March 2022

METHOD TO PROVIDE SECURE AND RELIABLE MFA AND SSO USING HYPER LEDGER

Niranjan M M

METHOD TO PROVIDE SECURE AND RELIABLE MFA AND SSO USING HYPER LEDGER

AUTHOR:
Niranjan M M

ABSTRACT

Multi Factor Authentication (MFA), or Two-Factor Authentication (2FA) is a method where-in users are required to present more than one type of evidence to authenticate on a system. Although MFA is by far the best defence against most password-related attacks (brute-force, credential stuffing, password spraying etc.,), but it is having its own disadvantages. Hence, we need a method to overcome the disadvantages of MFA, by removing the dependencies on mobile (authenticator app, SMS, call) or any external device without compromising on security, removing single point of failure, having less costly/complex SSO. having decentralisation with distributed authentication architecture., having add on security with Access-control architecture and having more trustworthy system. The techniques presented herein is one such method to incorporate the above using Hyper Ledger. This method mitigates the availability issues of the 2FA and SSO by customizing its functionality and distributing its data using permission blockchain (Hyper Ledger) technology over the network.

DETAILED DESCRIPTION

Multi Factor Authentication (MFA), or Two-Factor Authentication (2FA) is a method where-in users are required to present more than one type of evidence to authenticate on a system. For example, companies are using 2FA with different authentication options such as SMS, Call, Authenticator app etc., Although MFA is by far the best defence against most password-related attacks (brute-force, credential stuffing, password spraying etc.,), but it is having its own disadvantages:

- MFA increases the management complexity for both administrators and end users.
- Many less technical users may find it difficult to configure and use MFA. Requiring MFA may prevent some users from accessing the application.

- Types of MFA that require users to have specific hardware can introduce significant costs and administrative overheads.
- Users may become locked out of their accounts if they lose or are unable to use their other factors.
- MFA introduces additional complexity into the application.
- Processes implemented to allow users to bypass or reset MFA may be exploitable by attackers.
- Many MFA solutions add external dependencies to systems, which can introduce security vulnerabilities or single points of failure.

Also, 2FA with SMS authentication method is vulnerable to SMS redirect attacks and even companies are planning to discontinue this option, to align with NIST standards, InfoSec, and IT, but still, we need Mobile device for authenticator push. Hence, we need a method to overcome the above disadvantages of MFA, by

- Removing the dependencies on mobile (authenticator app, SMS, call) or any external device without compromising on security.
- Removing Single point of failure.
- To have less costly/complex SSO.
- To have decentralisation with distributed authentication architecture.
- To have add on security with Access-control architecture.
- To have more trustworthy system.

The techniques presented herein is one such method to incorporate the above using Hyper Ledger. This method mitigates the availability issues of the 2FA and SSO by customizing its functionality and distributing its data using permission blockchain (Hyper Ledger) technology over the network. The Blockchain data structure possesses inherent properties that can be useful to improve an SSO service's availability and hence, its overall functionality and reliability:

- More secure 2-Factor Authentication
- More reliable SSO
- User controlled identity

As per this method, upon successful authentication of the wireless/wired client (i.e., completion of first level of authentication), Server generates the "Trusted Token" and add to the Hyper Ledger and send the same token to the client. As part of two-factor authentication, client would update the token to the Hyper Ledger and asks Server to validate. This completes the authentication loop, and which is like the current 2FA using authenticator push/OTP. Then Server validates the token at the Hyper Ledger, which is written by the client to complete the two-factor authentication process.

"Trusted token" would be generated by the Server after it has been identified as trusted server by the Blockchain Provider (BP) as explained below:

1. To attest the Server, Blockchain Provider (BP) retrieves the Public Attestation Identity Key PK_AIK_SERVER and sends a "nonce" to the Server.
2. The Server answers with a quote that contains this "nonce" and its current Platform Configuration Registers (PCR) values that are stored inside the local TPM of the Server.
3. This quote is signed by the Private Attestation Identity Key SK_AIK_SERVER of the server to ensure the integrity of the response.
4. After the receipt of the payload from the server, BP compare the "nonce" to check the freshness of the attestation and if this matches the expected value, compare the received PCR values with a library of trusted node configurations.
5. Now onwards Server would be considered as Trusted Server.
6. Optionally BP can continue to re-validate the freshness of the attestation periodically using above steps.
7. Note here, vTPM (or software TAM) is used for the virtual/cloud/container deployments.

Later random "Token" is populated using HKDF and signed using Private Attestation Identity Key by the Server, to generate "Trusted Token", so that other servers that are part of the same blockchain (Hyper Ledger) can use Server's Public Attestation Identity Key to validate the Trustworthiness of the Token (and hence trustworthiness of the Server) during SSO.

A. Two-Factor Authentication Protocol:

Figure-1 explain this method with respect to 2FA as below:

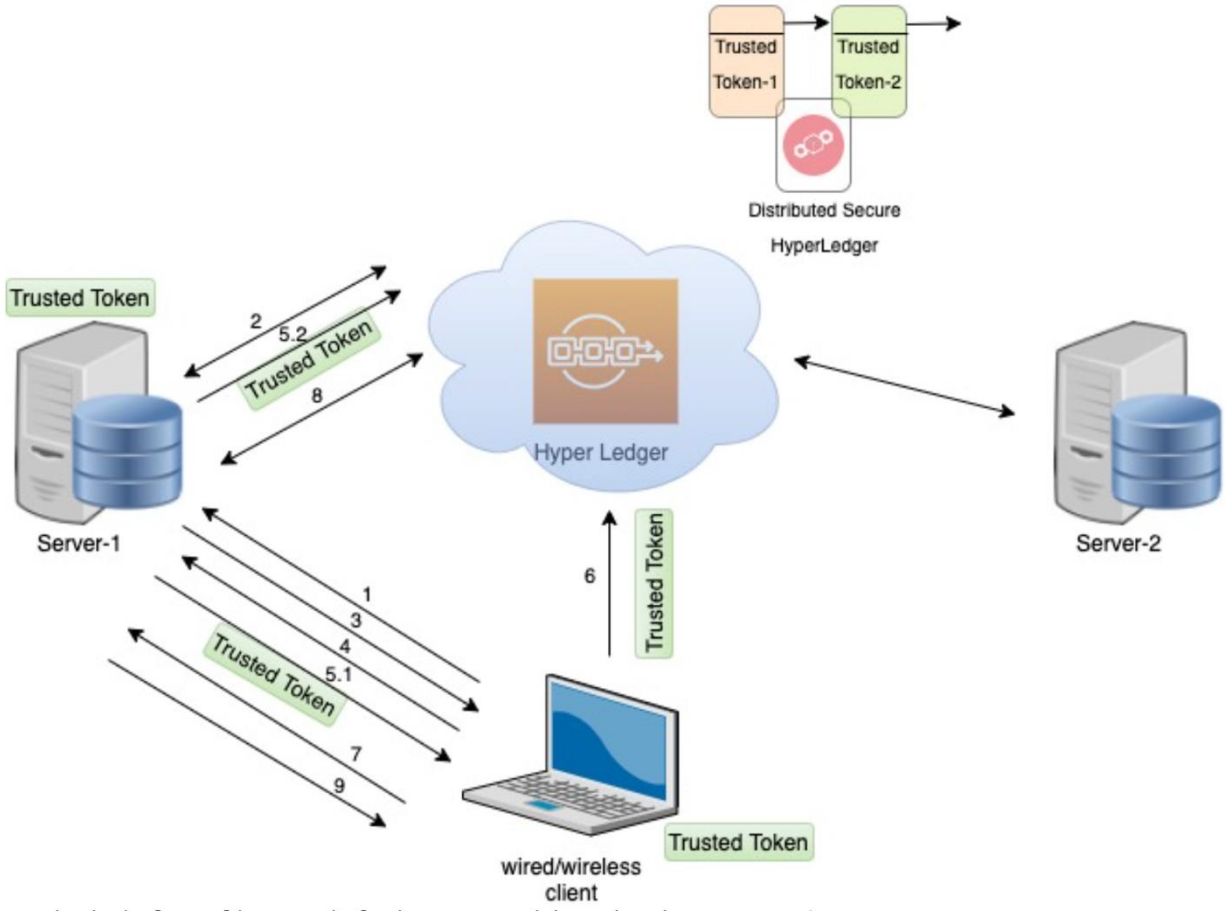

Figure-1

1. Wired/wireless client sends login request with a signed message to Server-1.
    a. Some of the client devices will have SUDI certificate stored securely in TPM (a tamper proof TAM chip viz., ACT2).
    b. In some of the client devices, certificate is downloaded manually (similar to the way we download certificate required for EAP-TLS etc.,)
    c. In some other client devices, certificate is downloaded using Mobile Device Management (MDM).
    d. The client device sign the message (containing login request, optional client device information etc.,) with private key which was provisioned using any of the above methods.

   e. Server uses the corresponding public key of the client (using certificate chain) to validate the signature of the client device.
2. Server-1 verifies the ownership of client's account via Hyper Ledger (permission blockchain).
3. Server-1 sends the login page to the client requesting the user credentials.
4. Wired/wireless client provides the login credentials.
5. Server-1 validates the credentials and authenticate the client, which completes first level of authentication, after that Server generates "Trusted Token" for this client as explained above, then
   5.1. Server-1 sends a signed trusted token to the Client.
   5.2. Also, Server-1 adds the signed trusted token to the Hyper Ledger.
6. Client writes the trusted token to the Hyper Ledger.
7. Client prompts the Server-1 to verify the trusted token written to the Hyper Ledger.
8. Server-1 verifies the trusted token at the Hyper Ledger which was written by the client.
9. This completes authentication loop and in-turn two-factor authentication succeeds.

Note:

- Trusted token added by Server-1: The signed token added by server is having session timeout and added immediately after creation along with timestamp. This is used for token validation and revocation. This is used later in SSO
- Trusted token added by Client device: Client adds the trusted token, to inform Server to validate as part of Multi-Factor Authentication (MFA). Client trusted token is also signed using client's private key to prove the identity of the client device.
- In the SSO phase: Server-2 uses the trusted token written by Server-1 having session timeout which would help in revocation of the token after session expiry. Optionally, in the SSO, client's identity can be verified using the signed token added by the client.

B. Single Sign-On Protocol:

Figure-2 explain this method with respect to SSO protocol as below:

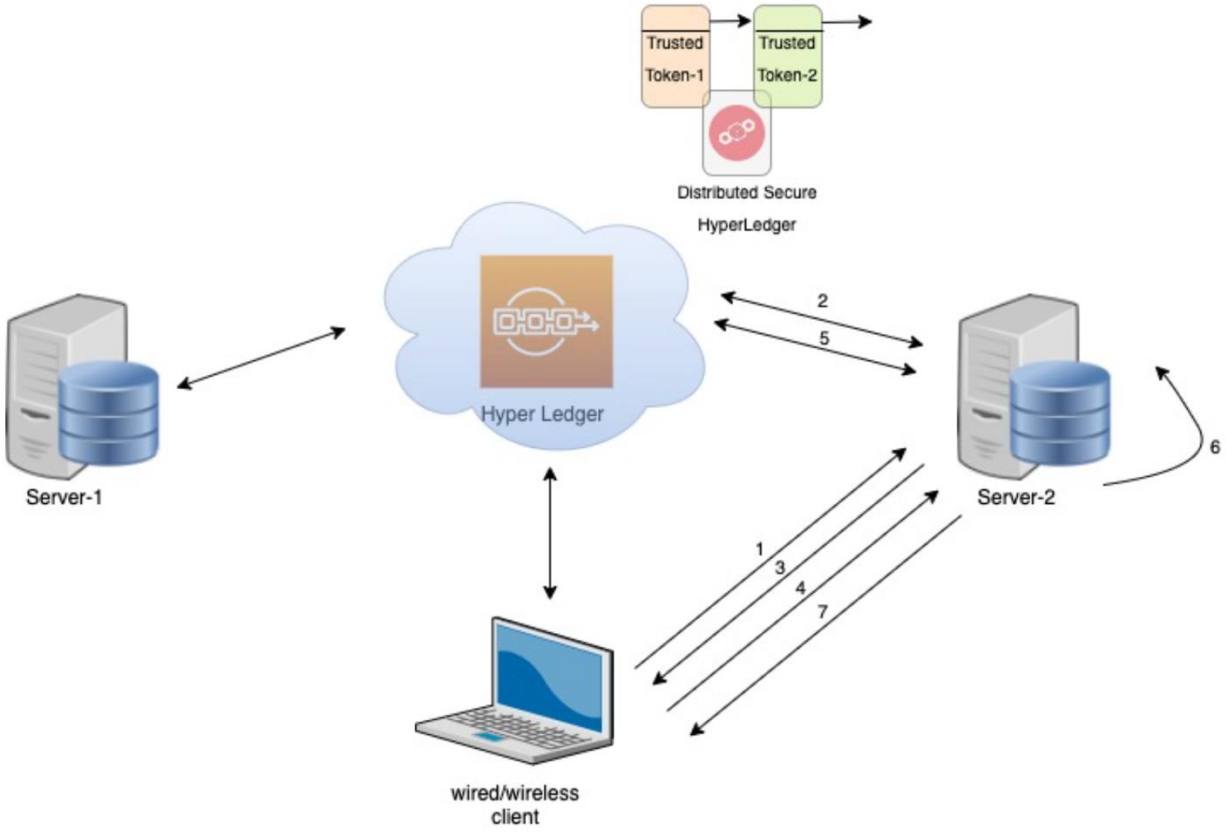

Figure-2

1. Wired/wireless client initiates SSO handshake (cookie, its own public-key and requesting public key of Server-2) with Server-2.
2. Server-2 verifies ownership of client's account via Hyper Ledger (permission blockchain)
3. Server-2 sends its public key to the Client.
4. Client validates the server, and it requests for Authenticating against the previous updated trusted token in the Hyper Ledger for client.
5. Server-2 retrieves details of Client's trusted token from the Hyper Ledger.\
6. Server-2 confirms that trusted token is associated with this Client provided by Server-1.
7. This complete SSO authentication.

Use of permission blockchain (Hyper Ledger), inherently provides security against reply attack, impersonate attack etc., as the record/block is immutable. Hyper Ledger is a private permissive blockchain and is used in multiple enterprise use cases. In this use case, using Hyper Ledger to share the signed trusted token of the client across Servers to provide MFA and SSO.

In this method, following four things are taken care to achieve the verifiable credentials (i.e., verify the credential without having to check with the issuer) and do more than that of paper credentials.

- Verify the Issuer (Server-1): The verifier (Server-2) has to know who issued the token (it is verifiable presentation and not the credential itself) based on the identifier and cryptographic signature (of Server-1). From the token, verifier (Server-2) gets an identifier of the issuer (Server-1). Verifier (Server-2) looks into the Hyper Ledger to get the public key associated with the issuer (Server-1) to verify the signature (of Server-1) of signed token. Thus the identity of the issuer (Server-1) is known.
- Verify that token is not altered: The verifier (Server-2) has to verify that the token has not been altered by verifying the cryptographic signature across the token. Based on the identifier (of Server-1) the verifier (Server-2) gets set of public keys from the Hyper Ledger and verifies the signatures. Thus, the verifier knows no one has tampered with the token.
- Verify that token is not revoked: The issuer (Server-1) of the client's token periodically updates client's token (based on session timeout or upon login failure etc.,) on the hyper ledger indicating that token has been revoked. If the client's token is revoked, then unable to create a proof of non-revocation. If the client device can generate the proof, the verifier can check it. Thus, the verifier knows that token has not been revoked.
- Verifying the client device: The binding of the token to the client device is done using privacy-preserving cryptographic magic (called a Zero Knowledge Proof) that prevents having a unique identifier for the client device. Thus, no Personally Identifiable Information (PII) is needed for sharing trusted data.

The technique presented herein remove dependencies on mobile (authenticator app, SMS, call) without compromising on security. This method removes single point of failure and less costly/complex SSO. Moreover, this method uses decentralised (distributed authentication

architecture) with access control. Additionally, this method is more trustworthy, as even Servers are attested (trustworthiness) by the blockchain provider before starting the service.