This is a work-in-progress draft by Shunfan Zhou and Hang Yin. We welcome comments and experimentation with the idea.

We would like to give special thanks to Prof. Andrew Miller for proposing the use of the Certificate Transparency mechanism to monitor configuration changes in centralized DNS servers.

Previous Articles

In [Early Thoughts on Decentralized Root-of-Trust

](https://collective.flashbots.net/t/early-thoughts-on-decentralized-root-of-trust/3868), we described our motivation to update the TEE Root-of-Trust from hardware-kept unrecoverable secret keys to a smart-contract-governed software (i.e., DeRoT).

In [Key Management Protocol For Decentralized Root-of-Trust

](https://collective.flashbots.net/t/key-management-protocol-for-decentralized-root-of-trust/4004) , we delve deeper into the details of its key management protocol, bringing support for upgradable and migratable TEE applications, and advanced security features like key rotation.

The Problem

In this article, we answer a practical question: How to ensure that the certificate for a specific domain is controlled and can only be controlled by a TEE program (in our case, the DeRoT).

Once we can achieve this, DeRoT, as the root of trust, can then sign more certificates for every application it has verified. This will be an important building block to enable users to seamlessly verify whether an application is actually running within valid TEEs using the existing TLS-based protocols (like HTTPS). The DeRoT-controlled certificate can not only prove its identity under this paradigm, but extend the chain of trust from the blockchain ecosystem to any website.

Given the key management protocol mentioned above, the DeRoT becomes the root-of-trust of the whole system as it is responsible for verifying all the TEE hardware and managing the data encryption. Its identity can be verified with the RootPubKey

published on-chain.

If the DeRoT can use this RootKey

as the certificate's private key, then it owns a TLS certificate which is verifiable on chain and can issue certificates to valid TEE applications running in verified hardware. Since the DeRoT is running inside TEE, it is not hard to generate a certificate that never leaves TEE. The real challenge here is how to ensure this certificate is tightly bound with a domain and cannot be replaced, given that the DNS servers are centralized services that cannot be controlled by smart contract or TEE application.

Building Blocks

Here we introduce two extra building blocks:

- Certification Authority Authorization (CAA) in DNS record. This is used to specify which certificate authorities (CAs) are permitted to issue certificates for their domain. It prevents unauthorized issuance of SSL/TLS certificates. With CAA, we can ensure that the DeRoT-owned certificate is the only valid one for its domain.
- Certificate Transparency (CT) is an Internet security standard for monitoring and auditing the issuance of digital certificates. Although the CAA ensures the DeRoT ownership over the certificate, the CAA itself, which can be configured in centralized service providers, may be changed. The CT logs can be used for two purposes: 1) to verify whether the CAA is really set to the expected certificate, and 2) to enable users to detect and reject unauthorized CAA modifications.

System Design

Here we introduce the certificate managing module in the DeRoT serving as a reverse proxy for applications running in the network. It's worth noting that as a part of DeRoT, it runs in TEE and is also governed by the smart contract.

[

image

1278×688 53.3 KB

](https://collective.flashbots.net/uploads/default/original/2X/1/16718207a1f0825fe1d8cfd650deab4bc0b19839.png)

DeRoT Certificate Registration

The DeRoT will generate a x.509 certificate using the RootKey

as the private key. Compared with other TLS certificates, the only difference is that the RootPubKey

is available on-chain, so any users can verify that this certificate is really owned by DeRoT.

This certificate is sent to CA (e.g. Let's Encrypt) for signing, then a DNS CAA record is registered. In the example above, the CAA record means "only letsencrypt.org is allowed to issue certificates forderot.com".

We further utilize CAA records to restrict the authentication methods, as some of them are not secure. For example, when verifying with HTTPS, Let's Encrypt only requires you to place a specific text file at http:///verify.txt

to confirm domain ownership. However, Let's Encrypt cannot verify whether the endpoint is within a TEE. To address this, we limit the authentication method to API access only, which can be fully controlled by a TEE. This ensures that there is no possibility of issuing the certificate to an incorrect target.

TEE App Registration

Although a senior user can request to verify the Remote Attestation report from a TEE app directly, most of the users could delegate the verification to DeRoT by verifying its certificate.

To get reverse proxy from DeRoT, a TEE app needs to register itself to DeRoT, during which DeRoT will do the RA verification on it to ensure the DeRoT policy is followed.

Protection against Unauthorized Certificate Changes

Since the DNS servers are centralized services which cannot be controlled by smart contracts, there is risk that the attackers will change the DNS record to break the limitation on the certificate owner.

Although we cannot technically prevent this, we can monitor the unexpected DNS record updates and notify the clients to avoid trusting invalid certificates.

Implementation

- An implementation runnable on Intel TDXGitHub - Phala-Network/dstack with
- Docker image support
- Basic key management and certificate managing implementation
- Docker image support
- Basic key management and certificate managing implementation