, sending a reset link to a backup email is not satisfactory, especially when the email provider does not support any 2FA mechanism.

Thus, the relying parties enable multiple forms of 2FA, so that users can apply any to access to their account. In practice, we notice that the relying parties always enable at least another 2FA mechanism for recovery purposes; the most popular one is SMS-based OTP. Ironically, the design of FIDO U2F was motivated to replace these 2FA solutions that offer less security. Readers can refer to [16] for in-depth analysis.

In all the relying parties that we examined (namely Google, Dropbox, Facebook, FastMail and login.gov), the settings interface imperatively asks for a phone number before enabling the U2F authentication. This number is used to receive OTP in case the FIDO authenticator is lost or stolen. Users experience is slightly different for Google, since the phone number is required for the account creation, and is automatically associated when U2F is activated.

It is straightforward to see that the resulted security is as weak as the weakest enabled 2FA. The deployed U2F solutions allow attackers to target a weaker 2FA mechanism in order to permanently deactivate the enabled U2F authentication. The attack scenario is as follows: the attackers guess users credentials, compromise SMS-based OTP and disable U2F authentication. The *remember me* feature can make this scenario worse, because users will not detect such a modification in their security settings.

However, this scenario relies on the assumption that attackers must first compromise users credentials. Even though such an assumption is quite common, it does not lead to practical attacks. During our analyses, we identified an equivalent attack scenario against Facebook in which attackers are not required to have victims' credentials; they only need to get into the SMS-based OTP once in order to deactivate U2F for good. The recent attack against the CEO of Twitter [17] in September 2019 shows that our scenario can lead to effective attacks in practice.

9.1 Responsible Disclosure

We have notified Facebook about this security vulnerability through their bug program. The security team acknowledged the attack and closed our Whitehat report of number #112614246876669 on December 2nd 2019.

9.2 Attack Description

We suppose that the victim has already enabled U2F on her Facebook account. We also suppose that the attacker has a Facebook account and

knows the victim's phone number. The goal is to make the victim more vulnerable by completely disabling U2F. The attack works as follows:

— The attacker signs in her own Facebook account.
— She enters the victim's number phone while activating the U2F authentication.
— Facebook sends a code by SMS to the victim's phone.
— The attacker gets this code by the *Physical Proximity* capability or by tricking the victim.
— When the attacker enters the verification code, Facebook validates the attacker request and associates the phone number to the attacker.
— Facebook finds it strange that a phone number is linked to two accounts. Therefore, it decides to silently deactivate all the 2FA mechanisms of the victim's account. Facebook only sends a notification about the association of the phone number, nothing about the deactivation of 2FA.

To sum up, Facebook cannot associate the same phone number to two different U2F accounts. Indeed, when the same number is used twice, the first U2F activation is disabled permanently. We claim that this scenario is a serious attack vector for multiple reasons. First, it can be carried out remotely though phishing websites. Indeed, a malicious website can fake some authentication problem and ask the victim to enter the received code. Second, it can be automated if the attacker can read the victim's SMS; which is easier than compromising the FIDO authenticator. Third, it requires to target the victim only once for a permanent 2FA deactivation. Fourth, no security alert is made by Facebook about the deactivation. In addition, the settings are not trivial, so we expect that victims detect the attack much lately.

10 Conclusion

This work provides the first systematic analysis on the undocumented U2F *remember me* option. We show that its security impact is not well mastered and often underestimated. Our study points out that the solutions found in the wild significantly weaken the initial security proposed by U2F. Our attack against Facebook abusing its 2FA recovery option and new attack vectors led by *remember me* solutions shows that U2F needs to be better understood to stay secure. Nevertheless, we consider these options essential for usability and acceptability of this protocol by the greatest

number. We expect that this paper will highlight current weaknesses and offer the leads for service providers to improve user security.

References

1. Anne Adams and Martina Angela Sasse. Users are not the enemy. *Commun. ACM*, 42(12):40–46, 1999.
2. FIDO Alliance. Simpler, stronger authentication. `https://fidoalliance.org`. Accessed: 2020-03-01.
3. Dirk Balfanz, Arnar Birgisson, and Juan Lang. Fido u2f javascript api v1.0. Technical report, FIDO Alliance, May 2015.
4. Joseph Bonneau, Cormac Herley, Paul C. van Oorschot, and Frank Stajano. The quest to replace passwords: A framework for comparative evaluation of web authentication schemes. In *IEEE Symposium on Security and Privacy*, pages 553–567. IEEE Computer Society, 2012.
5. Elie Bursztein. The bleak picture of two-factor authentication adoption in the wild. `https://elie.net/blog/security/the-bleak-picture-of-two-factor-authentication-adoption-in-the-wild`. Accessed: 2020-03-01.
6. Naked Security by Sophos. Yubico recalls fips yubikey tokens after flaw found. `https://bit.ly/2RjiUlL`. Accessed: 2020-03-01.
7. Jessica Colnago, Summer Devlin, Maggie Oates, Chelse Swoopes, Lujo Bauer, Lorrie Faith Cranor, and Nicolas Christin. "it's not actually that horrible": Exploring adoption of two-factor authentication at a university. In *CHI*, page 456. ACM, 2018.
8. Alexei Czeskis, Michael Dietz, Tadayoshi Kohno, Dan S. Wallach, and Dirk Balfanz. Strengthening user authentication through opportunistic cryptographic identity assertions. In *ACM Conference on Computer and Communications Security*, pages 404–414. ACM, 2012.
9. Sanchari Das, Andrew Dingman, and L Jean Camp. Why johnny doesn't use two factor a two-phase usability study of the fido u2f security key. In *Financial Cryptography*, Lecture Notes in Computer Science. Springer, 2018.
10. Antonio González-Burgueño, Damián Aparicio-Sánchez, Santiago Escobar, Catherine A. Meadows, and José Meseguer. Formal verification of the yubikey and yubihsm apis in maude-npa. In *LPAR*, volume 57 of *EPiC Series in Computing*, pages 400–417. EasyChair, 2018.
11. Jeremiah Grossman. Cross-site tracing (xst). Technical report, WhiteHat Security, Janvier 2003.
12. Network Working Group. Hotp: An hmac-based one-time password algorithm. `https://tools.ietf.org/html/rfc4226`. Accessed: 2020-03-01.
13. HSBC. Secure key. `https://bit.ly/2KdVY5u`. Accessed: 2020-03-01.
14. Internet Engineering Task Force (IETF). Totp: Time-based one-time password algorithm. `https://tools.ietf.org/html/rfc6238`. Accessed: 2020-03-01.
15. Charlie Jacomme and Steve Kremer. An extensive formal analysis of multi-factor authentication protocols. In *CSF*, pages 1–15. IEEE Computer Society, 2018.

16. Juan Lang, Alexei Czeskis, Dirk Balfanz, Marius Schilder, and Sampath Srinivas. Security keys: Practical cryptographic second factors for the modern web. In *Financial Cryptography*, volume 9603 of *Lecture Notes in Computer Science*, pages 422–440. Springer, 2016.
17. Dave Lee. Twitter CEO and co-founder Jack Dorsey has account hacked. `https://www.bbc.com/news/technology-49532244`. Accessed: 2020-03-01.
18. Rolf Lindemann, Davit Baghdasaryan, and Brad Hill. Fido security reference v1.0. Technical report, FIDO Alliance, December 2014.
19. Salah Machani, Rob Philpott, Sampath Srinivas, John Kemp, and Jeff Hodges. Fido uaf architectural overview v1.1. Technical report, FIDO Alliance, February 2017.
20. NIST. Digital identity guidelines. `https://pages.nist.gov/800-63-3/sp800-63b.html`. Accessed: 2020-03-01.
21. Olivier Pereira, Florentin Rochet, and Cyrille Wiedling. Formal analysis of the FIDO 1.x protocol. In *FPS*, volume 10723 of *Lecture Notes in Computer Science*, pages 68–82. Springer, 2017.
22. Google Play. Google authenticator. `https://bit.ly/1kuly5f`. Accessed: 2020-03-01.
23. Joshua Reynolds, Trevor Smith, Ken Reese, Luke Dickinson, Scott Ruoti, and Kent E. Seamons. A tale of two studies: The best and worst of yubikey usability. In *IEEE Symposium on Security and Privacy*, pages 872–888. IEEE Computer Society, 2018.
24. RSA. Rsa securid hardware tokens. `https://www.rsa.com/content/dam/en/data-sheet/rsa-securid-hardware-tokens.pdf`. Accessed: 2020-03-01.
25. Sampath Srinivas, Dirk Balfanz, Eric Tiffany, and Alexei Czeskis. Universal 2nd factor (u2f) overview v1.2. Technical report, FIDO Alliance, April 2017.
26. Blase Ur, Fumiko Noma, Jonathan Bees, Sean M. Segreti, Richard Shay, Lujo Bauer, Nicolas Christin, and Lorrie Faith Cranor. "i added '!' at the end to make it secure": Observing password creation in the lab. In *SOUPS*, pages 123–140. USENIX Association, 2015.
27. Can I Use. Samesite cookie attribute. `https://caniuse.com/#feat=same-site-cookie-attribute`. Accessed: 2020-03-01.
28. Mike West and Mark Goodwin. Same-site cookies. `https://tools.ietf.org/html/draft-ietf-httpbis-cookie-same-site-00`. Accessed: 2020-03-01.
29. Jeff Jianxin Yan, Alan F. Blackwell, Ross J. Anderson, and Alasdair Grant. Password memorability and security: Empirical results. *IEEE Security & Privacy*, 2(5):25–31, 2004.
30. Yubico. Works with yubikey catalog. `https://www.yubico.com/works-with-yubikey/catalog`. Accessed: 2020-03-01.
31. Yubico. Yubikey. `www.yubico.com/products/yubikey-hardware`. Accessed: 2020-03-01.
32. ZEPHORIA. The top 20 valuable facebook statistics – updated july 2019. `zephoria.com/top-15-valuable-facebook-statistics`. Accessed: 2020-03-01.