An Interoperable Architecture for Usable Password-Less Authentication

Matthew Casey[1], Mark Manulis[2], Christopher J. P. Newton[2], Robin Savage[3], and Helen Treharne[2(✉)]

[1] Pervasive Intelligence Ltd., Fleet, UK
m.casey@pervasive-intelligence.co.uk
[2] Surrey Centre for Cyber Security, University of Surrey, Surrey, UK
{m.manulis,c.newton,h.treharne}@surrey.ac.uk
[3] SSP Ltd., Halifax, UK
robin.savage@ssp-worldwide.com

Abstract. Passwords are the de facto standard for authentication despite their significant weaknesses. While businesses are currently focused on implementing multi-factor authentication to provide greater security, user adoption is still low. An alternative, WebAuthn, uses cryptographic key pairs to provide password-less authentication. WebAuthn has been standardised and is resilient to phishing attacks. However, its adoption is also very low; the barriers to adoption include usability and resilience of keys. We propose a novel architecture for password-less authentication designed to improve usability and deployability. Our architecture is based on the WebAuthn standards and supports registration and login to web-services. We support a WebAuthn authenticator that generates and uses the key pairs on the client device by providing resilience for these key pairs by using a backup key store in the cloud. We also propose a WebAuthn authenticator using a key store in the cloud so that password-less authentication can be used interoperably between devices. We also assess the properties of these architectures against identified threats and how they can form the basis for improving usability and lowering the technical barriers to adoption of password-less authentication.

Keywords: Authentication · Password-less · Crypto-hardware · Key management · Security · WebAuthn

1 Introduction

Passwords are the de facto standard for authentication from e-commerce to online banking, yet they are a weak form of authentication [29]. To strengthen them, current advice [15,27] focuses on making passwords more complex (but not too complex), avoiding password re-use (cf. [17]) and expiry, reducing the number of passwords we use and using password managers, for example LastPass [22]. These measures are designed to balance security with usability, where usability

A. Saracino and P. Mori (Eds.): ETAA 2020, LNCS 12515, pp. 16–32, 2020.
https://doi.org/10.1007/978-3-030-64455-0_2

is crucial in ensuring that we achieve acceptable levels of security. For the general population, however, only a minority of people appear to know how to protect themselves online [2], and hackers take advantage of this with 65% of malicious groups using phishing as their prime attack vector [32]. In simple terms therefore, passwords are easy to use but offer far lower levels of security than we need, especially for accounts that require stronger security, such as a user's prime email account or for online banking. The weaknesses of passwords make it desirable to replace them with stronger, password-less techniques which cannot so readily be subject to phishing attacks. In this paper we explore why password-less technologies are not being widely adopted and propose an architecture which can be used to unify existing password-less technologies and standards to make them interoperable in order to overcome their limitations and to help drive adoption.

Contributions: in this paper we review the barriers to adoption of password-less authentication technologies and propose how these barriers might be overcome. Our approach is to take advantage of the usability of password-less technologies, but to overcome issues of deployment to make them available for the general population through a unified architecture that provides an interoperable approach to password-less authentication adhering to the published standards [5]. We outline how the architecture can be designed, such that with strong security, trust and usability, we can help drive adoption.

In Sect. 3 we describe password-less authentication solutions and the user and business barriers to adoption. In Sect. 4 we propose how the technical barriers can be overcome through architectures that can be used to unify current and future standards-based password-less solutions. These architectures are described in Sects. 4.1 and 4.2 and we consider the trust assumptions and threats related to these architectures in Sect. 4.3. In Sect. 5 we consider how the architectures might be used by users and how this will affect how the systems are designed. Finally in Sect. 6 we conclude with the required next steps.

2 Background

To strengthen password authentication, *multi-factor authentication* (MFA) or (its more common subset) *two-factor authentication* (2FA) has been used, alongside password blacklists and throttling with lockouts [15,27]. Yet while these methods can improve security and, in particular, studies have found 2FA to offer acceptable levels of usability (cf. [31]), even here these extra measures can still be attacked because they rely on something that we know (or write down).

For example, some systems still use the far weaker *two-step verification* which requires the user to provide two pieces of information that they know, such as a password and memorable data, both of which can be compromised in the same way through phishing (cf. [28]). To correctly implement MFA, systems should require that a user provide information from two or more independent factors: 1) something they know, 2) something they have or 3) something they are [15]. However, even here, 2FA in particular can be compromised, for example weaknesses arise when users combine both independent factors into a single

device (logging in from a phone which also receives the text message), or when the second factor is intercepted (text messages can be re-directed by compromising the telecommunications provider [21]).

While MFA provides stronger security compared to passwords alone, a bigger problem however is the lack of uptake of this and similar technologies. For example, in 2018 Google reported that less than 10% of users were using 2FA for their Google accounts [25].

For businesses, one of the drivers causing adoption of stronger authentication is the increasing cost of data breaches, with a global average cost per breach of $3.92m [18], a rise in cost by 12% over 6 years, and with one survey reporting that 94% of businesses cite that data breaches in the previous 12 months have influenced their security policies [33]. With malicious attacks causing 51% of data breaches in 2019 [18], businesses need stronger protection. In general, 58% of organisations believe that 2FA is the most likely access control tool which will be used to protect their systems [33], while 49% believe it is single sign-on and 47% biometric authentication. This shows that stronger access controls are being adopted, with one survey in 2019 [34] reporting 60% of organisations using 2FA or password-less technologies, and a further 29% looking at adoption or expansion of these technologies. However, 26% also cite complex implementation challenges, 26% customer friction and 10% expense as barriers to adoption.

3 Adoption of Password-Less Authentication

It is clear that both users and businesses struggle with the adoption of stronger authentication. If additional steps are implemented, such as basic forms of 2FA, user adoption is low because of perceived usability issues or a lack of understanding. For businesses, while the roll-out of stronger authentication is seen as beneficial, there is concern about implementation complexity. Here then, there is a clear need for stronger authentication, but the barriers to adoption are currently high for users, even if the majority of businesses are moving towards adoption. Microsoft have been promoting password-less authentication both for business (Windows Hello for Business [23]) and other users (Microsoft Authenticator App [24]) and these can work well in a Microsoft environment. Instead of promoting 2FA based on text messages and one-time passwords, or proprietary solutions, in this paper we discuss how a greater emphasis should be placed on password-less solutions using public key cryptography, which offer far better levels of security (thwarting phishing attacks [26]), usability and reduced management costs [37].

Password-less authentication [5,12] allows WebAuthn users to login to web applications using a cryptographic key pair. Once registered with their public key, to log in, the web application issues a challenge which must be signed using the user's private key that is then verified by the web application using the corresponding public key. This challenge-response protocol [5] is resistant to phishing because no credentials are ever exchanged, and instead relies upon the private key being kept secret (and here it is typically unknown and inaccessible to the user). Even if the encrypted challenge-response communication were

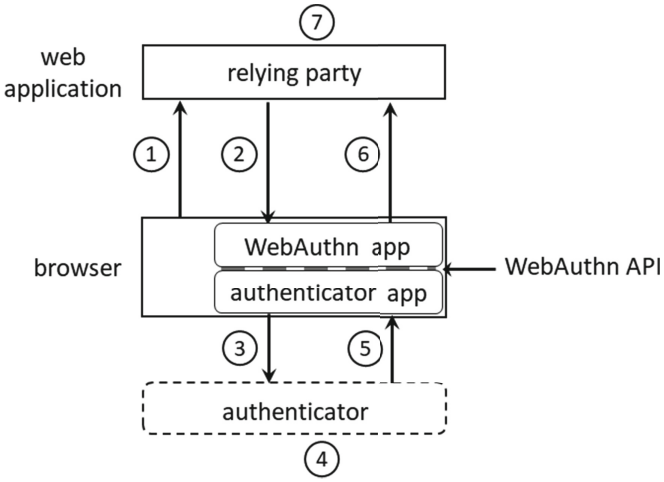
intercepted, it cannot be used in a replay attack because a different challenge would be issued. Also, if a hacker were able to, say, clone the device which is holding the private key, an incremental usage counter can be used to reduce the likelihood that the clone could be used successfully to login. As a potentially usable and secure technology which offers far greater protection than passwords, why has adoption been slow and why has it not supplanted techniques which use passwords? Password-less authentication should be better for both users and businesses. There is clearly an uptake by industry, [10], but there are still barriers to overcome, including preconceptions, knowledge of techniques, expense and deployment (cf. [6]).

Password-less authentication was recently standardised in the W3C WebAuthn recommendation [5] and it is this proposal that we focus on here. WebAuthn is supported in all major web browsers and this gives businesses confidence to develop solutions which will work with any of them. Figure 1 represents the data flows which support registration and login to a web application using WebAuthn. In the W3C documentation login is referred to as the authentication ceremony. In this paper since we use the word authentication in several contexts, we use the term login instead of the WebAuthn authentication term. In the figure we show an authenticator app separate from the authenticator itself, in some cases this may be a single entity. The WebAuthn protocol requires a user to authenticate with the authenticator, this might be using biometrics, a PIN, or a passphrase. The web application, known as a relying party, can decide what security level it will accept for this authentication. We now outline the protocols used for registration and login.

When a user wishes to register an account with a relying party they connect to it (1). The relying party sends the user a challenge (2) which is passed to the authenticator (3). The user needs to authenticate themselves (4) and the authenticator generates a new signing key pair against an identifier for the relying party (5). The identifier, public key and signed challenge are then sent back to the relying party for verification (6) and storage against the newly created account (7).

At a later time when the user wishes to login (1), the relying party sends a challenge to the user (2) and it is passed to the authenticator (3). The user needs to authenticate themselves (4) and then the authenticator signs the challenge using the same private key (5) and sends it back (6). Login is successful if the challenge signature is validated against the public key for the user (7).

WebAuthn defines the protocols and data structures necessary to support registration and login to web applications. In particular it defines the interface for a WebAuthn authenticator which is used to generate the necessary key pairs (on registration) and sign the challenges when a user wishes to login to a relying party. This standard grew out of the work of the FIDO Alliance on the Fast Identity Online protocol and Client to Authenticator Protocol (current versions are FIDO2 [12] and CTAP2 [7]) and many systems base their current implementations on these standards.



	Registration	Login
(1)	user registers	user requests access
(2)	server sends challenge data	server sends challenge data
(3)	challenge passed to authenticator	challenge passed to authenticator
(4)	user verification	user verification
(5)	authenticator generates key pair	authenticator signs the challenge
(6)	public key and attestation sent to server	response sent to server
(7)	server validates response	server validates response

Fig. 1. Data flows in WebAuthn

When built into a device, an authenticator (a *platform authenticator*) is typically just used to protect private keys and other secrets used on that device only. This protection is achieved by, for example, using biometric access to apps, like that provided by Apple's Face ID. Other authenticators are designed to allow the associated keys to be used on any compatible computer via, for example, USB or NFC, and these are known as *roaming authenticators*. These devices are used to hold (or re-generate) key pairs for signing and offer portability between devices, but have limited capacity.

Both platform and roaming authenticators have the same problem: if a device with a platform authenticator or a roaming authenticator is lost or damaged, the data they hold is lost and hence so is a user's access to their registered relying parties. At the moment users of roaming authenticators would need to purchase them in pairs and create backup access by registering both of them with a relying party. This would mitigate the problem of loss or damage, but does not get over their limited capacity. Platform authenticators do not have the same capacity problem, but even when private keys are backed up to the cloud where they are typically encrypted so that they can only be decrypted on

the corresponding phone [3]. So, while either type of authenticator offers strong security and improved usability over passwords their different capabilities are confusing, and this prevents adoption.

This highlights that usability is only one aspect which affects adoption of password-less authentication. Bonneau et al. [6] developed a wider, subjective framework for the comparison of password and password-less authentication methods and their properties. Crucially, this included deployability and security, as well as usability. They concluded broadly that usability and security can be improved through measures such as single sign-on (reducing the need for multiple passwords), but that of the technologies surveyed, most were an improvement in security relative to passwords. However, they also highlighted that every alternative was harder to deploy in some way. Specifically, they were less accessible, more expensive, less compatible with browsers or servers, less mature or proprietary. This is backed up by a study on secure communications tools, which found that usability is not the prime barrier to adoption, but that interoperability, low quality, lack of trust and misunderstanding were also factors [1]. Password-less solutions offer far greater security only if such technologies are actually used, therefore how they are implemented and deployed is just as crucial.

3.1 Adoption Challenges

From this we can summarise the barriers to adoption of password-less authentication faced by users and businesses as follows (building on [6]):

User Adoption Barriers:

Knowledge: With perhaps only 15% of people having sufficient knowledge of how to protect themselves online [2], and the majority of people using passwords, shifting to stronger authentication will take persuasion. Although when mandated by a service provider, people do learn how to adopt new authentication technologies.

Capabilities: There are over 70 FIDO2-certified authenticators [13] available on the market. They each offer different capabilities, such as the ability to roam between devices, and the number of keys that they can hold. They also differ in levels of protection, from no hardware-based protection (Level 1) over to uncertified (Level 2) and certified (Level 3) use of trusted tamper-resistant hardware [4,20,35]. A transparent comparison of capabilities would help, but a unified set of capabilities which meet minimum usability and security requirements would further promote adoption.

Expense: Hardware authenticators have an associated cost (for example, the latest generation of Yubikeys start at $45 per authenticator and Google Titan from $25) While crypto-hardware currently only tends to be built into more expensive devices (Google Pixel 3 from $399 and iPhone from $449). Adoption can therefore be expensive.

Privacy and trust: How do users know that their information is secure and that their privacy is not being compromised? Transparent design and associated information is needed to demonstrate trust in the underlying security.

Availabillty: While WebAuthn is supported by all major web browsers, service providers must adopt WebAuthn in their web applications for it to be available to users. Without this developer support, user adoption is not possible.

Registration and use: When a user has chosen a solution they must be able to deploy password-less access as easily as using a password. Studies on the deployment of Yubikeys (for 2FA) have shown that simple changes to registration instructions can improve adoption [11], so clear guidance is needed.

Resilience: If a roaming hardware authenticator, host computer or phone is lost, key pairs are lost with the device. This reliance on a single device to hold key pairs provides tangible physical security, but is inconvenient.

Service Provider Adoption Barriers:

Security and trust: Which solutions offer the best security for the needs of the business? For example, 2FA is currently the predominant technology being rolled out, perhaps because of its maturity and availability [33,34]. Businesses must be able to select a trusted solution which complies with the levels of security and certification they need (for example, FIPS 140-2 [30]).

Implementation complexity: Some businesses already consider the adoption of stronger authentication mechanisms potentially too complex [34] and these perceptions and the actual complexity of solutions must be overcome.

Roll-out and support: Roll-out of new technologies takes effort in design, implementation, marketing and support, as well as the direct cost for the service. Which solution presents the best value for the available budget? Does the solution reduce on-going support costs? For example, anecdotal evidence for the higher education sector in the UK suggests that Microsoft's Azure MFA has been adopted because of its lower cost.

Interoperability: When deploying password-less authentication to staff within a business, the authentication mechanism, such as a hardware authenticator, can be mandated to reduce complexity. When deploying to consumers, there is little choice over what types of password-less authenticator are used. Interoperable solutions are therefore required with adherence to standards [5,12].

3.2 Overcoming Adoption Barriers

These barriers to adoption lead us to the following aims in designing solutions to promote the adoption of password-less authentication:

- To unify technologies using standards to take advantage of crypto-hardware in a way which provides a minimum set of capabilities for all, therefore reducing confusion over which solution is right for users and businesses.
- To enable secure interoperability between solutions to provide backup and portability of key pairs to improve usability and mitigate against loss.

– To implement security by design and formally verify protocols to instil trust.
– To widely disseminate guidance on password-less technologies, their benefits and use to reduce preconceptions and increase adoption.

In this paper we focus on the first two aspects in order to propose cloud architectures for password-less authentication and we are mindful of the associated adoption challenges when designing them.

4 General Cloud Architectures for Password-Less Authentication

To achieve adoption by business, any approach that is proposed to support password-less authentication must conform, as far as possible, with existing standards, in particular WebAuthn and accepted hardware solutions [35,38]. We do not want to reinvent solutions, especially since a number are well established, rather we want to define a unified framework to support password-less authentication which is interoperable to present a unified user experience. We propose a cloud service approach that works for three use cases:

Use case 1 provides more resilience and reliability for devices that use a platform authenticator (e.g. a biometric reader, a Trusted Platform Module (TPM) [35] or an Intel Software Guard Extensions (SGX) [20] enclave) or a roaming authenticator (e.g. Yubikey) to provide a WebAuthn authenticator. While a local hardware authenticator may offer high levels of security and privacy, if the authenticator can only be used on a single device (i.e. platform authenticator), this restricts users to only using password-less access on that device. While a roaming authenticator offers portability, albeit with limited capacity, if the device is lost then access to relying parties is lost. Our first use case, outlined in Sect. 4.1, provides the same level of security as existing solutions, where the authenticator is local to the device and moreover a cloud service makes the key pairs available to other devices a user wishes to use and alleviates the limitation of key capacity and loss/failure of devices.

Our approach supports two further uses cases where the main functionality of the authenticator is provided by the cloud service application. Section 4.2 provides more details of the architecture that supports these two use cases.

Use case 2 is where a user device has a platform authenticator or roaming authenticator that can be used to authenticate to the cloud service. The cloud service uses a cloud authenticator to generate and manage key pairs that are used to access relying parties. This may be because the authenticator hardware is not WebAuthn compatible or because of limited capacity in the authenticator hardware. This can be considered as the hybrid use case.

Use case 3 is where a user device does not have a platform authenticator or access to a roaming authenticator. This use case is more akin to a password manager, but is now a secure key pair manager, where similar weaknesses of authenticating to the cloud service application are exhibited, but the vulnerabilities that can be introduced by password managers are avoided [8].

Note also that a user may have more than one use case across their devices - for example a user may posses a roaming authenticator that is compatible with one device (use case one), but a second device with is not compatible with the hardware (use case three). Our architecture allows the strongest possible authentication to be used on each device.

One important point to note is that in all cases the cloud service uses a hardware based security module (SM) that handles the keys and encrypts them for storage. An SM could be realised for example using a TPM or, with suitable enclave software, for example, SGX.

4.1 Cloud Service for Backup and Resilience

We propose that users' keys are stored in the cloud using a cloud service. Figure 2 shows a user device and the cloud service. The user device has a local authenticator, either a platform or a roaming authenticator (in what follows we will use the term local authenticator to cover both of these possibilities). The components of the cloud service are an application, a security module that is responsible for managing the keys and a key store that stores encrypted keys and associated data. Thus, even if an attacker gained access to the encrypted keys they would be unusable. This cloud service can be used to backup the keys from the authenticator but can also be used to support the migration of the keys to new devices. This key migration protocol already exists for TPMs. The benefit of this is that it adds resilience and avoids the necessity of registering a new device with each of the relying parties that the user accesses when a user changes device.

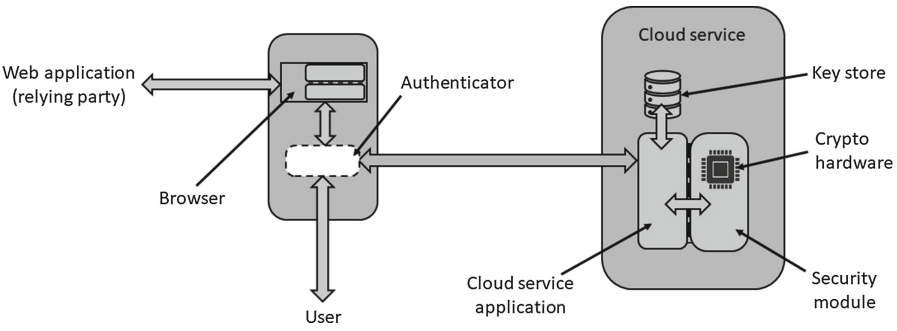


Fig. 2. Using a key store as backup.

When a user registers with a relying party a key pair is generated by the local authenticator and the private key can then be backed-up to the cloud service by

setting up a secure channel between the local authenticator and the cloud SM. To set up the secure channel the user needs to authenticate to the cloud service. Since the device has access to a local authenticator this can also be done using WebAuthn and hence provides strong authentication.

When a user wishes to access a relying party they simply use the local authenticator and do not need to use the cloud service. For this use case we do not consider the situation where the number of encrypted keys in the key store is greater than the storage capacity of the local authenticator. This model could be extended to allow encrypted keys to be swapped in an out as required. However, our second use case provides the ability to handle extra keys when the local authenticator has limited storage.

4.2 Cloud Service as Authenticator

For users whose local authenticator has limited WebAuthn key storage, or those without a local authenticator at all we propose a cloud service application that acts as the WebAuthn authenticator (see Fig. 3). When the cloud service acts as the authenticator the key pairs are generated, used to sign challenges and protected by the crypto-hardware inside the SM.

As discussed earlier, this architecture supports two use cases. Firstly, where a user device has access to local authenticator and authentication to the cloud service can be done using WebAuthn. Secondly, when a user device does not have access to a local authenticator and authentication to the cloud service is be acheived by using a master password together with some form of software authenticator (for example, Google Authenticator). Once authenticated, the WebAuthn protocol works as usual.

When a user first registers with a relying party, the WebAuthn protocol issues a challenge which is relayed to the cloud service through the interface using TLS 1.2 [19] or above. The cloud service then uses its SM to generate a new key pair associated with the relying party, signs the challenge and returns the signature, public key and attestation information. When the user next logs into the relying party, a challenge is issued and again signed by the cloud service.

4.3 Basic Trust Assumptions and Threats

In this section we examine the trust assumptions and threats related to each use case.

Backup and Resilience Service: To understand how this service can resist attack, we have listed our trust assumptions and identified possible threats and their mitigation.

1. We assume that the cloud service application together with the cloud service provider are architected in such a way as to ensure resilience in the case of hardware/software failures via replication/clustering etc.

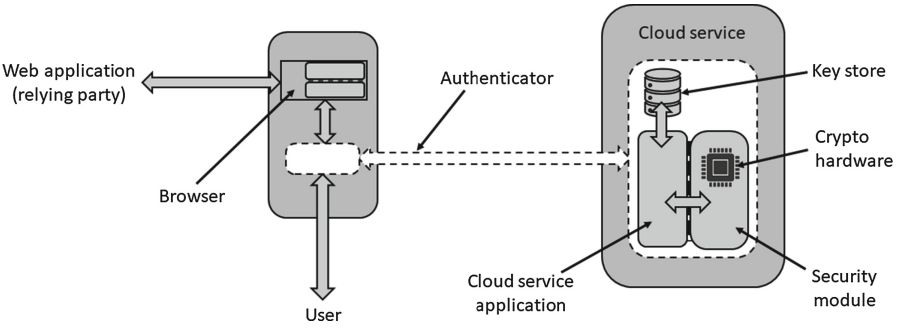


Fig. 3. WebAuthn configuration using a cloud service with SM

2. We do not assume that the communication channels between the different parts of the system are secure by default and so TLS 1.2 or above should be used when the user's device connects to the cloud service. In addition, once a user has authenticated to the cloud service application a secure channel should be established between the local authenticator on the user's device and the cloud service's SM to further secure the keys and data being transferred.
3. The cloud service application is trusted for integrity and availability of associated data and, where necessary, encrypted keys. In addition it is assumed to correctly pass information between the SM, a user's device and a relying party for (i) exchange of WebAuthn challenge and response messages and (ii) migration of keys.
4. We assume that the SM can be trusted and that includes the integrity and confidentiality of the keys generated and processed by the SM. Should the SM be compromised an attacker would be able to access all plaintext data and monitor its operations. User devices must therefore attest to the correct operation of the SM prior to any actions being performed. In addition, each user's data within the key store should be encrypted at rest, preferably using a key derived from the user's password provided this password was set up when registering with the cloud application service (although this would remove any possibility of password reset). The system design should ensure that these credentials are only be available within the SM once a user has authenticated and has established a secure channel.
5. We assume that a user's device is operating correctly. Our confidence in this assumption can be reinforced if the device can perform local attestation to confirm the state of its system before the SM establishes a secure channel for communication. This is in addition to the attestation used to confirm the state of the SM to the user.
6. We assume that the user's device correctly authenticates with the cloud service. The use of the local authenticator for WebAuthn authentication to the cloud service means that the cloud service application can rely of this authentication. Where a local authenticator is not available, authentication must be via a password with a second factor.

Cloud Authentication Service: The same assumptions listed above are made about this service, but when there is no local authenticator some of the mitigations are no longer possible, in particular those for items 5 and 6. Authentication to the cloud service can use a strong password and a second factor, such as the use of a software authenticator or a time-based one-timed password. However, attestation of the user's device will not be possible. There is clearly a trade off between the ability to use any device and the extra security provided by just restricting access to devices with local authenticators.

In addition since the cloud service is used to generate and use key pairs for specific relying parties, user privacy may be compromised if a malicious cloud service records user logins. This possibility can be mitigated by using, for example, the hash of the relying party's identifier rather than the relying party's identifier itself when using the cloud authentication service.

5 Using Our Password-Less Authentication Architectures Across Multiple Devices

Our approach must support users with multiple devices, since many users have more than one device which are likely to have different security capabilities, for example a phone, tablet and laptop. Thus the combination of devices can be categorised as follows:

Combination 1. Multiple devices all of which use local authenticators (either platform or roaming) supporting use case 1,
Combination 2. Multiple devices none of which use local authenticators (either platform or roaming), i.e. supporting use case 3,
Combination 3. Multiple devices some of which can use local authenticators (either platform or roaming), others with no available local authenticator, which is the most heterogrenous combination of capabilities supporting use cases 1, 2 and 3.

The different combination types have their own security implications that will influence the overall implementation of our approach but clearly the architectures presented in Sects. 4.1 and 4.2 may be used simultaneously by multiple devices.

The aspects we need to consider are when (i) users register new devices with the cloud service, (ii) register with a relying party, (iii) access a relying party and (iv) a user a loses a device.

5.1 Registration and Authentication of a User Device with the Cloud Service

When a user registers their device with the cloud service using their local authenticator a WebAuthn key pair is generated which can be used to give them access. The user would also need to provide an alternative means of authentication to use in case their device becomes lost or damaged. This would ideally be another device with a local authenticator, but the user may need to fall back on using a

password and a second factor not associated with their WebAuthn device. Once a device with a local authenticator is registered it can be used to vouch for any further registrations of devices with local authenticators to ensure that they are associated with the same user.

Users without any devices with a local authenticator would need to register with the cloud service using a password and second factor as there is no device to register in this case; this is analogous to registering with a password manager. The mechanism used to register with the cloud service determines what a user would also need to do when subsequently authenticating to the cloud service. The use of multiple devices does not introduce security complexity for this aspect.

Recall, that once a device with a local authenticator is registered, it can then be automatically authenticated to the cloud service as necessary. If a device does not have a local authenticator, it would either need to do this each time or this can be made less of a problem by allowing a user to stay authenticated from a particular device for some time (using a token for example).

5.2 Registering with a Relying Party

We have already described how a single device with or without a local authenticator registers with a relying party. We now consider each combination and highlight what needs to be considered.

For combination 1, a device uses its local authenticator to register a user with a relying party. Then we anticipate that at pre-defined intervals a user's local keys are backed up to the cloud service. If the user has more than one device with a local authenticator then the keys would be synchronised across all devices so that the same relying parties can be seamlessly visited from any of them using the same key pair. If a local authenticator cannot store all of the keys then swapping of keys in and out of this local authenticator may be possible. If a local authenticator cannot synchronise then the associated user device would be required to utilise the cloud authenticator (combination 3).

For combination 2 the use of multiple devices introduces no further complications, the user logs in to the cloud service and is then able to register with relying parties and the keys are stored in the cloud. Once authenticated to the cloud service, the keys can then be available to the user from any of their devices. There are no capacity issues in this case.

For combination 3 those devices with access to a local authenticator could behave in the same way as combination 1 or they could just use the local authenticator to support authentication to the cloud service. In this case the ability to use the cloud authenticator would remove any capacity issues. Those without access to a local authenticator would need to authenticate to the cloud service using the registered password and second factor and would use the cloud authenticator to register with relying parties. The policies for sharing keys across devices would need to be carefully considered. In the case of devices in combination 3 not having a local authenticator you would expect that all keys be shared, as is the case in a password manager. In the case of a device which uses local authenticator for authentication, it would also be natural to share the keys with devices with no

local authentication but accepting that there is some downgrade in the security. In the case of a device which has a local authenticator and we consider the sharing of its keys with devices that have a local authenticator only used for authentication with the cloud service, and with devices that have no local authenticator, then implementation policies may need to be introduced to restrict this sharing. For example, if the device with a local authenticator which stores its keys locally is used to access a bank account, then it may be appropriate to restrict other devices with weaker security capabilities from accessing it.

5.3 Logging on to a Relying Party

We have already described how a single device with and without a local authenticator accesses a relying party. Since the controlling of access to the keys and policies is determined when registering with a relying party, no further issues are introduced when accessing it in any of the combinations.

5.4 Loss of a Device and Revocation

For a user with a single device with a local authenticator their keys will be regularly backed up so that should their device become unusable/lost/stolen and need to be replaced they can register a new device with a local authenticator using the two factor authentication previously set up. Once the new device is registered the WebAuthn data will be downloaded and can be used. All of the user's previous relying party registrations can still be used. As above, if there are any capacity issues with downloading the keys then, as above, the device would not use the local authenticator to store the keys for the relying party but rather use the cloud key store. If the replacement device did not have a local authenticator, the keys would simply remain in the cloud and be accessed from there. If a device with no local authenticator was to be replaced by another then the replacement would similarly either download the keys or retrieve them from the key store.

An implementation would also need to consider the revocation of keys in the case of a lost or stolen device. If the device had no local authenticator (and hence did not store any WebAuthn private keys) then only its two-factor authentication would need to be deactivated on the cloud service to prevent anyone from access the cloud service on user's behalf. If the device was equipped with a local authenticator then any WebAuthn private keys that were simultaneously stored on the device and the cloud store must be revoked. The user would first use these private keys in the cloud authenticator to login to relevant relying parties and ask them to deactivate corresponding WebAuthn public keys, before deleting these private keys from the cloud store and deactivating cloud service access for the lost device.

6 Conclusion and Future Work

This paper proposes a cloud approach that works for three use cases. The first use case increased the resilience of devices that have a platform or a roaming

authenticator to provide a WebAuthn authenticator and also strong authentication to the cloud service application. The second and third use cases moved the main authenticator functionality from a user's device to the cloud service application and this is where the innovation in our approach lies; the two use cases differ only in the use of a local authenticator if available to authenticate with the cloud service. It provides the opportunity for a variety of devices to utilise our cloud approach with differing levels of security depending on how authentication to the cloud service application is achieved. The issues of authenticating to a cloud service is not unique to our approach, but are challenges that would apply to any cloud based system. Our motivation for this research was to minimise the friction of an authentication approach in order to make secure password-less authentication more feasible for users with less technical experience and who cannot afford to upgrade or supplement their existing hardware. Therefore, we proposed an approach that would be applicable to a range of users whose devices and hardware have different capabilities.

The next step in this research is to focus on the development of the proposed cloud authenticator and its protocols and it will be important to take recent work by Chen et al. [9] and by Fryman et al. [14] into account. This will require definition of WebAuthn specific binding to ensure inter-operability. Once the architecture has been fully developed with its corresponding security protocols then a formal analysis will need to be conducted to verify its security. Such analyses can expose threats which can then be mitigated. Formal analysis has already been conducted for WebAuthn [16] and our experience of formally analysing direct anonymous attestation protocols will also be relevant [36]. Furthermore, a reference implementation of the architecture will also serve to promote discussion with industry. It will also be important to examine the socio-technical aspects of the architecture to re-visit the barriers to user adoption because reflecting these concerns in architecture design is particularly important.

Acknowledgments. This research was funded by EPSRC through the DICE project, EP/N028295/1. The authors would like to thank the reviewers for reviewing this paper and for their helpful comments.

References

1. Abu-Salma, R., Sasse, M.A., Bonneau, J., Danilova, A., Naiakshina, A., Smith, M.: Obstacles to the adoption of secure communication tools. In: IEEE S&P 2017, pp. 137–153, May 2017. https://doi.org/10.1109/SP.2017.65
2. Ames, A., Stannard, J., Stellmacher, D.: UK cyber security survey 2019. https://www.ipsos.com/ipsos-mori/en-uk/uk-cyber-security-survey-2019/ (2019)
3. Apple Inc.: Storing keys in the secure enclave — apple developer documentation. https://developer.apple.com/documentation/security/certificate_key_and_trust_services/keys/storing_keys_in_the_secure_enclave/ (2020)
4. Arm Limited: Trustzone - Arm developer. https://developer.arm.com/ip-products/security-ip/trustzone/ (2019)
5. Balfanz, D., et al.: Web authentication: an API for accessing public key credentials level 1, March 2019, https://www.w3.org/TR/2019/REC-webauthn-1-20190304/

6. Bonneau, J., Herley, C., Oorschot, P.C., Stajano, F.: The quest to replace passwords: a framework for comparative evaluation of web authentication schemes. In: 2012 IEEE S&P, pp. 553–567, May 2012. https://doi.org/10.1109/SP.2012.44
7. Brand, C., et al.: Client to authenticator protocol (CTAP), January 2019. https://fidoalliance.org/specs/fido-v2.0-ps-20190130/fido-client-to-authenticator-protocol-v2.0-ps-20190130.html
8. Carr, M., Shahandashti, S.: Revisiting security vulnerabilities in commercial password managers. In: International Conference on ICT Systems Security and Privacy Protection, IFIP SEC 2020, February 2020. https://sec2020.um.si/
9. Chen, S., Barbosa, M., Boldyreva, A., Warinschi, B.: Provable security analysis of fido2. Cryptology ePrint Archive, Report 2020/756 (2020). https://eprint.iacr.org/2020/756
10. Cimpanu, C.: Microsoft: 150 million people are using passwordless logins each month. https://www.zdnet.com/article/microsoft-150-million-people-are-using-passwordless-logins-each-month/ (2020)
11. Das, S., Russo, G., Dingman, A.C., Dev, J., Kenny, O., Camp, L.J.: A qualitative study on usability and acceptability of Yubico security key. In: Proceedings of the 7th Workshop on Socio-Technical Aspects in Security and Trust, pp. 28–39. STAST 2017, Association for Computing Machinery (2018). https://doi.org/10.1145/3167996.3167997
12. FIDO Alliance: FIDO2: Moving the world beyond passwords using WebAuthn & CTAP. https://fidoalliance.org/fido2/ (2019)
13. FIDO Alliance: Certified products. https://fidoalliance.org/certification/fido-certified-products/ (2020)
14. Frymann, N., Gardham, D., Kiefer, F., Lundberg, E., Manulis, M., Nilsson, D.: Asynchronous Remote Key Generation: An analysis of Yubico's proposal for W3C WebAuthn. In: ACM CCS 2020. ACM (2020), https://dx.doi.org/10.1145/3372297.3417292
15. Grassi, P.A., et al.: NIST special publication 800–63B: Digital identity guidelines: authentication and lifecycle management, June 2017. https://pages.nist.gov/800-63-3/sp800-63b.html
16. Guirat, I.B., Halpin, H.: Formal verification of the W3C web authentication protocol. In: 5th Annual Symposium and Bootcamp on Hot Topics in the Science of Security. HoTSoS 2018, Association for Computing Machinery (2018). https://doi.org/10.1145/3190619.3190640
17. Hussain, T., Atta, K., Bawany, N., Qamar, T.: Passwords and user behavior. J. Comput. **13**, 692–704 (2018). https://doi.org/10.17706/jcp.13.6.692-704
18. IBM Security: Cost of a data breach report 2019. https://databreachcalculator.mybluemix.net/ (2019)
19. IETF: The transport layer security (TLS) protocol. https://tools.ietf.org/html/rfc5246 (2008)
20. Intel: Intel® software guard extensions. https://software.intel.com/en-us/sgx/ (2020)
21. Jacobs, F.: How Russia works on intercepting messaging apps. https://www.bellingcat.com/news/2016/04/30/russia-telegram-hack/ (2016)
22. LastPass: LastPass technical whitepaper. https://support.logmeininc.com/lastpass
23. Microsoft: Passwords-less protection. https://query.prod.cms.rt.microsoft.com/cms/api/am/binary/RE2KEup
24. Microsoft: Enable passwordless sign-in with the microsoft authenticator app. https://docs.microsoft.com/en-us/azure/active-directory/authentication/howto-authentication-passwordless-phone (2019)

25. Milka, G.: Anatomy of account takeover. In: Enigma 2018 (Enigma 2018). USENIX Association, Janunary 2018. https://www.usenix.org/node/208154
26. Mirian, A., DeBlasio, J., Savage, S., Voelker, G.M., Thomas, K.: Hack for hire: exploring the emerging market for account hijacking. In: WWW 2019. p. 1279–1289. ACM (2019). https://doi.org/10.1145/3308558.3313489
27. National Cyber Security Centre: Password administration for system owners. https://www.ncsc.gov.uk/collection/passwords/updating-your-approach/ (2018)
28. National Cyber Security Centre: Setting up two-factor authentication (2FA). https://www.ncsc.gov.uk/guidance/setting-two-factor-authentication-2fa/ (2018)
29. National Cyber Security Centre: Passwords, passwords everywhere. https://www.ncsc.gov.uk/blog-post/passwords-passwords-everywhere/ (2019)
30. National Institute of Standards and Technology: FIPS 140–2: Security requirements for cryptographic modules, June 2001. https://csrc.nist.gov/publications/detail/fips/140/2/final
31. Reese, K., Smith, T., Dutson, J., Armknecht, J., Cameron, J., Seamons, K.: A usability study of five two-factor authentication methods. In: Fifteenth Symposium on Usable Privacy and Security (SOUPS 2019). USENIX Association, Santa Clara, CA, August 2019. https://www.usenix.org/conference/soups2019/presentation/reese
32. Symantec: Internet security threat report (ISTR) 2019. https://www.symantec.com/content/dam/symantec/docs/reports/istr-24-2019-en.pdf (2019)
33. Thales: 2019 Thales access management index. https://safenet.gemalto.com/identity-access-management-index/ (2019)
34. ThumbSignIn, One World Identity, Gluu: Customer authentication practices 2019. https://thumbsignin.com/customer-authentication-report-2019/ (2019)
35. Trusted Computing Group: TPM 2.0 library specification. https://trustedcomputinggroup.org/resource/tpm-library-specification/ (2016)
36. Wesemeyer, S., Newton, C., Treharne, H., Chen, L., Sasse, R., Whitefield, J.: Formal analysis and implementation of a TPM 2.0-based direct anonymous attestation scheme. AsiaCCS 2020 (to appear) (2020). https://ethz.ch/content/dam/ethz/special-interest/infk/inst-infsec/information-security-group-dam/research/publications/pub2020/eccdaaimp-asiaccs20.pdf
37. World Economic Forum: Passwordless authentication: The next breakthrough in secure digital transformation. http://www3.weforum.org/docs/WEF_Passwordless_Authentication.pdf (2020)
38. Yubico: Yubico — YubiKey strong two factor authentication. https://www.yubico.com/ (2020)