# Glove Technical Design

## Confidential Computing Overview

There are three components in an enclave-based architecture:

- The enclave
- The host
- The client(s)

The host is the VM/computer/operator within which the enclave is running in. The host has the ability to start and stop the enclave and also acts as the communication medium between the enclave and its remote clients. This is because enclaves run in a very restricted environment and typically do not have access to hardware such as the network and disk, and so rely on the host to relay messages back and forth.

Depending on the use case the host may be considered untrusted by the enclave and clients. It might be communication between the client and enclave must be fully confidential and the host considered a malicious observer. In such a scenario the encryption must be end-to-end between the enclave and client. This typically requires a custom communication channel and rules out TLS, since that would terminate the encryption at the host.

For other use cases, the host observing the communication may not be an issue but it must be prevented from manipulating or re-ordering messages.

Thus, the enclave can be thought of as a secure isolated environment within a broader untrusted host environment.

The enclave process should be as simple as possible, only performing those tasks that can only be done inside the enclave environment. Everything else should be offloaded to the host. This reduces the complexity of the code base and makes it easier to audit, which is a necessary component for establishing trust with the enclave. Less code also reduces the surface area for security vulnerabilities inside the enclave.

A key aspect of an enclave design in remote attestation. This is a document produced by the enclave, signed by the enclave vendor, which cryptographically proves to the client it's communicating with a genuine secure enclave. This is because the enclave vendor is the root of trust in a confidential computing system.

The client must verify and check the attestation document. Not doing so defeats the entire purpose of having a confidential computing architecture, and from a trust point of view, the client may as well be communicating with the host directly without an enclave.

## Requirements

For project Glove, the three components mentioned above would be:

- Enclave - running the Glove mixing protocol
- Host - Glove GovProxy operator running the Glove service
- Client - Kusama account voting on a particular referendum who has delegated to the Glove Proxy to anonymise their vote. Most likely the client will be represented as a user on the Glove web portal.

The Glove operator is trusted to not leak the account holders' real votes, and so messages between the client and enclave do not need to be encrypted. This simplifies the design as the encryption from the client can be terminated at the host and so a standard TLS connection is sufficient.

However, the client needs assurance the operator faithfully implements the Glove protocol and does not manipulate voting in anyway. This requires the enclave to produce signed confirmation messages which prove to the client the operator implemented Glove correctly ("Glove proofs").

💡 It's **not** a requirement that other network participants who are not using Glove be able to confirm as well.

The first implementation of Glove will use AWS Nitro Enclaves and this design will be based on that. However, in future Glove might expand to support other enclave technologies. To that end, where appropriate, the AWS Nitro implementation detail will be abstracted away.

# Architecture

## Keys

There are several cryptographic keys involved in the Glove service:

- KSM holder account key (″client key″) - Kusama account key for the KSM holder who has delegated their vote to the Glove GovProxy. This is owned by the KSM holder.
- Glove proxy account key (″proxy key″) - Kusama account key for the Glove GovProxy. This is owned by the Glove operator.
- Enclave mixer signing key (″enclave mixer key″) - For confirming on-chain votes as coming from Glove. This is owned by the enclave.
- Enclave image signing key (″enclave image key″) - For signing the enclave executable image. This is owned by the Glove operator.
- Enclave vendor attestation key (″attestation key″) - For verifying the enclave′s signed attestation document. This is owned by the enclave vendor and is well-known.

Both the proxy key and enclave image key are owned by the Glove operator. However, they have different lifecycles and usage. The proxy key is hot and will be used each time the operator needs to submit votes on-chain, whilst the image key will only used on the rare occasion to sign a new version of the enclave image. They should thus be stored in separate locations to reduce the impact of any security breach.

💡 The management of the enclave image key is out of scope of this design, but if it is stored in an AWS KMS instance, that instance must reside in a separate AWS account to the Glove service account.

## Components

The Glove architecture consists of the following main components:

- Web Frontend - this is the Glove web portal. It will need access to the KSM holder′s client key. It does the following:

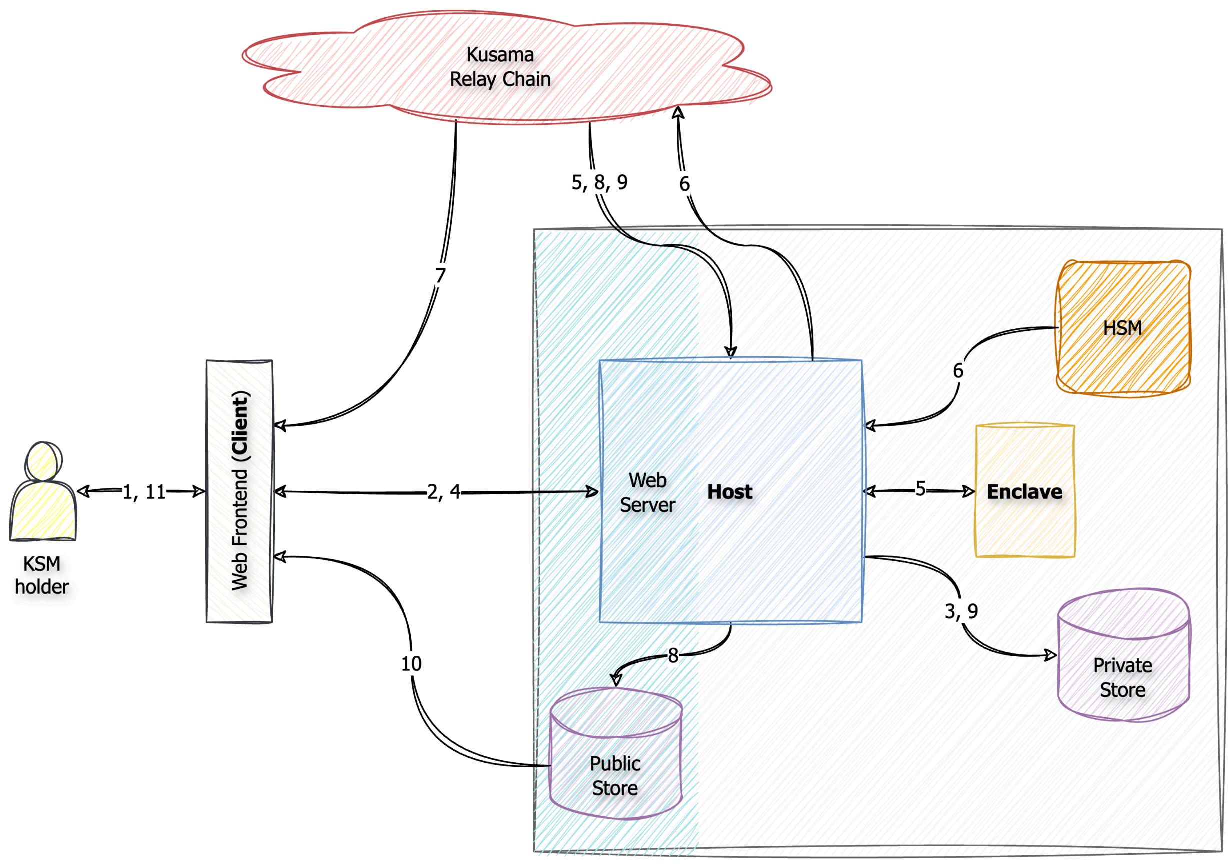
- (Optional) Assign the Glove GovPoxy as a proxy for the KSM holder. This may also be done outside of the Glove workflow
- Sign the voting request for a referendum as specified by the KSM holder with their client account key. This is to prevent tampering by the host
- As the enclave client, verifies the host has engaged the vote mixer enclave correctly by performing its own checks

- Host (AWS EC2) - this is the Glove operator's web service which runs the vote mixer enclave. It has the following responsibilities:
  - Run a web server to receive REST requests from the web frontend
  - Connect to the Kusama relay chain to receive updates on who has assigned it as their voting proxy
  - Run the vote mixer enclave and submit its voting results back to the relay chain
  - Listen for any Glove participants who decide to vote directly so that they can be removed from the Glove mixing.
- Enclave (AWS Nitro) - this is the vote mixer enclave which will execute the Glove mixing protocol on voting requests it receives from its clients via the host. Crucially, it will sign these voting events to prove to the client it was involved in the process.
- Private store (DB) - for storing in-progress votes to ensure high-availability. Each entry will represent a voting request by a KSM holder which hasn't been confirmed by Glove:
  - Request nonce
  - KSM holder account
  - Referendum ID
  - Information about their original vote to Glove
- Public store (AWS S3) - for storing signed proofs the Glove protocol was executed correctly. As the proofs are primarily for the benefit of the client, it is sufficient to store them in a centralised location off-chain.

- HSM (AWS KMS) - to store the Glove proxy's account key, which will be used by the host

## Voting workflow

The following diagram summaries the key steps in the Glove voting process:



1. After assigning the Glove as their voting proxy, the KSM holder enters in their voting direction, conviction and amount on the web frontend (client).
2. The client will generate a random `u128` nonce, attach it to the voting request, sign it with the client key, and send both the request and signature to the web server running on the host. The nonce is required to detect replay attacks if multiple voting requests are submitted and so the client is advised to keep this around until the referendum closes.
3. The host will store it to a private data store

4. The web server sends back a HTTP `OK` as confirmation the vote request was received. Notice, this does not mean the vote was mixed or submitted onto the chain. The client can update the UI to a “Vote acknowledged” state.
5. At certain moments, depending on Glove policy (details of which are out of scope of this design), the host will send all the voting requests for the referendum to the enclave to be mixed. The enclave will performing the mixing process as detailed below and produce a Glove proof.
6. The host will submit these instructions as proxied votes to the relay chain, using the proxy  key retrieved from the HSM.
7. The client’s assigned mixed vote will eventually appear on-chain. However, for the client this is not proof the Glove protocol was followed as the host could have submitted its own votes using the proxy key.
8. Once all the votes are confirmed on-chain, the host will persist the Glove proof and attestation document to the public store
9. Once the host is notified the referendum is closed, the voting request is deleted from the private store.
10. The client will download the Glove proof and attestation document and do the following:
    a. Confirm its latest vote is present by checking the nonce
    b. Make sure the proxy address and referendum ID are as expected
    c. Verify the signature is valid using the enclave mixer key
    d. Verify the mixer key came from a genuine Glove enclave by verifying the attestation (see Proof of Mixing for details).
11. If the client successfully verifies the proof, it updates the UI and informs the KSM holder details of the proxied vote.

## Glove proof generation

1. Verify the signature of each voting request using the client public key to ensure they’ve not been modified

2. Execute the Glove mixing algorithm on them as a single group. The result will be a list of randomised votes for each participant.
3. Create the following data structure and sign it with the mixer key:
    - Proxy account (this is implied from the attestation document)
    - Referendum ID
    - List containing each randomised vote:
        - KSM account
        - Assigned random vote
        - Request nonce

The enclave's current attestation document is appended to this signed data structure, and together this is called the Glove proof.

## Remixing

The client only verifies the Glove proof once the referendum has closed. This is because there are several scenarios where the requested votes need to be mixed again:

- The host detects an existing Glove participant has voted directly with their account. That participant's original voting request needs to be removed.
- If a new KSM holder joins the Glove process after the host has already submitted votes
- The KSM holder themselves changes their voting preference and submits another voting request
- As elaborated further below, the host suffers a failure and needs to submit all votes again to ensure nothing was left out

This may result in a client seeing their on-chain vote change with each re-mixing. However, until a valid Glove proof is published at the end, they are to be treated as transient.

## Proof of Mixing

The fundamental component of the architecture is the enclave mixer key and the signed proof it produces. The key is randomly generated inside the enclave and its private portion never leaves the enclave environment. Verifying the enclave's remote attestation document proves any on-chain vote which also has a matching Glove proof was mixed according to the Glove protocol.

The remote attestation document is generated on enclave boot up and has the following relevant components:

- Public portion of the enclave mixer key
- The Glove proxy address (i.e. the public portion of the proxy key)
- Enclave code measurement
- Public portion of the enclave image key

All of this is signed by the enclave vendor attestation key. As the enclave vendor is the root of trust, a valid signature proves the attestation document came from a real secure enclave.

This does not however prove the enclave is running the Glove protocol. For that one of two checks need to be made:

- The enclave code measurement is a hash of the code the enclave environment is executing. The Glove operator will need open source or make this code publicly available, along with instructions of how to generate this same code measurement. By auditing the code to confirm it's correctly implementing the Glove protocol, and confirming the generated code measurement is the same as the one in the attestation, proves the attestation document represents a Glove enclave.
- If auditing the code is not possible, the enclave image key in the attestation document can be checked against the key the Glove operator has made public. This proves the attestation came from an enclave built by the Glove operator, but assumes the Glove operator is trusted to implement Glove correctly.

Once it's confirmed the enclave is running Glove, the final check is to make sure the proxy address in the attestation is as expected.

💡 The AWS Nitro Enclave attestation document provides the code measurement (`PCR2`) and image key (`PCR8`) fields. For the mixer key and proxy address, they will be serialised into the `user_data` field.

## Failure Recovery

The host does not delete the KSM holder's voting request from the private store until the referendum has closed. If at any point the service goes down or is restarted, it can safely recover all the voting requests.

The first thing the host does on startup is load all the voting requests from the private store. Any voting requests for referenda which are now closed are deleted from the store, leaving only voting requests for open referenda.

There are several scenarios where the host will need to immediately execute the Glove algorithm for a referendum:

1. The Glove proof for the referendum is not present in the public store, but the conditions for executing the Glove algorithm are now met
2. The Glove proof for the referendum is present:
   a. There is a mismatch between the voting requests loaded from the private store and the assigned votes in the proof.
   b. There is no mismatch, but the conditions for executing the Glove algorithm are met again

## Threat Modelling

There are several attacks a malicious actor who gains access to the Glove system could perform to undermine Glove, in increasing order of severity:

1. Tricking participants into thinking on-chain votes from the Glove proxy were not in accordance to Glove when in fact they were
2. Manipulating the voting requests unnoticed, i.e. Glove participants believing the proxy is functioning correctly. This can broken down further:
   a. Removing voting requests

  b. Replaying old voting requests
  c. Changing voting requests. This is not possible as the requests are signed by the KSM holder.
  d. Adding a new voting request not from a Glove participant. This will always be detected by the Kusama network.
4. Revealing participants' private voting requests, both for current and past referenda
5. Tricking participants into thinking on-chain votes came from the Glove proxy when in fact they didn't

Which attacks are possible depend on the components compromised.

## Private Store

| Attack | Vulnerability | Mitigation |
|---|---|---|
| 2a | Voting requests can be deleted directly from the data store | Detected by Glove participant as their entry will not be in the Glove proof |
| 3 | No, since data stores such as AWS DynamoDB store all data encrypted | |

## Public Store

| Attack | Vulnerability | Mitigation |
|---|---|---|
| 1 | Glove proofs can be deleted from the store | Store the proof instead on-chain in a `remark` extrinsic |

## HSM

| Attack | Vulnerability | Mitigation |
|---|---|---|
| 2 | In the unlikely event of a compromise of the proxy account key, arbitrary voting | Detected by all Glove participants as there isn't a Glove proof. |

| Attack | Vulnerability | Mitigation |
|---|---|---|
| | from the proxy account can occur | As an extra layer of defensive, AWS KMS has a secure enclave release feature where secret material can only be released to a specific AWS Nitro enclave. |

## Host

| Attack | Vulnerability | Mitigation |
|---|---|---|
| 2a | Not passing in voting request to the enclave | Detected by Glove participant as their entry will not be in the Glove proof |
| 2b | Updated voting request can be dropped | Detected by Glove participant as their entry in the Glove proof will contain the old nonce |
| 3 | Yes, both from the web server after TLS termination, and from the private store after encryption at rest | Implement end-to-end encryption between the client and enclave |
| 4 | No, since the host cannot access the enclave mixer key to forge Glove proofs | |

## Enclave

In the event of the mixer key being leaked out of the enclave.

If there is a breach of the enclave then the attacker gets access to the mixer key and are able to forge fake Glove proofs. This would only affect open referenda as the attacker can “mix” votes as they please. Past voting requests for closed referenda would be unaffected as they are not stored anywhere.

## Client

The client may try to disrupt the Glove process for other Glove participants by first voting with Glove and then directly. Though the host will be looking for this behaviour, and will kick out the offending client, the malicious client might try to do this near the end of the referendum and leave little time for the host to take corrective action.

## Further Improvements

As highlighted above, end-to-end encryption between the client and enclave removes the impact of a security breach on the host where the private voting requests could then be leaked. It also has an added benefit of removing the need to trust the Glove operator, which may increase Glove adoption.

Storing the Glove proofs on-chain in `remark` extrinsic would remove the need for the public store.