Toward Cloud-based FIDO Authentication with Secure Credentials Recovery

Momoko Shiraishi
The University of Tokyo, Japan

Takahiro Shinagawa
The University of Tokyo, Japan

ABSTRACT

FIDO is an alternative to password authentication for logging into web services securely through public key authentication. However, credentials for FIDO is unique to physical authentication devices, leading to lockout from associated web services in case access to the devices is lost. We propose Cloudauthn, a cloud-based FIDO authentication scheme that can perform key management and recovery without compromising security. To avoid storing credentials directly in untrusted clouds, Cloudauthn introduces *certifying keys* that serve as proxies for device's credentials and are securely managed in TEEs in the cloud. To facilitate key recovery and revocation for many web services, Cloudauthn revokes old credentials and register new ones efficiently using the certifying keys.

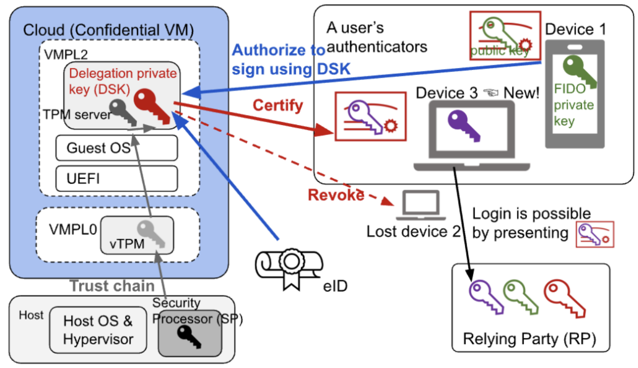

Figure 1: Overview of Cloudauthn

1 INTRODUCTION

Fast IDentity Online (FIDO) [4] has attracted significant attention as a notable alternative to traditional password authentication. While password authentication is inherently difficult to counter deceptive attacks against humans such as phishing and man-in-the-middle attacks (MITM), FIDO can mitigate such attacks by leveraging a public key authentication scheme designed for use in web services.

However, FIDO authentication still has challenges in securely recovering access to services when the access to authenticator devices is lost. To preserve security, FIDO credentials, which include private keys, are stored on devices, such as smartphones or security keys. Therefore, if the devices are damaged, lost, or stolen, the user may be locked out of FIDO-authenticated web services.

There are several issues with secure credentials recovery. The first is *credentials availability*; access to credentials must be easily recoverable even if one or all authenticator devices are lost to withstand worst case scenarios such as disasters. The second is *credentials security*; credentials should not be extracted from authenticator devices to prevent credentials leakage. The third is *recovery scalability*; access to previously registered web services can be quickly restored when a new authenticator device is registered even if there are hundreds of registered web services [6].

Previous studies have proposed several approaches to storing credentials in alternative locations. For example, some studies proposed having a backup token dedicated to recovery [3, 8], while others proposed introducing a group signature that allows login from multiple devices [2]. These approaches, however, could result in a complete loss of credentials when all authenticator devices are lost. Passkey [1] achieves credentials availability by copying credentials across multiple devices, but extracting from authenticator devices is undesirable for the security reason.

We propose Cloudauthn, a cloud-based FIDO authentication scheme that can achieve credentials availability and scalability without compromising the security. The key idea of Cloudauthn is to introduce certifying keys that act as proxies for authenticator devices. To achieve credentials availability, Cloudauthn stores certifying keys in the cloud and enables authenticator devices whose FIDO authentication public keys are certified by the certifying keys, even new ones, to log in to previously registered web services. Even If all authenticators are lost, existing identity proofing methods, such as a government eID or ePassport, could be used to certify new authenticators using certifying keys. To achieve credentials security, Cloudauthn encrypts the certifying keys with authenticator devices and manages them in the Trusted Execution Environments (TEEs) so that even cloud vendors cannot access the certifying keys. To achieve recovery scalability, under Cloudauthn, the recovery procedure, i.e., the task of associating a new authenticator with a certifying key, requires execution merely once. Also, a revoked authenticator list certified by certifying keys is sent to every website all at once.

2 DESIGN AND IMPLEMENTATION

Design. Figure 1 shows an overview of Cloudauthn. The core idea to ensure credentials availability is certifying keys. The key is called a 'certifying key' because the key certifies that a FIDO key belongs to the legitimate user. Websites allow login requests from any authenticator, provided that the authenticator's FIDO authentication public key has been certified (signed) by the certifying key.

To preserve the security of certifying keys, Cloudauthn maintain the keys in a TEE; our current implementation uses a confidential VM based on AMD SEV-SNP as the TEE. Within this VM, a vTPM (virtual Trusted Platform Module) is emulated and also a TPM server is deployed separately. The TPM server stores certifying keys for each user's authenticator in NV (non-volatile) files. This separation ensures confidentiality and integrity, as the vTPM's state data stored by a hypervisor is otherwise vulnerable to access from the untrusted hypervisor [7].

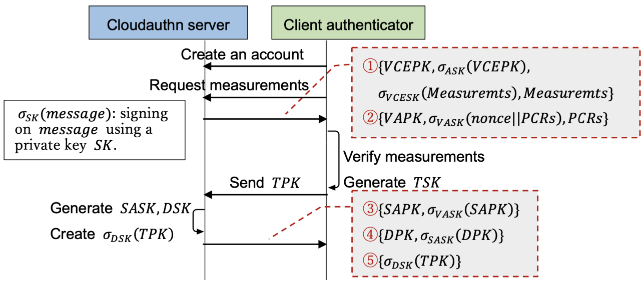

Figure 2: Registration with Cloudauthn Service

Implementation. When a VM starts, the AMD-SP (Security Processor) measures the VM's vTPM. If the measurement is correct, an attestation key pair ($VASK$, $VAPK$) is generated in the vTPM. The public key ($VAPK$) is included in an attestation statement, signed by the AMD versioned chip endorsement key ($VCESK$).

Before interacting with RPs, a user sets up an account with Cloudauthn and registers their authenticator. The key steps, as outlined in Figure 2, include:

- Through attestation, AMD-SP validates the vTPM and UEFI binary(①), while vTPM confirms the guest OS and apps, including the TPM server(②).
- Proofs using the TPM server's attestation key pair ($SASK$, $SAPK$) show that the certifying keys (DSK, DPK) are within the legitimate TPM server, and are sent to the user (③, ④).
- The signature ⑤ is created by a certifying key on a temporary public key. Users have a sufficiently large number of temporary signing and certifying keys equivalent to the number of RPs they will use. Each temporary public key (TPK) is signed by a certifying key.

In the TPM server, each NV file is encrypted with a symmetric key. This key is doubly wrapped by a VM's symmetric key and a user's asymmetric key to prevent certifying key leakage.

To add a new authenticator in Cloudauthn, users authenticate with an existing one, then decrypt and duplicate the NV file. Following the registration in Figure 2, the NV file is re-encrypted using the new authenticator's key. Users can also use methods like eIDs for manipulating certifying keys, distributing the NV file's encryption key across clouds through secret sharing. Sensitive data, such as eID data, are encrypted with the user's own key, ensuring they remain undisclosed to the cloud.

After registering an authenticator with Cloudauthn, the user can register it with any RP and log in, similar to the existing FIDO process. The difference lies in the user also sending a set of data related to the certifying key including ① - ⑤. ① - ④ ensures the legitimacy of the location where the certifying key is generated, forming a trust chain from the AMD root key to the certifying key ($AMD\ root\ key \rightarrow ASK(AMD\ signing\ key) \rightarrow VCESK \rightarrow VASK \rightarrow SASK \rightarrow DPK$). The legitimacy of where both the FIDO key (FSK, FPK) and the temporary key are generated is proved by the authenticator's attestation key's signature ($\sigma_{AASK}(...||challenge||TPK||FPK||...)$ (⑥)). Since the RP specifies the FIDO key pair in terms of algorithm and other criteria, the temporary key used before account opening is prepared. Now both the temporary and FIDO keys are signed by the same attestation key (⑥) and the temporary key is linked to the certifying key(⑤ and an additional signature $\sigma_{TSK}(challenge)$(⑦)) , therefore the FIDO key becomes indirectly linked to the certifying key. The RP then stores both the certifying key and FIDO key. The certifying key's association with the RP's domain is cloud-synchronized and shared across all user authenticators.

At the RP, since the RP recognizes the user's consistent certifying key, login is possible with any authenticator, even unregistered ones, by presenting the ⑤, ⑥ and ⑦ for that authenticator.

If a hardware authenticator becomes inaccessible, recovery starts with Cloudauthn login using an available hardware authenticator or the alternative identity verification method. The NV file of the inaccessible device is deleted, and the device is added to a revocation list. The user then shares the list with all relevant domains. If all hardware authenticators are lost, a new one is registered to Cloudauthn via the hardware-independent method, allowing the user to resume regular logins at each RP with the new authenticator.

Performance evaluation. We have deployed the Cloudauthn service on an Azure instance with an AMD EPYC 7763v CPU (32-core, 128 GB RAM) and a RP on a GCP instance with an Intel Xeon E5-2696V3 CPU (16-core, 60 GB RAM). Both run on Ubuntu 20.04 LTS. The TPM server uses the ibmtpm module [5]. The client device is a MacBook Pro 13.3-inch mid-2020 with a 2.3 GHz Core i7, 16 GB RAM, and Touch ID, running Big Sur 11.7.10 with Safari 14.1.3. Registration time with Cloudauthn increases with more certifying keys; it takes about 57.75 seconds for 10 keys and 500.81 seconds for 100 keys. Registering with a RP takes 621.89 milliseconds. These results show that the proposal is feasible for practical use.

Future Work. More refined proofs of security for the proposed approach are expected. A detailed comparison with other methods in terms of security and performance is also desired.

REFERENCES

[1] Apple. 2023. Supporting Passkeys. https://developer.apple.com/documentation/authenticationservices/public-private_key_authentication/supporting_passkeys. (2023). Online; accessed 2023-11-01.

[2] Sunpreet S Arora, Saikrishna Badrinarayanan, Srinivasan Raghuraman, Maliheh Shirvanian, Kim Wagner, and Gaven Watson. 2022. Avoiding lock outs: Proactive FIDO account recovery using managerless group signatures. *Cryptology ePrint Archive* (2022).

[3] Nick Frymann, Daniel Gardham, Franziskus Kiefer, Emil Lundberg, Mark Manulis, and Dain Nilsson. 2020. Asynchronous Remote Key Generation: An Analysis of Yubico's Proposal for W3C WebAuthn. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. 939–954.

[4] Jeff Hodges, J Jones, Michael B Jones, Akshay Kumar, and Emil Lundberg. 2021. Web authentication: An API for accessing public key credentials level 2. *World Wide Web Consortium, Cambridge, MA, USA* (2021).

[5] IBM. 2023. Software TPM 2.0. https://sourceforge.net/projects/ibmswtpm2/files/latest/download. (2023). Accessed: 2023-11-01.

[6] Sanam Ghorbani Lyastani, Michael Schilling, Michaela Neumayr, Michael Backes, and Sven Bugiel. 2020. Is FIDO2 the kingslayer of user authentication? A comparative usability study of FIDO2 passwordless authentication. In *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 268–285.

[7] Vikram Narayanan, Claudio Carvalho, Angelo Ruocco, Gheorghe Almási, James Bottomley, Mengmei Ye, Tobin Feldman-Fitzthum, Daniele Buono, Hubertus Franke, and Anton Burtsev. 2023. Remote attestation of SEV-SNP confidential VMs using e-vTPMs. *arXiv preprint arXiv:2303.16463* (2023).

[8] Alex Takakuwa. 2019. *Moving from Passwords to Authenticators.* Ph.D. Dissertation.