Let's Authenticate: Automated Cryptographic Authentication for the Web with Simple Account Recovery

James S. Conners
Brigham Young University

Daniel Zappala
Brigham Young University

Abstract

The FIDO2 Alliance is proposing standards for cryptographic authentication that are intended to replace passwords. We show that their standards could lead to difficult registration, cumbersome account recovery, and potential privacy leaks and tracking. We propose an alternative architecture based on certificates instead of bare keys that provides automatic registration and login and simplified account recovery. We use a framework to compare our approach to related work, illustrating usability, security, and privacy trade-offs.

1 Introduction

Researchers seeking to improve user authentication face a quandary — passwords have well-known flaws, but no alternative has been shown to be better. It's well known that users choose weak passwords [3], that website rules can be counterproductive [4] and have difficulty remembering strong passwords [11]. As a result, they often choose weak passwords or reuse passwords across sites. Password managers greatly simplify password use, but adoption is far from universal [9]. Moreover, even when users do select strong, unique passwords, service providers may instead become a weak link if they fail to follow best practices with respect to password storage. Despite these issues, a thorough review of alternatives by Bonneau et al. found that no replacement schemes comes close to supplying the benefits of passwords, and "none even retains the the full set of benefits that legacy passwords already provide" [1]. They show that when replacements offer significant security benefits in comparison to passwords they are also more costly or more difficult to use. Limitations of proposed alternatives to passwords mentioned by Herley and van Oorschot include requiring users to purchase or carry additional hardware, fragility due to a single point of failure such as for a single sign-on scheme, or having inadequately studied usability [5]. Decades of research suggest there is unlikely to be a solution that kills off passwords, and tradeoffs among solutions will always exist.

Despite this bleak outlook, in recent years there has been a resurgence of interest in augmenting or replacing passwords with cryptographic authentication, meaning authentication backed by public-key cryptography. FIDO2 is being touted as the standard for public-key-based authentication on the web and has begun to be adopted by web browsers. FIDO2 works by having users adopt authenticators, which can be internal to their device or external, such as a hardware token (e. g., Yubikey) or a smartphone app (e. g., Duo). The authenticator generates a private-public keypair for a website at registration and the website then stores the public key as an identifier. To authenticate to the website, the user requests sign-in, and the website requests the user to sign some piece of data with the private key that matches the stored public key. The user signs the data, returns it, and is authenticated.

While this momentum is encouraging, substantial work is still needed to ensure that cryptographic authentication provides a usable alternative to passwords for users. The FIDO2 standards are still very early in the deployment phase and only a little work has been done to date demonstrating their usability for second factor authentication [10], with no work done on their use for first factor authentication, as a password replacement. There are numerous scenarios that must be considered, including registration, login, and account recovery, as well as privacy concerns to consider.

In this paper we explore an alternative design, called Let's Authenticate, that provides automated account registration and login, along with simple account recovery when an authenticator is lost. Let's Authenticate is inspired by Let's Encrypt and seeks to similarly make it easy to issue certificates to users, which they can then use to register and login

USENIX Symposium on Usable Privacy and Security (SOUPS) 2019.
August 11–13, 2019, Santa Clara, CA, USA.

to websites. Certificates are issued based on proving ownership of an account (*e.g.*, with a username and password), which allows for easy re-issuance if an authenticator device that stores certificates is lost. We believe grounding ownership of accounts in ownership of passwords, at least initially, can help users make the transition to stronger cryptography by bootstrapping it off a method they already use. In addition, using certificates instead of bare public keys allows for easy registration, login, and renewal, as well as authorization and deauthorization of devices for account access. We explain how Let's Authenticate works, compare this approach to other work to illustrate usability, security, and privacy trade-offs, and outline a research agenda.

2 Related Work

Cryptographic authentication has been most prominently used in Belgium [2] and Estonia [7], where citizens are issued an electronic identification card, or eID. Citizens can obtain an eID at a municipal center by proving their identity, for example with a passport. These eIDs act as a standard identification card, but have three certificates installed directly into the card, on secure hardware. These three certificates can be used for authentication, non-repudiation, and to identify the card to the government for certificate verification. Many countries have now adopted eIDs, and some are adding systems that allow a smartphone to be used in place of the eID using a PKI-capable SIM card and a user-assigned PIN.

Several academic systems based on cryptographic authentication have been created. Loxin [12] proposed issuing a single certificate to each user, which would be stored in a smartphone app and used to authenticate to web sites. The authors suggest revocation can be handled by generating a backup key pair that can be printed out and used in case the smartphone is lost. The n-Auth system [8] proposed using public keys and an authentication protocol to enable a smartphone to register and login users on websites. Their system does not allow users to have multiple authenticators or to recover accounts after an authenticator is lost.

Mozilla Persona, a now defunct authentication method, was a set of protocols that allowed email providers to issue their users certificates certifying ownership of a given email address. Persona provided a simpler registration and login process by using the user's email address as their identity, meaning websites did not need additional information during signup. It also implemented the BrowserID protocol, allowing communication between the browser and email provider to obtain certificates. To allow for recovery, Persona suggested users could add a secondary email address to their account. Persona protects user privacy by ensuring that email providers are not given information they can use to track user logins to websites. However, colluding websites can still track users because email addresses are used as an identifier—users would need to generate separate email addresses per website to avoid this.

Persona had significant problems with adoption. First, websites were required to implement Persona. Second, web browsers were required to implement the BrowserID protocol. Finally, email providers were required to become identity providers. They attempted to solve these issues by implementing a fallback identity provider, creating a cross-browser library for the BrowserID protocol, and by hosting a verification API so that websites can easily verify user credentials.

FIDO2 is an effort to replace passwords on the web, consisting of two standards. First, WebAuthn [10], standardized by the W3C, specifies how a web application can use JavaScript to help a user register and login using an authenticator. An authenticator handles cryptographic logins and can be either an app on a device (such as a laptop) or an external device such as a hardware token or a smartphone. Second, CTAP2, standardized by the Fido Alliance, specifies a method for web browsers to communicate with external authenticator via USB, BlueTooth, or NFC.

WebAuthn provides a standard user experience for authenticating over the web using public-key cryptography. It is a generic standard that allows for authenticators to provide either second-factor or first-factor authentication. The basic actions include (a) a user registers an authenticator with a relying party, (b) the authenticator generates a public/private key pair unique to the relying party, and (c) the user can then authenticate to the relying party at any subsequent time. Authenticators can sign each new key pair with an attestation certificate, certifying that a specific key pair originated within a given device. The attestation certificate can be chained to a root of trust. Relying parties can specify features that an authenticator must support, such as being an external authenticator or obtaining user verification, such as through a password, PIN, or biometric. These features are also certified through an attestation certificate. The key used for attestation certificates is stored in hardware when the authenticator is manufactured, though authenticators without this capability can use self-signed certificates.

3 WebAuthn Analysis

The WebAuthn standard provides strong security but has not yet been tested with respect to usability as a password replacement, nor been analyzed with respect to privacy. We examine some of the architectural choices made in designing WebAuthn from the perspective of usability and identify several concerns:

- *Difficult and confusing registration:* Registration in WebAuthn is handled by the relying party. While WebAuthn provides a unified authentication flow for users, providing some consistency, a user must still register an authenticator for each individual website visited. Work done

registering an authenticator at one site does not reduce the work needed to register the authenticator with a second site. A website can impose restrictions on which authenticators it supports, so users may not know whether they have a valid authenticator until registration time. Since registration for some sites will be for second-factor authentication and others for first-factor, users may be confused about how to use authenticators and when they need to use them for which sites.

- *Cumbersome account recovery:* Backup of login credentials is not supported. Thus, if a user registers a single authenticator with a web site, then loses that authenticator, they will lose access to their account. To provide account recovery, a user is required to register multiple authenticators so that backup authenticators can be used in case one is lost. Moreover, if a lost authenticator does not require user verification (*e.g.*, just a button press), then someone who finds or steals the authenticator can gain access to accounts and lock out the true owner. In this case, a race condition is created, whereby the first to login and deactivate other devices (the true owner or the thief) will win permanent access to the account. A website could alternatively support traditional account recovery methods for users, but this lessens the security offered by the scheme and requires each website to follow best practices.

- *Potential for authenticator bloat:* Web sites are free to impose restrictions on authenticators, so a user may need different authenticators for different sites. The spec currently provides enough freedom that it could be possible that each website could require unique authenticators, though this is not in keeping with the spirit of the specification. However, even if the number of different authenticators a user is required to obtain is small, this could cause confusion. For example, a user may not remember which authenticator is used for which site or may not have a given authenticator in their possession at the time of login. In addition, the spec allows software authenticators, which should use a secure enclave for storing private keys. If a user is required to own multiple authenticators, along with other secure applications, storage in an enclave could become scarce.

We also consider issues related to privacy in WebAuthn. The WebAuthn spec has been designed with privacy in mind, based on the idea that authenticators generate a separate public and private key pair for each relying party. Ideally, this means that a relying party learns no information about the user from their authentication and colluding relying parties cannot track users across websites. There are, however, a number of ways in which privacy can be violated in WebAuthn:

- *Privacy leaks are possible:* Relying parties prompt users for a username or email address, thus colluding relying parties could track users if they consistently use the same information. This is the same problem that exists for current password-based authentication.

- *Tracking is possible:* Relying parties control the type of attestation used. In the worst case, attestation could be used to identify and track an individual user, because each authenticator may have a unique attestation key. The WebAuthn spec advises manufacturers of authenticators to assign non-unique attestation keys in batches of 100,000, but there is no way to enforce this and a manufacturer may simply be careless. The spec also allows for anonymization certificate authorities, which help a user to hide their identity, but relying parties may require that this is disabled.

Users will thus need significant help with ensuring they do not unknowingly reveal metadata or use authenticators or attestation methods that can be used to track them.

4 Let's Authenticate

Our design of the Let's Authenticate architecture is based on three goals: (a) automated issuance of certificates for registration and login, (b) easy account recovery in case of lost authenticators, and (c) preserving privacy for users. We describe our architecture, explain how it meets these goals, and discuss how we overcome some of the challenges identified in the WebAuthn specification.

4.1 Architecture

Let's Authenticate consists of four entities: a website, a client device, an authenticator, and a certificate authority. As with WebAuthn, the website can prompt the user to register or login. The user interacts with the web browser on the client device and an authenticator, which can be built into the client device or be a separate device like a smartphone. The authenticator uses the certificate authority to obtain certificates that allow it to register and login with the website.

To illustrate the architecture and our design goals, we walk through the account creation process and the registration/login process. We consider the case where the authenticator is a smartphone.

To create an account, the authenticator prompts the user to enter a username and password. The authenticator checks with the Certificate Authority to ensure the username is unique and creates an account with the Certificate Authority using this username and password combination.

Figure 1 shows the registration/login process.

1. When the user wishes to register or login with a website, the website provides a unique code to the client device.

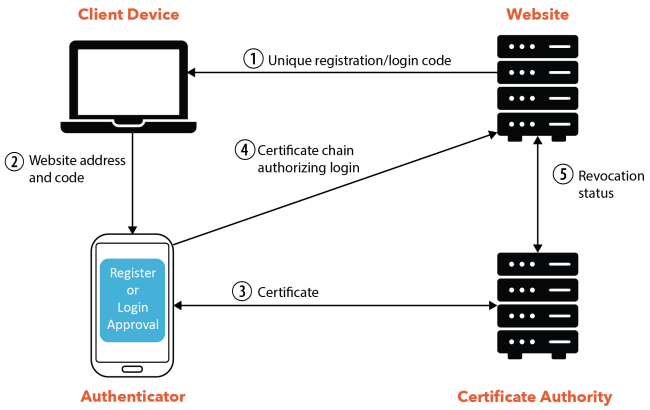

Figure 1: Let's Authenticate registration/login process

2. The web browser sends the website address and its code to the authenticator (e. g., using CTAP2) or the user enters the code on the authenticator of their choice (e. g., by scanning a QR code or manually entering it).

3. If the user approves the registration or login with the website on the authenticator, then the authenticator requests a certificate for this website. To do this the authenticator will generate a unique identifier using a one-way hash composed of the username, password and website origin. The certificate authority issues a certificate for that identifier and stores the unique identifier with that user's account.

4. The authenticator sends the website a certificate signing the code with one of its key pairs, along with the certificate signing that key and the unique identifier from the CA.

5. The website verifies the certificates and checks revocation status, then optionally creates an account for that unique identifier and logs in the client device.

We note that registration and login are automated. Because the user obtains a certificate attesting to a unique identity, the relying party can check whether an account already exists create one if needed. The user only needs to be prompted regarding whether they want a new account created for this web site. This greatly simplifies the burden for the user.

Moreover, by using certificates tied to a unique identifier associated with the user's account, account recovery is very simple. If a user loses their authenticator, they can obtain a new authenticator, and from that authenticator enter their account credentials. Once they prove ownership of the username/password, and generate new public/private key pairs, the Let's Authenticate CA can provide them with certificates for their unique identifiers. The user can recover the unique identifiers because they are composed of the username, password, and website address. They can then login to the accounts they own with these certificates.

Let's Authenticate protects privacy for users by ensuring that each website receives a certificate for a different unique identifier, so colluding websites cannot track users. Certificates contain no identifying information. In addition, the Certificate Authority issues certificates for the unique identifiers but does not receive information about which websites the user logs into. The Certificate Authority cannot reverse the one-way hash used for the identifier and does not know the user's password so it cannot compute hashes itself.

Let's Authenticate also automates revocation for users. A user can enter a name that identifies the authenticator. When a user loses an authenticator, they can revoke access by using a new or backup authenticator and proving ownership of the username/password. The CA can keep a Certificate Revocation List or CRL listing the certificates that have been revoked for the lost authenticator. Websites can periodically download updates to the CRL in aggregate, so that Certificate Authorities do not learn which certificates are being queried.

We note that certificates should be protected in case the authenticator is lost. Private keys can be stored in secure enclaves and the certificate store can be protected with separate authentication, such as a PIN or biometric.

4.2 Account challenges

We currently use a password as the method for proving ownership of an account. A traditional username/password combination is familiar to users and could serve as a good bridge to stronger cryptographic authentication. The primary difficulty is encouraging users to adopt strong passwords, similar to the need for a strong master password for a password manager. Users will need to remember this password in case of authenticator loss, or to setup additional authenticators, and could be encouraged to keep the password offline, such as by writing it down and keeping it in a safe place. To prevent theft and online guessing attacks, the CA uses a PAKE protocol, such as OPAQUE [6], where all that is disclosed is a cryptographic proof of knowledge of the password. It is important that the CA does not know the user's password because this prevents them from deriving unique identifiers for websites.

It may be possible to design other challenge methods, for example to avoid reliance on users needing to choose and remember a strong password. One such challenge could be performed using secure messaging, such as Signal or iMessage. This allows the use of familiar text message verification, while allowing for stronger security properties than SMS by using end-to-end encryption.

4.3 Certificates vs. Keys

Using certificates for authentication instead of bare keys provides some significant advantages. First, registration and login

can be automated across multiple authenticators, since the Certificate Authority attests to ownership of a unique identifier. Second, recovery is simple because a user can easily obtain a new certificate for a unique identifier once they prove their identity to a CA. Third, certificates can prevent authenticator bloat because as long as a user has a valid certificate for an account, it doesn't matter which authenticator they use. At first glance, certificates appear to offer worse privacy because usually certificates contain metadata identifying a party. However, by avoiding additional metadata such as a name or email address and avoiding reusing identifiers we can ensure certificates don't enable identification or tracking.

5 Discussion

We compare authentication schemes in Table 1 using a subset of relevant properties from *The Quest To Replace Passwords*, by Bonneau et al. [1]. Ratings for passwords, LastPass, and Mozilla Persona are taken directly from their work. We rate WebAuthn and Let's Authenticate using their definitions for the properties. The properties are grouped, in order, as usability, deployability, and security/privacy properties.

Memory-wise Effortless: A user does not have to remember any kind of secret. *Quasi* support is granted if a user only has to remember a single secret. WebAuthn support depends on whether a given authenticator uses a PIN or biometric for user verification. Let's Authenticate has *quasi* support because a user has to remember a single password.

Scalable for Users: The scheme does not increase effort required by the user for each successive account. WebAuthn does not support thus property because each website requires an individual registration. Let's Authenticate is scalable because a single registration process with the Certificate Authority is all that is required.

Nothing to Carry: A user does not have to carry an extra device in order to use the scheme. *Quasi* support is given if the device is something that the user would carry everywhere already, such as a cellular device, excluding mobile computers and tablets. WebAuthn and Let's Authenticate support depend on the authenticator, since these could be a physical token or software that is included with the client device.

Easy to Learn: A user, with no experience with the scheme, can figure out how to use the system with minimal trouble and recall how to use the scheme. WebAuthn and Let's Authenticate were both deemed to support this property because they are similar to other schemes given this rating. However, both systems require user studies to demonstrate adequate ease of use.

Efficient to Use: The time the user spends when they try to authenticate is reasonably short, including registration with a website. WebAuthn depends on how many authenticators the user owns. Since each website can specify different requirements, a user may have to fiddle with a variety of devices or software. Let's Authenticate is rated as supporting

Table 1: Rating of authentication systems

System	*Memory-wise Effortless*	*Scalable for Users*	*Nothing to Carry*	*Easy to Learn*	*Efficient to Use*	*Easy Recovery from Loss*	*Negligible Cost to User*	*Server Compatible*	*Resilient to Phishing*	*Resilient to Theft*	*No Trusted Third Party*	*Unlinkable*
Passwords			●	●	●	●	●	●		●	●	●
LastPass	○	●	○	●	●	○	○	●	●	●		●
Mozilla Persona	○	●	●	●	●	●	●			●		
WebAuthn	◖		◖	●	◖		◖		●	◖	◖	◖
Let's Authenticate	○	●	◖	●	●	●	●		●	●		●

● full support, ○ *quasi* support, ◖ support depends on the authenticator type or the website, (blank) no support

the property because a user is only required to use a single authenticator of their choosing.

Easy Recovery from Loss: A user can conveniently regain the ability to authenticate if a token is lost or credentials forgotten. *Quasi* support is given if recovery is possible but extra time or inconvenience is incurred. WebAuthn requires user to separately pre-register backup authenticators for each website. Let's Authenticate makes it easy to recover an account by proving ownership of a username/password. In addition, Let's Authenticate could provide backup account recovery mechanisms if the user forgets their password.

Negligible Cost to User: The total cost for each user of the scheme, including costs client costs for extra devices, and any costs to the website for equipment and software, is negligible. *Quasi* support is given if there is small but non-negligible cost. WebAuthn support depends on the authenticator requirements for each website. It is possible for each website to require the use of a different, specific authenticator and the user may be required to purchase multiple devices. Let's Authenticate does not require the user to purchase an additional device.

Server Compatible: The scheme is compatible with text-based passwords, so websites do not need to change their authentication method. Neither WebAuthn nor Let's Authenticate are server compatible.

Resilient to Phishing: An attacker cannot collect credentials that can later be used to impersonate a user. Both WebAuthn and Let's Authenticate are resilient to phishing since cryptographic credentials are used for each site and these are negotiated by the browser or authenticator based on the website's identity.

Reslient to Theft: If the scheme uses a physical device, another person who gains access to the device cannot use the object for authentication. *Quasi* support is awarded if

protection, such as a PIN is used. Some WebAuthn authenticators may use a PIN or biometric, but others may not. Let's Authenticate authenticators have the same property, but the architecture includes additional protection from theft by allowing the owner of a device to deauthorize it after proving knowledge of their account password.

No Trusted Third Party: The scheme does not rely on a trusted third party. Websites using WebAuthn may rely on attestations from manufacturers of authenticators, *e.g.* to ensure that the user must enter a PIN or biometric to use the device. Websites that rely on these would be using a trusted third party. Let's Authenticate uses identity providers as trusted third parties.

Unlinkable: If websites collude, they cannot learn from information passed during authentication whether the same user is authenticating to both. For WebAuthn this depends on whether manufacturers of authenticators use non-unique attestation keys. For Let's Authenticate, each website is sent a different certificate using a unique identifier and a unique private key. The unique identifier is opaque to websites and Certificate Authorities, and linkability requires collusion between a Certificate Authority (which has a list of identifiers per user) and websites.

Overall, we find that Let's Authenticate has significant advantages relative to WebAuthn, particularly that it is scalable for users and easy recovery from loss. It likely also has advantages in being efficient to use, having negligible cost to users, resilience to theft, and being unlinkable, since these all depend on how authenticators are used. These changes come at the cost of using a trusted third party to issue certificates. As compared to Mozilla Persona, Let's Authenticate has the advantage of being resilient to phishing and unlinkable, at the cost of not currently being compatible with websites.

6 Future Research

Future work must be done to fully examine our proposed system. We plan to conduct an in-depth security and privacy analysis. We also plan to run a longitudinal user study to ensure Let's Authenticate certificates meet the goals we have outlined. User opinions are of particular interest, specifically regarding password replacement, ease of use, and the security and privacy options provided by the architecture. We are also exploring designs for additional account challenges and are considering how using short-lived certificates could simplify revocation.

References

[1] 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 *2012 IEEE Symposium on Security and Privacy*, pages 553–567. IEEE, 2012.

[2] Danny De Cock, Christopher Wolf, and Bart Preneel. The Belgian electronic identity card (overview). In *Sicherheit*, volume 77, pages 298–301, 2006.

[3] Dinei Florencio and Cormac Herley. A large-scale study of web password habits. In *Proceedings of the 16th International conference on World Wide Web*, pages 657–666. ACM, 2007.

[4] Dinei Florêncio, Cormac Herley, and Paul C Van Oorschot. An administrator's guide to Internet password research. In *28th Large Installation System Administration Conference*, pages 44–61, 2014.

[5] Cormac Herley and Paul Van Oorschot. A research agenda acknowledging the persistence of passwords. *IEEE Security & Privacy*, 10(1):28–36, 2011.

[6] Stanislaw Jarecki, Hugo Krawczyk, and Jiayu Xu. Opaque: an asymmetric PAKE protocol secure against pre-computation attacks. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 456–486. Springer, 2018.

[7] Tarvi Martens. Electronic identity management in estonia between market and state governance. *Identity in the Information Society*, 3(1):213–233, 2010.

[8] Roel Peeters, Jens Hermans, Pieter Maene, Katri Grenman, Kimmo Halunen, and Juha Häikiö. n-auth: Mobile authentication done right. In *Proceedings of the 33rd Annual Computer Security Applications Conference*, pages 1–15. ACM, 2017.

[9] Aaron Smith. What the public knows about cybersecurity. Pew Research Center, 2017. https://www.pewinternet.org/2017/03/22/what-the-public-knows-about-cybersecurity/.

[10] W3C. Web authentication: An API for accessing public key credentials, March 2019. https://www.w3.org/TR/webauthn/cross-platform-attachment.

[11] Jeff Yan, Alan Blackwell, Ross Anderson, and Alasdair Grant. Password memorability and security: Empirical results. *IEEE Security & Privacy*, 2(5):25–31, 2004.

[12] Bo Zhu, Xinxin Fan, and Guang Gong. Loxin—a solution to password-less universal login. In *2014 IEEE Conference on Computer Communications Workshops*, pages 488–493. IEEE, 2014.