December 2022

TWO-PARTY WEBAUTHN TOKEN ACTIVATION

Richard Barnes

Adam Goodman

Carson Hoffman

TWO-PARTY WEBAUTHN TOKEN ACTIVATION

AUTHORS:
Richard Barnes
Adam Goodman
Carson Hoffman

ABSTRACT

In order to prevent attacks such as phishing, an enterprise needs their users to log in using a World Wide Web Consortium (W3C) Web Authentication (WebAuthn)-based authenticator. Current WebAuthn authenticator devices present a number of problems for an enterprise. For example, outsourcing authentication device distribution logistics to a device vendor brings great operational benefits to an enterprise, however this traditionally requires that a large amount of trust be placed in the vendor. Techniques are presented herein that split an authenticator's secret between the two parties (i.e., an enterprise and a vendor), requiring active collaboration by the parties to issue an authenticator. This prevents both the device vendor alone, and read-only compromises of the enterprise, from issuing unauthorized or duplicated keys, while maintaining the ability to delegate logistics management to the vendor.

DETAILED DESCRIPTION

In order to prevent attacks such as phishing, an enterprise needs their users to log in using a World Wide Web Consortium (W3C) Web Authentication (WebAuthn)-based authenticator. Such an authenticator is a secure hardware device that implements the WebAuthn protocol. The narrative that is presented below focuses on dedicated hardware authenticators that lack Internet connectivity and do not allow secret keys to be moved across devices. However, the techniques presented herein (which will be described and illustrated below) may also be applied to software-based authenticators, but with differing tradeoffs.

Current WebAuthn authenticator devices present several problems for an enterprise. For example, delegating authenticator provisioning logistics to an authenticator vendor is highly desirable, but such a delegation requires trusting that the vendor will not create illicit authenticators that would allow them to log in as a user. With hardware authenticators, it is not typically possible for the vendor to communicate with the authenticator after the authenticator device has been sent to a user. "Cloning" is the primary attack that is possible under this constraint; attacks based on injecting other faults into the authenticator (e.g., bad randomness) require future interaction with the authenticator. Further, if a user loses their authenticator and then obtains a new one, they must re-enroll the new authenticator with every website where they used the old one. Additionally, the privacy features of WebAuthn may prevent an enterprise from verifying that an authenticator that is presented by a user is one that the enterprise issued to the user.

To address the challenges that were described above, techniques are presented herein that support splitting an authenticator's secret between an enterprise and a vendor, requiring active collaboration by those parties to issue an authenticator.

Within aspects of the techniques presented herein, there are three principal actors – a user, an enterprise, and a vendor. The enterprise and the vendor are represented by computer systems which may or may not involve human operators. The user is a human that employs a computer system to interact with the enterprise's systems and the authenticator.

Within further aspects of the techniques presented herein, there are three mandatory processes that are of interest – issuance, initialization, and login. Additionally, there is one optional process encompassing reissuance. The reissuance process is optional on a per-authenticator basis; an enterprise and a vendor must agree at the time that an authenticator is created whether an authenticator may be reissued.

A first process, as referenced above, encompasses issuance and it may be employed when an enterprise wishes to provide an authenticator to a new user. For this process, an enterprise can instruct a vendor to send an authenticator to a user. Then, the enterprise and the vendor engage in a distributed key generation protocol to generate a signature key pair (skA, pkA) for the authenticator, where pkA is known to both the enterprise and the vendor; skA is secret-shared into to two parts sksE and sksV which are known to the enterprise and vendor, respectively; and the enterprise and authenticator vendor only know their respective shares of skA (i.e., they do not know skA itself).

Next, the enterprise associates to the user's account the public key pkA and the enterprise sends its share sksE to the user through a channel that is independent of the vendor (such as, for example, email or postal mail). The vendor then installs its share sksV on the authenticator and sends the authenticator to the user. It is important to note that at this point, the authenticator is in an "uninitialized" state and cannot be used to traceably log in to an enterprise website.

If the enterprise and vendor agree to enable reissuance for the instant authenticator, then the enterprise stores its key share sksE and the vendor stores its key share sksV. Otherwise, the key shares sksE and sksV are destroyed once they are transmitted to the user (in a vendor-independent channel and in the token, respectively)

Figure 1, below, depicts elements of an exemplary arrangement that supports an issuance process according to aspects of the techniques presented herein and reflective of the above discussion.

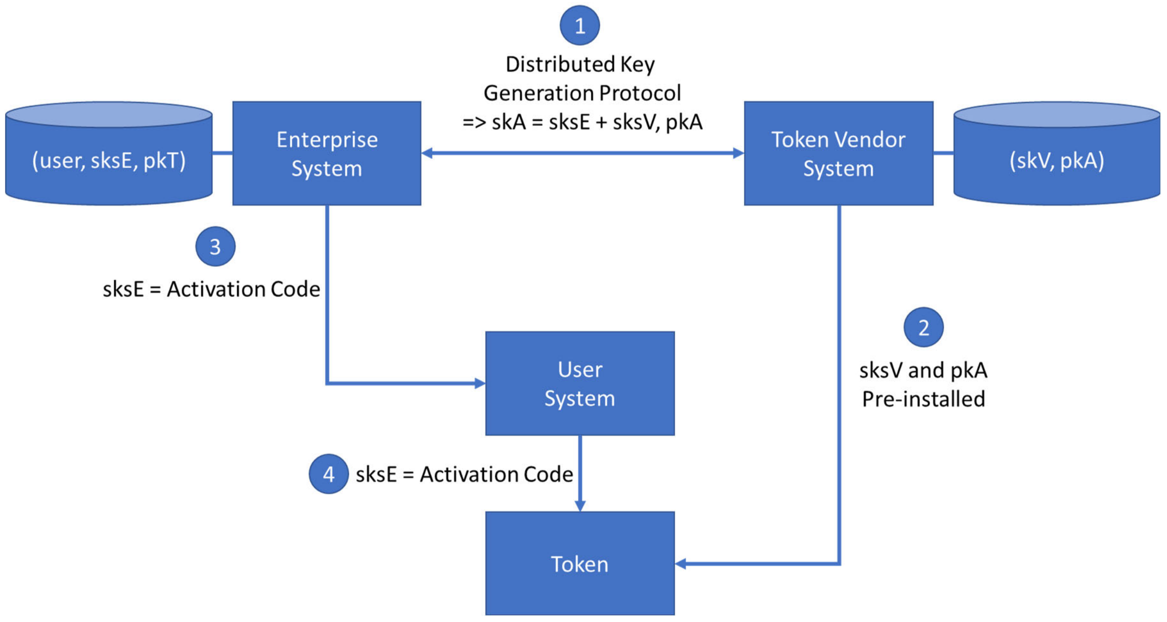

Figure 1: Exemplary Issuance Process

A second process, as referenced above, encompasses initialization and it may be employed when a user wishes to set up an authenticator for subsequent use when logging in. In this process, a user can receive the key share sksE from an enterprise and an authenticator (containing the key share sksV) from a vendor. Then, the user enters the enterprise's key share into the authenticator, possibly by attaching the authenticator to a computer system and entering the key share by means of appropriate software or possibly using an interface on the authenticator itself. Next, the authenticator combines the two key shares (sksE and sksV) to recover the digital signature key skA and then deletes the key shares. The authenticator is now initialized and may be used to securely log into an enterprise website.

A third process, as referenced above, encompasses logging in and it may be employed when a user wishes to log into a website. WebAuthn requires authenticators to generate unique, unlinkable keys for each web origin. An authenticator that is configured according to the techniques presented herein generates keys that are unlinkable by anyone who does not know the public key pkA.

Specifically, the authenticator derives origin-specific key pairs from the authenticator key pair (skA, pkA) using a key derivation function that provides the following algorithms – MapSK(sk, info), which maps a private key "sk" and a byte string "info" to a second private key, and MapPK(pk, info), which maps a public key "pk" and a byte string "info" to a second public key.

The two above-described algorithms should meet the following constraints – first, given a key pair (sk, pk) and a byte string "info", MapPK(pk, info) is the public key corresponding to the private key MapSK(sk, info); second, it must be difficult to determine the input private key "sk" given "info" and the output private key MapSK(sk, info); and third, it must be difficult to determine the input public key "pk" given "info" and the output public key MapPK(pk, info). One specific algorithm that meets the above-described criteria is specified in the Bitcoin Improvement Proposal (BIP) 32.

The "info" that is provided to the above-described algorithms must be specific to the origin for which the key pair is generated to preserve the unlinkability of keys across origins. For example, "info" might comprise the origin and a random credential identifier (ID).

When the user employs the authenticator to log into a website that is affiliated to the enterprise, the authenticator provides the derived per-origin public key pkAO = MapPK(pkA, info). If the enterprise also has access to the "info" it may re-run the algorithm MapPK to confirm that pkAO is indeed equal to MapPK(pkA, info).

Figure 2, below, depicts elements of an exemplary arrangement that supports a login process according to aspects of the techniques presented herein and reflective of the above discussion.

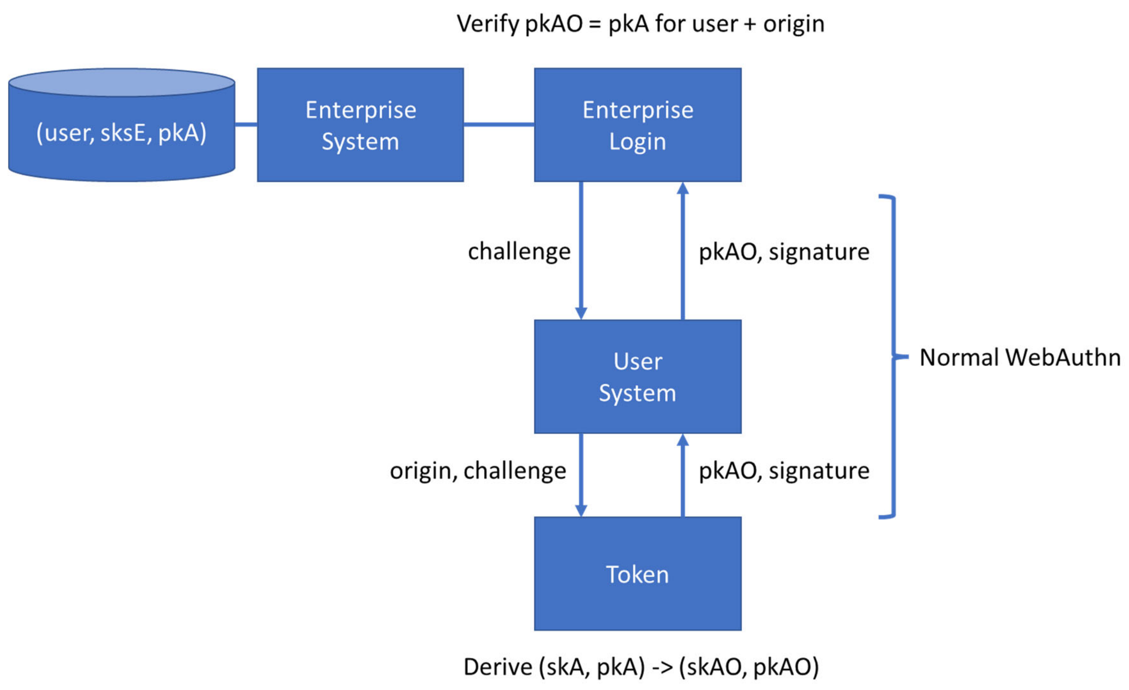

Figure 2: Exemplary Login Process

A fourth process, as referenced above, encompasses reissuance and may be employed when an enterprise wishes to send a new authenticator to a user that is equivalent to an old authenticator. In this process, instead of a distributed key generation protocol, an enterprise and a vendor engage in a resharing protocol that transforms their old key shares (sksE, sksV) into a new pair of key shares (sksE2, sksV2) that represent the same signature private key skA. The specific resharing protocol that may be used will depend upon the secret sharing scheme that is in use. The enterprise and vendor may perform such a resharing even outside of the context of a reissuance (e.g., for the compromise-recovery reasons that are noted below).

Following completion of the first step, the enterprise and the vendor each hold new, different shares of the same digital signature key. Those shares may be distributed to a user and then employed by the user to initialize an authenticator in the same way as for an initial issuance. Because the digital signature key on the new authenticator is the same as the old authenticator, the new authenticator will derive the same origin-specific keys, allowing the new authenticator to be used everywhere the old authenticator was registered with no additional steps.

Figure 3, below, depicts elements of an exemplary arrangement that supports a reissuance process according to aspects of the techniques presented herein and reflective of the above discussion.

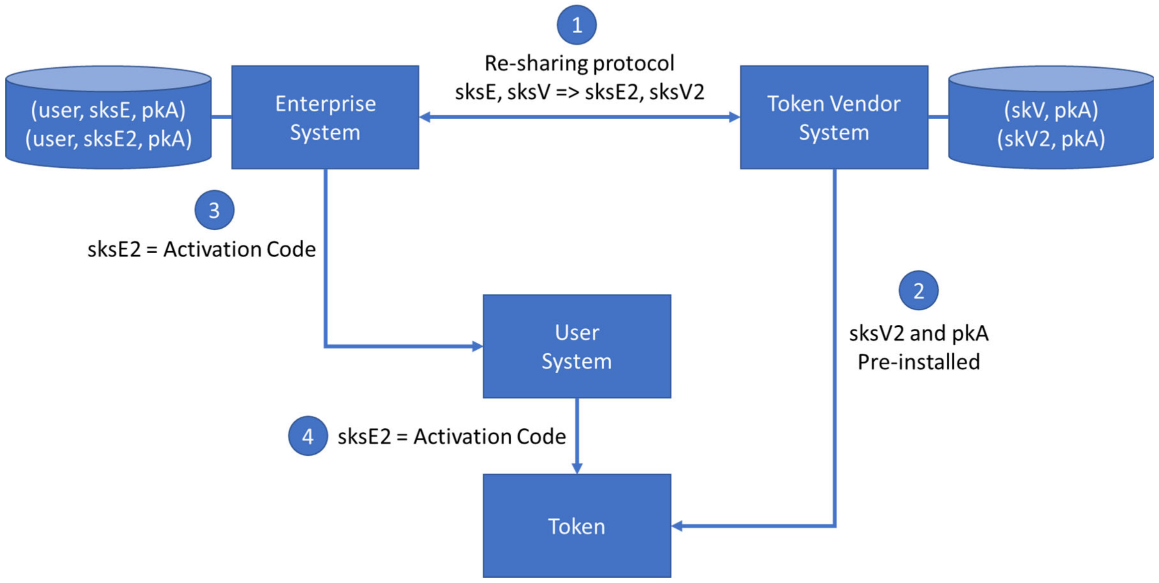

Figure 3: Exemplary Reissuance Process

The above-described scheme enforces a "two party compromise" rule for authenticators that may be reissued. Because a vendor only has one share of the digital signature key, it is incapable of creating the digital signature key that is needed to impersonate a user. The same goes for an enterprise (e.g., in the event of a data breach). A user's authenticator is only susceptible to cloning if both the enterprise and the vendor are compromised. If reissuance is not enabled for an authenticator, then no risk would arise as a result of a breach at either entity.

The resharing protocol in the reissuance step mitigates the risk of long-term share compromise. Compromising corresponding shares from both an enterprise and a vendor reveals the full secret, however compromising shares across reshare cycles (when one share is before a refresh and one share is after) does not reveal any information about the secret. Thus, the authenticator signing key skA is only compromised if an attacker compromises both the enterprise and the vendor within the same refresh window.

In summary, techniques have been presented that split an authenticator's secret between an enterprise and a vendor, requiring active collaboration by the two parties to issue an authenticator. This prevents both the device vendor alone, and read-only compromises of the enterprise, from issuing unauthorized or duplicated keys, while maintaining the ability to delegate logistics management to the vendor.