Bool Network: An Open, Decentralized,Secure Bitcoin Verification Layer

Bool Network Foundation

***Abstract*—Bitcoin's slow transaction throughput has long been a concern for its users. While Bitcoin's security and decentralization have always been its strengths, its limited transaction speed has hampered its ability to efficiently process large volumes of transactions. This is also the blockchain impossible triangle problem, how to solve this problem?The main solution is through layer2.Since Bitcoin has no smart contracts, there is no way to verify on the chain, the current layer2 solutions are all centralized.To resolve this issue, we present Bool Network –an open, decentralized and secure Bitcoin verification layer powered by MPC-based distributed key management over evolving hidden committees. More specifically, to protect the identities of the committee members, we propose a Ring verifiable random function (Ring VRF) protocol, where the real public key of a VRF instance can be hidden among a ring, which may be of independent interest to other cryptographic protocols. Furthermore, all the key management procedures are executed in the TEE, such as Intel SGX, to ensure the privacy and integrity of partial key components. A prototype of the proposed Bool Network is implemented in Rust language.**

***Index Terms*—bitcoin,layer2, evolving committee, threshold cryptography, ring VRF, trusted execution environment**

I. INTRODUCTION

Starting this year, the entire Bitcoin network has been inundated with the proliferation of the Bitcoin ecosystem, from NFTs to BRC-20 tokens. As the number of participants continues to grow, the Bitcoin network has become more congested, leading to higher costs. Addressing on-chain congestion and reducing gas fees has become a pressing demand for users. Discussions about Bitcoin scalability have once again gained momentum. To address this problem, many layer2 solutions have been proposed.In order to inherit the security of Layer1 blockchains, Layer2 blockchains must be capable of performing validation computations on the Layer1 blockchain. However, since Bitcoin lacks smart contract functionality, such validation computations are not possible. As a result, most current Layer2 solutions are unable to inherit the security of Bitcoin, leading to concerns among users about transferring their Bitcoin assets to Layer2 networks for transactions.

Our approach. In this work, we present Bool Network an open, decentralized and secure Bitcoin verification layer.It makes Layer2 to retain the advantages of the solution, e.g., excellent user-convenience and great scalability, while making Layer2 more decentralized and secure. More specifically, Bool Network is an open blockchain where hidden committees elected by cryptographic sortition are incentivized to handle the verification operations. The committee members can anonymously communicate over blockchain, while their identities are protected from the public. Initially, the committee members collaboratively create an account on each supported blockchain via multi-party computation (MPC). To transfer a token from Bitcoin to Layer2 chain, the user first transfers the token to Bool Network's account on Bitcoin, and sends the transfer request to the Bool Network. The committee then jointly sign (by threshold signature scheme) a transaction on Layer2 chain that issues the representative token to the user's account on Layer2 chain. The system structure of a committee-based scheme is shown in Fig. 1.

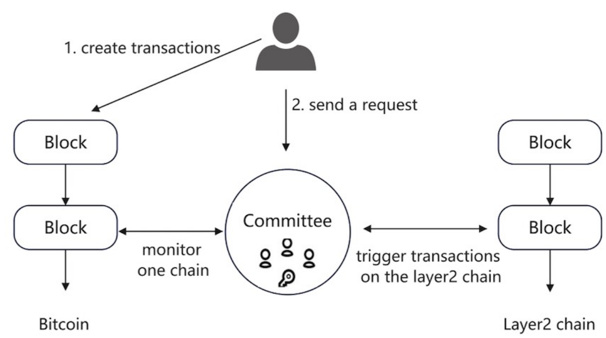

Fig. 1: System structure of a committee-based scheme

Using Bool Network's dynamic committee verification capability, the Bitcoin ledger data and the Layer2 ledger data can be synchronized to the bool chain. Users can withdraw assets from Bitcoin to the Layer2 for operations, ensuring that, regardless of any circumstances on the Layer2, assets can be safely retrieved to the corresponding user account on the Bitcoin. This way, any chain can become a Layer2 for Bitcoin, enabling unlimited scalability for Bitcoin.The system structure of the verification layer for Bitcoin is shown in Fig. 2.

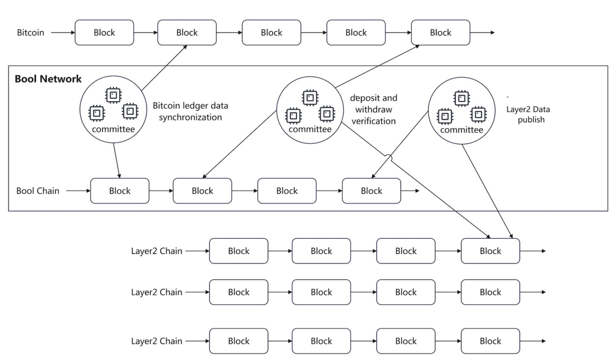

Fig. 2: The system structure of the verification layer for Bitcoin

To achieve evolving and hidden committee, Benhamouda *et al.* [3] proposed a complicated two-committee scheme, where a public nominating committee is first selected by cryptographic sortition, and then each nominating committee members will randomly select a holding committee member. Alternatively, we present a much more efficient solution based on a new notion, called *Ring VRF* (R-VRF). In Ring VRF, the real public key of a verifiable random function (VRF) instance can be hidden among a ring. More specifically, we design a non-interactive zero knowledge protocol that allow the prover to show that he/she knows the sk corresponding to one of the public keys among the ring and the output of the pseudorandom function (PRF) $\mathsf{PRF}_{\mathsf{sk}}(x)$ is correct. In particular, showing $\mathsf{PRF}_{\mathsf{sk}}(x) = v$ w.r.t. the ring $(\mathsf{pk}_0, \ldots, \mathsf{pk}_{N-1})$, we have the following relation

$$\mathcal{R}_{\mathsf{VRF}} = \left\{ \begin{array}{l} ((\mathsf{pk}_0, \ldots, \mathsf{pk}_{N-1}, x, v), \mathsf{sk}, \ell) \mid \\ \ell \in \{0, \ldots, N-1\} \ \wedge \ (\mathsf{pk}_\ell, \mathsf{sk}) \in \mathcal{R}_{\mathsf{pk}} \ \wedge \\ v \leftarrow \mathsf{PRF}_{\mathsf{sk}}(x) \end{array} \right\}$$

Remark: Furthermore, to avoid identity leakage from the network-layer, only an anonymous broadcast channel is not enough because the adversary can leverage network delays to identify who is the committee member [15]. A sanitization protocol on the application layer such as SABRE [1], or reliable broadcast mechanisms [6], [7] can be utilized to address this problem, as suggested in [15]. However, this is out of scope of this paper.

In addition, to achieve minimal malware attack interface, we deploy all the key-related programs on a trusted execution environment (TEE), such as Intel SGX, to make sure that even a malicious committee member cannot extract his/her secret shares. Moreover, TEE enforces key share erasure after committee switching. Note that, TEEs can be heterogeneous; namely, the TEEs of the MPC participants can be from different manufactories. Therefore, the threshold cryptosystem used in our scheme ensures that even if side channel attacks against some TEE chips are possible, the adversary cannot reconstruct the secret as long as the majority of TEEs remain secure.

II. Preliminaries

A. Hybrid Encryption

Our scheme utilizes hybrid encryption over blockchain for peer-to-peer communication. The hybrid encryption scheme HEnc consists of the following 4 PPT algorithms.

- $\sigma \leftarrow \mathsf{Setup}(1^\lambda)$: take input as security parameter $\lambda \in \mathbb{N}$, and output a group parameter σ.
- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\sigma)$: pick random $s \leftarrow \mathbb{Z}_q^*$ and set $h := g^s$, and output $(\mathsf{pk} := (g, h), \mathsf{sk} := s)$.
- $C \leftarrow \mathsf{Enc}_{\mathsf{pk}}(\sigma; m)$: pick random $r \leftarrow \mathbb{Z}_q$; compute $c_1 := g^r$ and $c_2 := h^r$; set $k \leftarrow \mathsf{hash}(c_2)$; compute $u := \mathsf{AES\text{-}GCM}_k(m)$; output $C = (c_1, u)$.
- $m \leftarrow \mathsf{Dec}_{\mathsf{sk}}(\sigma; C)$: compute $c_2 := (c_1)^{\mathsf{sk}}$; set $k \leftarrow \mathsf{hash}(c_2)$; output $m := \mathsf{AES\text{-}GCM}_k^{-1}(u)$.

Clearly, the above hybrid encryption scheme HEnc is IND-CPA secure under the DDH assumption and the semantic security of AES-GCM encryption mode. In the actual implementation, hybrid encryption requires a trusted curve group parameter, and we adopt the well-known curve 25519.

B. Commitment

A commitment scheme allows a sender to commit to a value, and later he can reveal the value by opening the commitment. A secure commitment scheme shall have two properties: hiding and binding. We denote a commitment as $c = \mathsf{Com}(m; r)$, where m is the committed value(message) and r is the randomness. The hiding property and binding property are defined as follows:

- Hiding: for any PPT adversary $\mathcal{A}$, $\Pr[(m_0, m_1) \leftarrow \mathcal{A}; b \leftarrow \{0,1\}; c \leftarrow \mathsf{Com}(m_b) : \mathcal{A}(c) = b] \approx \frac{1}{2}$
- Binding: for any PPT adversary $\mathcal{A}$, the proability $\Pr[(m_0, r_0, m_1, r_1) \leftarrow \mathcal{A} : m_0 \neq m_1, \mathsf{Com}(m_0; r_0) = \mathsf{Com}(m_1, r_1)] \approx 0$

A Petersen commitment [20] is defined as follows. $c := \mathsf{Com}_{\mathsf{ck}}(m; r) := g^m h^r$, where $\mathsf{ck} = h$ is the commitment key whose discrete logarithm is unknown to the committer. Pedersen commitment is additively homomorphic; namely, $\mathsf{Com}_{\mathsf{ck}}(m_0; r_0) \cdot \mathsf{Com}_{\mathsf{ck}}(m_1; r_1) = \mathsf{Com}_{\mathsf{ck}}(m_0 + m_1; r_0 + r_1)$.

In some use case, we don't need homomorphic property. We simple use the salted hash based commitment defined as $c := \mathsf{Com}(m; r) = \mathsf{hash}(m, r)$.

C. Blockchain

Our system is built on top of a blockchain platform. We assume parties can use the underlying blockchain as a non-blocking broadcast channel. Namely, if party P_1 posts a message m_1 over the blockchain at the i-th round, any other parties will receive it no later than $(i + \delta)$-round, where δ is a known bound. For recording wealth, the blockchain also serves as an account model ledger so that stake holders can lock some amount of tokens to apply for a committee member. Formally, we abstract the blockchain platform as a UC functionality $\mathcal{G}_{\mathsf{blockchain}}$ in Fig. 5 in Sec. III, below.

D. Verifiable Random Functions and Cryptographic Sortition

A verifiable random function (VRF) [18] is a pseudorandom function that enables the key holder to prove the correctness of the output. A VRF scheme VRF consists of the following 3 PPT algorithms:

- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\lambda)$: The key generation algorithm produces a public key pk and a secret key sk.
- $(v, \pi) \leftarrow \mathsf{Eval}_{\mathsf{sk}}(x)$: The evaluation algorithm takes secret key sk and pre-image x, and image v and a proof π.
- $0/1 \leftarrow \mathsf{Verify}_{\mathsf{pk}}(x, v, \pi)$: The verifier algorithm takes input as the public key pk, the pre-image x, the image v, and the proof π, and it outputs 0 for rejection or 1 for acceptance.

Besides pseudo-randomness, a secure VRF scheme should also have the following properties.

- Completeness: for any x, given $(v, \pi) \leftarrow \mathsf{Eval}_{\mathsf{sk}}(x)$, we have $\mathsf{Verify}_{\mathsf{pk}}(x, v, \pi) = 1$.
- Uniqueness: no PPT adversary $\mathcal{A}$ can output a public key pk, a pre-image x, and (v_1, π_1, v_2, π_2) such that $v_1 \neq v_2$ and $\mathsf{Verify}_{\mathsf{pk}}(x, v_1, \pi_1) = \mathsf{Verify}_{\mathsf{pk}}(x, v_2, \pi_2) = 1$.

A formal definition can be found in [18].

It is well-known that cryptographic sortition [10] can be realized by VRF schemes. Over the blockchain, suppose $(\mathsf{pk}_i, \mathsf{sk}_i)$ are associated with party P_i. For a public input x, each party P_i can compute $(v_i, \pi_i) \leftarrow \mathsf{VRF.Eval}_{\mathsf{sk}_i}(x)$. Then, v_i can be used to select the committee with public verification. In our system, we would like to further hide the committee member's identities by proposing a new notion called Ring VRF (cf. Sec. IV-A, below).

E. Trusted Execution Environment

Trusted execution environment (TEE) is designed to guarantee confidentiality and integrity of computations. It is an isolated part that can store sensitive data and can issue attestation to prove correctness of computation. In practice, Intel SGX and ARM Trustzone are popular candidates of TEE. Although a few side-channel attacks, e.g. [25], [16], [24], have been explored against those TEE candidates, new designs and fixes are proposed on a monthly basis [5], [21], [17], [19], [23]. Hence, we take TEE as an acceptable hardware guard to ensure privacy of secret key components. In this work, our benchmarks are executed on the Intel SGX platform for its readily deployed remote attestation infrastructure; however, we emphasize that our protocol can also be implemented on any other TEE platforms.

III. System Overview and Security Model

A. System Architecture

In this section, we will give an overview of Bool Network platform with the example of cross-chain transfer between Bitcoin (BTC) and Ethereum (ETH). As depicted in Fig. 3, (C_1, C_2) denotes the cross-chain channel creation procedures; (A_1, A_2, A_3) denotes the lock-in procedures; (B_1, B_2, B_3) denotes the lock-out procedures.

Initially, a user creates an Bool Network account on the Bitcoin blockchain through the Bool Network committee (cf. C_1) and a smart contract anchored to the Bool Network account on Ethereum (cf. C_2); subsequently, the Bool Network committee members will sign a test message to check the channel's availability.

Each committee member runs light nodes of BTC and ETH to verify transactions. When the user wants to lock-in 1 BTC, he makes a transfer of 1 BTC to the address that was created in step C_1 (cf. A_1). After confirming the transaction, the Bool Network committee will use threshold signature to sign a transaction that creates 1 wBTC on Ethereum (cf. A_2). Finally, the signed transaction will be submitted to Ethereum (cf. A_3). The lock-out process is similar: first, the user destroys 1 wBTC on the Ethereum Anchor contract and specify the beneficiary address (cf. B_1). Then, the committee will monitor the event and generate the corresponding Bitcoin transaction (cf. B_2). Lastly, the transaction will be broadcast on Bitcoin network (cf. B_3).

Bool Network supports evolving hidden committee. We assume the underlying blockchain has a weak beacon oracle that can periodically produce an unpredictable but presumably biased random string. For instance, the hash digest of latest blocks. Define an epoch as a pre-defined number of rounds. At the beginning of each epoch, a new committee will be selected by cryptographic sortition using Ring VRF. The old committee will then handover the secret keys to the new committee in a secure and verifiable way. Fig. 4 shows an overview of the evolving hidden committee. More specifically, it consists of the following phases.

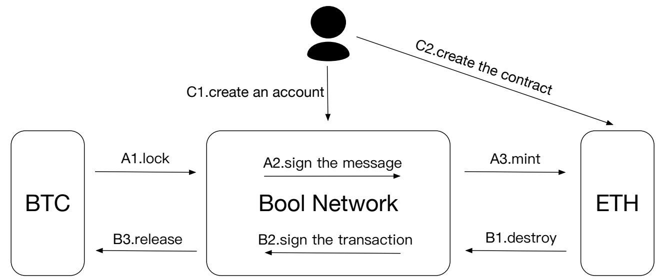

Fig. 3: Bool Network overview

1. Registration. Before an epoch starts, all Bool Network stakeholders can register for becoming an committee member. The registration phase works as follows. A stakeholder locks a certain amount of stake together with a public key. Later, the associated public key will be used for cryptographic sortition. For the sake of fairness and animosity, our system utilizes the flat mode, i.e., there is a fixed amount of stake to be locked per registration. If a stakeholder holds more stake, he/she can register several times to get a larger winning probability.
2. Sortition. For committee sortition, everyone first generates the weak beacon string x over Bool Network blockchain, which can be the hash of the previous blocks. Then, each stakeholder takes x as input and computes $\mathsf{PRF}_{\mathsf{sk}}(x)$. If $\mathsf{PRF}_{\mathsf{sk}_i}(x) = v < T$, then P_i randomly selects a ring $(\mathsf{pk}_1, \ldots, \mathsf{pk}_\ell)$ and posts message $(\mathsf{pk}_1, \ldots, \mathsf{pk}_\ell, v, \pi, \mathsf{epk})$ over the blockchain, where π is the Ring VRF proof, v is the PRF output, epk is a freshly generated ephemeral public key. Winning nodes are elected for next-epoch committee. From then on, they can be communicated with, using the ephemeral public key epk. As we can see, the ephemeral public key reveals nothing about the committee member's identity, and the real identity is hidden within the ring $(\mathsf{pk}_1, \ldots, \mathsf{pk}_\ell)$.
3. Handover. When a new committee is elected, the shared secret keys need to be passed from the previous-epoch committee to the new committee while keeping the public keys unchanged. Our protocol is maliciously secure with identifiable aborts. See sec. IV-E for details.
4. Threshold Signature. All transactions in our Bool Network shall be jointly signed by the committee using a threshold signature scheme. In particular, the threshold signature scheme for ECDSA [14] is adopted from the GG20 scheme by Gennaro and Godfeder [12] where the malicious party can be identified in a protocol abortion.

B. Key Management

As we mentioned above, all the key-related programs are deployed on a trusted execution environment (TEE). Specifically, on the initialization stage, a committee member loads

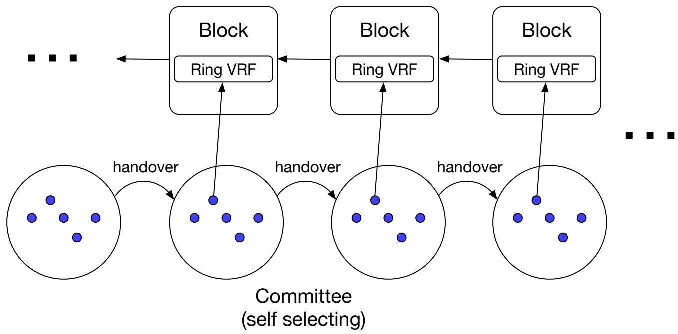

Fig. 4: Evolving committee overview

Functionality $\mathcal{G}_{\mathsf{blockchain}}$

The ideal functionality $\mathcal{G}_{\mathsf{blockchain}}$ is globally available to all participants. It is parameterized with a predicate $\mathsf{Validate}$.

Initialization:

- Upon initialization, set $\mathsf{Storage} := \emptyset$.

Storage:

- Upon receiving $(\textsc{Read}, \mathsf{sid})$ from P:
  - let $\mathsf{val} := \mathsf{Storage}[\mathsf{sid}]$;
  - return $(\mathrm{READ}, \mathsf{sid}, \mathsf{val})$ to the requestor.
- Upon receiving $(\textsc{Write}, \mathsf{sid}, \mathsf{inp})$ from P, do the following:
  - let $\mathsf{val} := \mathsf{Storage}[\mathsf{sid}]$;
  - if $\mathsf{Validate}(\mathsf{val}, \mathsf{inp}) = 1$, then set $\mathsf{Storage}[\mathsf{sid}] := \mathsf{val}||(\mathsf{inp}, P)$, return $(\textsc{Receipt}, \mathsf{sid})$ to the requestor;
  - Otherwise, return $(\textsc{Reject}, \mathsf{sid})$ to the requestor.

Check:

- Upon receiving $(\textsc{Check}, \mathsf{sid}, \mathsf{val})$, if $\mathsf{val} \in \mathsf{Storage}[\mathsf{sid}]$ then return $(\textsc{Check}, \mathsf{sid}, \mathsf{true})$; else return $(\textsc{Check}, \mathsf{sid}, \mathsf{false})$ to the requestor.

Fig. 5: The ideal blockchain functionality $\mathcal{G}_{\mathsf{blockchain}}$

Functionality $\mathcal{F}^{t,n}_{\mathsf{DKG}}[\mathbb{G}]$

The ideal functionality $\mathcal{F}^{t,n}_{\mathsf{DKG}}[\mathbb{G}]$ interacts with key generators $\mathcal{P} := \{P_1, \ldots, P_n\}$, an ideal adversary $\mathcal{S}$. It's parameterized with threshold t. Denote $\mathcal{P}_c$ as the set of corrupted generators, $\mathcal{P}_h := \mathcal{P} \setminus \mathcal{P}_c$ as the set of honest generators, and $|\mathcal{P}_c| \leq t-1$. $\mathcal{F}_{\mathsf{DKG}}$ maintains a set $\mathcal{N}$(initially set to $\emptyset$).

- Upon receiving $(\textsc{KeyGen}, \mathsf{sid}, P_i)$ from $P_i \in \mathcal{P}$, set $\mathcal{N} := \mathcal{N} \cup P_i$, send $(\textsc{KeyGenNotify}, \mathsf{sid}, P_i)$ to $\mathcal{S}$. Wait until $|\mathcal{N}| = n$.
- Upon receiving $(\textsc{CorruptShares}, \mathsf{sid}, (\{j, \mathsf{psk}_j\}_{P_j \in P_c}))$ from $\mathcal{S}$:
  - Pick $\mathsf{gsk} \leftarrow \mathbb{Z}_q$, and compute $\mathsf{gpk} := g^{\mathsf{gsk}}$;
  - Construct random polynomial $F(x) := \sum_{b=0}^{t-1} a_b \cdot x^b$ under the restriction $F(j) = \mathsf{psk}_j$ for $P_j \in \mathcal{P}_c$, and $F(0) = \mathsf{gsk}$;
  - Compute $\mathsf{psk}_i := F(i)$ and $\mathsf{ppk}_i = g^{\mathsf{psk}_i}$ for $i \in [n]$;
  - Send $(\textsc{Abort}, \mathsf{sid})$ to $\mathcal{S}$ and wait for an answer$(\textsc{Abort}, \mathsf{sid}, b)$;
  - Upon receiving the answer, if $b = 1$ then halt;
  - Otherwise, sand $(\textsc{KeyGen}, \mathsf{sid}, (\{\mathsf{ppk}_j\}_{j \in [n]}))$ to $\mathcal{S}$ and send $(\textsc{KeyGen}, \mathsf{sid}, (\mathsf{psk}_i, \{\mathsf{ppk}_j\}_{j \in [n]}))$ to $P_i, i \in [n]$.
- Upon receiving $(\textsc{ReadPk}, \mathsf{sid})$ from any party, return $(\textsc{ReadPk}, \mathsf{sid}, \mathsf{gpk})$ to the requestor.

Fig. 6: DKG ideal functionality $\mathcal{F}^{t,n}_{\mathsf{DKG}}[\mathbb{G}]$

the program $\Pi_{\mathsf{DKG}}, \Pi_{\mathsf{handover}}$ and Π_{sign} into his/her TEE and generates an remote attestation σ which proves that the initialization process is correct. He/she then sends σ to the attestation service and gets the attestation verification report π. Finally, the attestation verification report π is put on the blockchain to be publicly verifiable. Note that merely using cryptographic tools cannot guarantee a malicious committee would faithfully delete his/her secret shares upon request. The deployment of TEE prevents the secret shares from being extracted and enforces key share erasure after the handover protocol, providing better confidentiality of secret keys.

C. Universal Composability

We will do security analysis under the Universally Composable (UC) [8], [9] model. In the UC framework, a protocol is represented by a set of interactive Turing machines (ITMs), and each ITM represents the program to be run by a participant. In the following, we assume that all ITMs are probabilistic polynomial time (PPT).

The security proof is based on the indistinguishability between the real/hybrid world and the ideal world. Let $\mathsf{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}$ be the execution of protocol Π in the real world with adversary $\mathcal{A}$ and environment $\mathcal{Z}$; let $\mathsf{EXEC}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$ be the execution in the ideal world interacting with ideal functionality $\mathcal{F}$, simulator (ideal adversary) $\mathcal{S}$ and environment $\mathcal{Z}$. We say that Π UC-realizes $\mathcal{F}$ if for any PPT adversary $\mathcal{A}$, there exists a simulator $\mathcal{S}$ such that no PPT environment $\mathcal{Z}$ can distinguish between $\mathsf{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}$ and $\mathsf{EXEC}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$.

D. UC Ideal Functionalities

Our system utilizes the following UC functionalities as building blocks.

Blockchain. As shown in Fig. 5, the functionality $\mathcal{G}_{\mathsf{blockchain}}$ consists of three interfaces: READ, WRITE and CHECK. Both broadcasting messages and modification of the balance are through the WRITE interface. There is a predicate Validate that ensures the validity of the blockchain content.

Distributed Key Generation. We propose a UC secure distributed key generation (DKG) protocol with identifiable aborts in Sec. 11. The DKG functionality is captured by $\mathcal{F}_{\mathsf{DKG}}[\mathbb{G}]$. As shown in Fig. 6, the functionality chooses a random global secret gsk and shares it to the participants using Shamir's secret sharing. The adversary $\mathcal{S}$ is able to determine corrupted parties' secret shares, as, in the rushing adversary model, the corrupted parties can send messages after seeing the honest parties' messages in the real protocol.

Anonymous Cryptographic Sortition. The anonymous sortition functionality $\mathcal{F}^{p}_{\mathsf{Anon\text{-}sortition}}$ is an abstraction that captures our Ring VRF-based sortition protocol, as depicted in Fig. 7. Unlike the conventional cryptographic sortition, when an elected committee member reveal his/her winning the sortition, he/she does not publish his/her identity. Instead, he/she can publish a message m_i through the REVEAL interface, which is an ephemeral public key in our scheme.

Functionality $\mathcal{F}^{p}_{\mathsf{Anon\text{-}sortition}}$

The functionality $\mathcal{F}^{p}_{\mathsf{Anon\text{-}sortition}}$ interacts with a set of parties $\mathcal{P} := \{P_1, \ldots, P_n\}$ and adversary $\mathcal{S}$. It is parameterized with a variable $p \in [0,1]$, and tables Pub and T. Let $\mathcal{P}_c$ be the set of corrupted parties, $\mathcal{P}_h := \mathcal{P} \setminus \mathcal{P}_c$ be the honest parties.
Initially, set $\mathsf{Pub} := \emptyset$, $T := \emptyset$.

Initialization:

- Upon receiving $(\textsc{Init}, \mathsf{sid})$ from the adversary $\mathcal{S}$, for each party P_i, pick a random bit $b^{\mathsf{sid}}_{P_i}$ with $\Pr[b^{\mathsf{sid}}_{P_i}] = p$ and set $T[\mathsf{sid}] := \{< P_i, b^{\mathsf{sid}}_{P_i} >\}_{P_i \in \mathcal{P}_c}$, and then send $(\textsc{Init}, \mathsf{sid}, T[\mathsf{sid}])$ to the adversary $\mathcal{S}$.
- Upon receiving $(\textsc{Refresh}, \mathsf{sid})$ from $\mathcal{S}$, last step will be repeated.

Query:

- Upon receiving $(\textsc{Query}, \mathsf{sid})$ from $P_i \in \mathcal{P}$, if $b^{\mathsf{sid}}_{P_i}$ is initialized, send $(\textsc{Query}, \mathsf{sid}, b^{\mathsf{sid}}_{P_i})$ to the requestor.

Reveal:

- Upon receiving $(\textsc{Reveal}, \mathsf{sid}, m_i)$ from $P_i \in \mathcal{P}$:
  - If $b^{\mathsf{sid}}_{P_i} = 1$, then set $\mathsf{Pub}[\mathsf{sid}] := \mathsf{Pub}[\mathsf{sid}] \cup \{m_i\}$. Send $(\textsc{Reveal}, \mathsf{sid}, m_i)$ to the adversary $\mathcal{S}$.
  - Otherwise, ignore the request.
- Upon receiving $(\textsc{State}, \mathsf{sid})$ from any party $P_i \in \mathcal{P}$, send $(\textsc{State}, \mathsf{sid}, \mathsf{Pub}[\mathsf{sid}])$ to the requestor.

Fig. 7: Anonymous cryptographic sortition ideal functionality $\mathcal{F}^{p}_{\mathsf{Anon\text{-}sortition}}$

Handover. We design a UC secure handover protocol that allows the old committee to pass the shared secret keys to the new committee. Its functionality is captured by $\mathcal{F}_{\mathsf{handover}}[\mathbb{G}]$ as depicted in Fig. 8, the adversary can determine corrupted parties' secret shares.

Functionality $\mathcal{F}^{t_1,t_2,c_1,c_2}_{\mathsf{handover}}[\mathbb{G}]$

$\mathcal{F}^{t_1,t_2,c_1,c_2}_{\mathsf{handover}}[\mathbb{G}]$ interacts with previous-epoch committee members $\mathcal{P} := \{P_1, \ldots, P_{c_1}\}$, next-epoch committee members $\mathcal{Q} := \{Q_1, \ldots, Q_{c_2}\}$ and adversary $\mathcal{S}$. It is parameterized with threshold t_1, t_2, previous-epoch committee size c_1, next-epoch committee size c_2. Denote $\mathcal{P}_c$ as the set of previous-epoch corrupted parties, $\mathcal{P}_h := \mathcal{P} \setminus \mathcal{P}_c$ as the set of previous-epoch honest parties, and $|\mathcal{P}_c| \leq t_1 - 1$. Denote $\mathcal{Q}_c$ as the set of next-epoch corrupted parties, $\mathcal{Q}_h := \mathcal{Q} \setminus \mathcal{Q}_c$ as the set of next-epoch honest parties, and $|\mathcal{Q}_c| \leq t_2 - 1$. It maintains sets $\mathcal{M}, \mathcal{N}$(initially set to $\emptyset$).

- Upon receiving $(\textsc{Handover}, \mathsf{sid}, \overline{\mathsf{psk}}_i, \{\mathsf{ppk}^{(i)}_j\}_{j \in [c_1]})$ from P_i:
  - Set $\mathcal{N} := \mathcal{N} \cup i$, send $(\textsc{HandoverNotify}, \mathsf{sid}, P_i)$ to $\mathcal{S}$. Wait until $|\mathcal{N}| = c_1$;
  - Set $\{\mathsf{ppk}_j\}_{j \in [c_1]}$ as the majority of $\{\{\mathsf{ppk}^{(i)}_j\}_{j \in [c_1]}\}_{i \in [c_1]}$;
  - Set $\mathcal{M}$ as the first t_1 parties that satisfy $\mathsf{ppk}_j = g^{\overline{\mathsf{psk}}_j}$, set $\mathsf{psk}_j := \overline{\mathsf{psk}}_j$.
- Upon receiving $(\textsc{CorruptShares}, \mathsf{sid}, (\{j, \mathsf{npsk}_j\}_{Q_j \in \mathcal{Q}_c}))$ from $\mathcal{S}$:
  - Compute $\mathsf{gsk} = \prod_{j \in \mathcal{M}} \lambda_j \cdot \mathsf{psk}_j$, where $\{\lambda_j\}_{j \in \mathcal{M}}$ are Lagrange coefficients, i.e., $\lambda_j := \prod_{\ell \in \mathcal{M} \setminus \{j\}} \frac{\ell}{\ell - j}$;
  - Construct random polynomial $F(x) := \sum_{b=0}^{t_2-1} a_b \cdot x^b$ under the restriction $F(j) = \mathsf{npsk}_j$ for $Q_j \in \mathcal{Q}_c$, and $F(0) = \mathsf{gsk}$;
  - Compute $\mathsf{npsk}_i := F(i)$ and $\mathsf{nppk}_i = g^{\mathsf{npsk}_i}$ for $i \in [c_2]$;
  - Send $(\textsc{Abort}, \mathsf{sid})$ to $\mathcal{S}$ and wait for an answer$(\textsc{Abort}, \mathsf{sid}, b)$;
  - Upon receiving the answer, if $b = 1$ then halt;
  - Otherwise, send $(\textsc{Handover}, \mathsf{sid}, (\{\mathsf{nppk}_j\}_{j \in [c_2]}))$ to $\mathcal{S}$ and $(\textsc{Handover}, \mathsf{sid}, (\mathsf{npsk}_i, \{\mathsf{nppk}_j\}_{j \in [c_2]}))$ to $Q_i, i \in [c_2]$.

Fig. 8: Handover ideal functionality $\mathcal{F}^{t_1,t_2,c_1,c_2}_{\mathsf{handover}}[\mathbb{G}]$

IV. Evolving Committee

This section will describe the evolving committee secret sharing in detail, as well as Ring VRF $\Pi_{\mathsf{Ring\text{-}VRF}}$ and zk proof for decryption correctness Π_{Dec} as building blocks.

A. Ring VRF

In this section, we introduce a new notion, called *Ring Verifiable Random Function* (Ring-VRF). Given a set of public keys $\{\mathsf{pk}_0, \ldots, \mathsf{pk}_{N-1}\}$ and an input $x \in \{0,1\}^*$, R-VRF allows the user to invoke its private key sk to generate $v \leftarrow \mathsf{PRF}_{\mathsf{sk}}(x)$, and convince the public that

$$\mathcal{R}_{\mathsf{VRF}} = \left\{ \begin{array}{l} ((\mathsf{pk}_0, \ldots, \mathsf{pk}_{N-1}, x, v), \mathsf{sk}, \ell) \mid \\ \ell \in \{0, \ldots, N-1\} \ \wedge \ (\mathsf{pk}_\ell, \mathsf{sk}) \in \mathcal{R}_{\mathsf{pk}} \wedge \\ v \leftarrow \mathsf{PRF}_{\mathsf{sk}}(x) \end{array} \right\}$$

where $\mathcal{R}_{\mathsf{pk}} = \{(\mathsf{pk}, \mathsf{sk}) | \mathsf{pk} = g^{\mathsf{sk}}\}$.

We give a formal definition of Ring VRF as follows. It consists of 3 PPT algorithms.

- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\lambda)$: The key generation algorithm produces a public key pk and a secret key sk.
- $(v, \pi) \leftarrow \mathsf{Eval}(\mathsf{sk}, x, \{\mathsf{pk}_0, \ldots, \mathsf{pk}_{N-1}\})$: The evaluation algorithm takes as input secret key sk, x and a set of public keys $\{\mathsf{pk}_0, \ldots, \mathsf{pk}_{N-1}\}$. It outputs v and a proof π for $\mathcal{R}_{\mathsf{VRF}}$.
- $0/1 \leftarrow \mathsf{Verify}(x, v, \pi)$: The verifier algorithm takes as input (x, v, π) and outputs 0 or 1.

1) PRF: Let $\mathbb{G}$ be a cyclic group with prime order q, and denote g as the generator. $H : \{0,1\}^* \mapsto \mathbb{G}$ is a hash function that maps an arbitrary length string to a group element. In this work, we adopt PRF $\mathsf{PRF} : \{0,1\}^* \times \mathbb{Z}_q \mapsto \mathbb{G}$ as follows: $\mathsf{PRF}_{\mathsf{sk}}(x) := H(x)^{\mathsf{sk}}$. Note that, in the actual implementation, H can be implemented by the "Elligator 2" mapping proposed in [4].

2) Our construction: We first describe the intuition of the protocol. In the first step of our protocol, the prover create a new commitment of sk, denoted as $c = \mathsf{Com}_{\mathsf{ck}}(\mathsf{sk}; t)$. Then, the protocol can be viewed as two parts run in parallel. One part is inspired by one-out-of-many protocol [13] that ensures pk_ℓ is in the ring. The other part ensures that c and v corresponds to the same sk. Combining the two parts together we get the protocol for Ring VRF. In practice, Pedersen commitment can be used as the additive homomorphic commitment scheme, i.e., $\mathsf{Com}_{\mathsf{ck}}(a; r) = g^a h^r$.

In this paragraph we specify the polynomial $p_i(e)$ used in the protocol. Following the idea of [13], we write $i = i_1 \ldots i_n$ and $\ell = \ell_1 \ldots \ell_n$ in binary, and we let δ_{ij} be Kronecker's delta, i.e., $\delta_{\ell\ell} = 1$ and $\delta_{i\ell} = 0$ for $i \neq \ell$. We let $f_j = \ell_j e + a_j$, let $f_{j,1} = f_j = \ell_j e + a_j = \delta_{1\ell_j} e + a_j$ and $f_{j,0} = e - f_j =$

$(1-\ell_j)e - a_j = \delta_{0\ell_j}e - a_j$. Then, $p_i(e) = \prod_{j=1}^n f_{j,i_j}$ has the form:

$$p_i(e) = \prod_{j=1}^{n}(\delta_{i_j\ell_j}e) + \prod_{k=0}^{n-1} p_{i,k}e^k = \delta_{ij}e^n + \prod_{k=0}^{n-1} p_{i,k}e^k \quad (1)$$

Finally, Fig. 9 shows the Sigma protocol for the relation $\mathcal{R}_{\mathsf{VRF}}$. By Fiat-Shamir heuristic [11] we can transform it into a non-interactive zero knowledge proof. Moreover, we add a tag in the hash input, which is the ephemeral public key in our application scenario.

Sigma protocol for Ring VRF $\Pi_{\mathsf{Ring\text{-}VRF}}$

CRS: the commitment key $\mathsf{ck} = h$;
Statement: $\mathsf{pk}_0, \dots, \mathsf{pk}_{N-1}, x, v$;
Witness: sk, ℓ such that $\ell \in \{0, \dots, N-1\} \wedge \mathsf{pk}_\ell = g^{\mathsf{sk}} \wedge v = H(x)^{\mathsf{sk}}$.

Prover:

- $t \leftarrow \mathbb{Z}_q,\ c = g^{\mathsf{sk}}h^t$;
- For $i = 0, \dots, N-1$, compute $c_i = \mathsf{pk}_i / c$;
- For $j = 1, \dots, n$
  - $r_j, a_j, s_j, t_j, \rho_k \leftarrow \mathbb{Z}_q$;
  - $c_{\ell_j} = \mathsf{Com}_{\mathsf{ck}}(\ell_j; r_j)$;
  - $c_{a_j} = \mathsf{Com}_{\mathsf{ck}}(a_j; s_j)$;
  - $c_{b_j} = \mathsf{Com}_{\mathsf{ck}}(\ell_j a_j; t_j)$;
  - $c_{d_k} = \prod_i c_i^{p_{i,k}} \mathsf{Com}_{\mathsf{ck}}(0; \rho_k)$, using $k = j-1$ and $p_{i,k}$ from Eq. 1;
- $s', t' \leftarrow \mathbb{Z}_q,\ m_1 = g^{s'}h^{t'}, m_2 = u^{s'}$, using $u = H(x)$;
- $P \to V$: $(c, c_{\ell_1}, c_{a_1}, c_{b_1}, c_{d_0}, \dots, c_{\ell_n}, c_{a_n}, c_{b_n}, c_{d_{n-1}}, m_1, m_2)$.

Verifier:

- $e \leftarrow \mathbb{Z}_q$;
  In the non-interactive version, $e = \mathsf{hash}(\mathsf{tag}, \mathsf{ck}, \mathsf{pk}_0, \dots \mathsf{pk}_{N-1}, r, x, c, c_{\ell_1}, c_{a_1}, c_{b_1}, c_{d_0} \dots, c_{\ell_n}, c_{a_n}, c_{b_n}, c_{d_{n-1}}, m_1, m_2)$.
- $V \to P$: e.

Prover:

- For $j = 1, \dots, n$
  - $f_j = \ell_j e + a_j$;
  - $z_{a_j} = r_j e + s_j$;
  - $z_{b_j} = r_j(e - f_j) + t_j$;
  - $z_d = (-t)e^n - \sum_{k=0}^{n-1} \rho_k e^k$;
- $y_1 = s' + \mathsf{sk} \cdot e,\ y_2 = t' + te$;
- $P \to V$: $(f_1, z_{a_1}, z_{b_1}, \dots, f_n, z_{a_n}, z_{b_n}, z_d, y_1, y_2)$.

Verifier:

- For $i = 0, \dots, N-1$, compute $c_i = \mathsf{pk}_i / c$;
- For all $j \in \{1, \dots, n\}$, check
  - $c_{\ell_j}^e c_{a_j} = \mathsf{Com}_{\mathsf{ck}}(f_j; z_{a_j})$;
  - $c_{\ell_j}^{e-f_j} c_{b_j} = \mathsf{Com}_{\mathsf{ck}}(0; z_{b_j})$;
- check $\prod_i c_i^{\prod_{j=1}^n f_{j,i_j}} \cdot \prod_{k=0}^{n-1} c_{d_k}^{-e^k} = \mathsf{Com}_{\mathsf{ck}}(0; z_d)$, using $f_{j,1} = f_j$ and $f_{j,0} = e - f_j$;
- check $g^{y_1}h^{y_2} = m_1 c^e,\ u^{y_1} = m_2 v^e$, using $u = H(x)$;
- Output 1 iff all the checks pass.

Fig. 9: Sigma protocol for Ring VRF $\Pi_{\mathsf{Ring\text{-}VRF}}$

Theorem 1. *Let* Com *be an additive homomorphic non-interactive commitment scheme that is perfectly hiding and computationally binding with adversarial advantage* ε. *Protocol* $\Pi_{\mathsf{zk\text{-}VRF}}$ *as described in Fig. 9 for relation* $\mathcal{R}_{\mathsf{VRF}}$ *is a three-move public coin zero-knowlege protocol with perfect completeness,* $(n+1)$*-special soundness with adversarial advantage* ε *and perfect special honest verifier zero-knowledge.*

Proof. To prove that protocol $\Pi_{\mathsf{Ring\text{-}VRF}}$ is perfect complete observe that $\prod_{j=1}^n f_{j,i_j}$ is a polynomial in the challenge e of the form $p_i(e) = \delta_{i\ell}e^n + \prod_{k=0}^{n-1} p_{i,k}e^k$. When $\mathsf{pk}_\ell = g^{\mathsf{sk}}$, c_ℓ is a commitment to 0 we therefore get that $c_\ell^{\prod_{j=1}^n f_{j,\ell_j}}$ in the verification equation is a commitment to 0. Since $c_{d_k} = \prod_i c_i^{p_{i,k}} \mathsf{Com}_{\mathsf{ck}}(0; \rho_k)$, by the additive homomorphic property of Com, the terms involving $p_{i,k}$ are all cancelled. Hence, the verification equation $\prod_i c_i^{\prod_{j=1}^n f_{j,i_j}} \cdot \prod_{k=0}^{n-1} c_{d_k}^{-e^k} = \mathsf{Com}_{\mathsf{ck}}(0; z_d)$ always holds. And $g^{y_1}h^{y_2} = m_1c^e, u^{y_1} = m_2v^e$ also hold. Therefore the protocol is complete.

To prove that protocol $\Pi_{\mathsf{Ring\text{-}VRF}}$ is $(n+1)$-sound, we show that an adversary with probability ε of breaking $(n+1)$-soundness can be converted into an adversary that has probability ε of breaking the binding property of the commitment scheme.

Suppose the cheating prover creates $n+1$ accepting responses $f_1^{(0)}, \dots, y_2^{(0)}, \dots, y_2^{(n)}, \dots, z_d^{(n)}$ to $n+1$ different challenges $e^{(0)}, \dots, e^{(n)}$ on the same initial message $c, c_{\ell_1} \dots, m_2$, we first show that $\ell_j \in \{0, 1\}$. Pick two responses $f_j^{(0)}, z_{a_j}^{(0)}, z_{b_j}^{(0)}$ and $f_j^{(1)}, z_{a_j}^{(1)}, z_{b_j}^{(1)}$ to challenges $e^{(0)}, e^{(1)}$ on the commitments c_{a_j}, c_{b_j}. By combining the verification equations we get $c_{\ell_j}^{e^{(0)}-e^{(1)}} = \mathsf{Com}_{\mathsf{ck}}(f_j^{(0)} - f_j^{(1)}; z_{a_j}^{(0)} - z_{a_j}^{(1)})$ and $c_{\ell_j}^{e^{(0)}-f_j^{(0)}-e^{(1)}+f_j^{(1)}} = \mathsf{Com}_{\mathsf{ck}}(0; z_{b_j}^{(0)} - z_{b_j}^{(1)})$. Defining $\ell_j = \frac{f_j^{(0)} - f_j^{(1)}}{e^{(0)} - e^{(1)}}$ and $\gamma_j = \frac{z_{a_j}^{(0)} - z_{a_j}^{(1)}}{e^{(0)} - e^{(1)}}$ we extract an opening of $c_{\ell_j} = \mathsf{Com}_{\mathsf{ck}}(\ell_j; \gamma_j)$. Furthermore, since $c_{\ell_j}^{e^{(0)}-f_j^{(0)}-e^{(1)}+f_j^{(1)}} = c_{\ell_j}^{(1-\ell_j)(e^{(0)}-e^{(1)})} = \mathsf{Com}_{\mathsf{ck}}(\ell_j(1-\ell_j)(e^{(0)}-e^{(1)}); \gamma_j(1-\ell_j)(e^{(0)}-e^{(1)})) = \mathsf{Com}_{\mathsf{ck}}(0; z_{b_j}^{(0)} - z_{b_j}^{(1)})$, we either get a breach of the binding property of the commitment scheme or we have $\ell_j(1-\ell_j) = 0$, which implies $\ell_j \in \{0, 1\}$.

Let a_j be the number commited in c_{a_j}, from the verification equation $c_{\ell_j}^e c_{a_j} = \mathsf{Com}_{\mathsf{ck}}(f_j; z_{a_j})$ we get that $f_j^{(0)} = \ell_j e^{(0)} + a_j, \dots, f_j^{(n)} = \ell_j e^{(n)} + a_j$ for all $j = 1, \dots, n$ unless the adversary breaks the binding property of the commitment scheme, it must hold for all challenges. From $f_{j,1} = f_j$ and $f_{j,0} = e - f_j$ we get $f_{j,1} = \ell_j e + a_j$ and $f_{j,0} = (1-\ell_j)e - a_j$. For $i \neq \ell$ we therefore get that $\prod_{j=1}^n f_{j,i_j}$ is a degree at most $n-1$ polynomial $p_i(e)$, and for $i = \ell$ it is a polynomial of the form $p_\ell(e) = e^n + \dots$ This means we can rewrite $\prod_i c_i^{\prod_{j=1}^n f_{j,i_j}} \cdot \prod_{k=0}^{n-1} c_{d_k}^{-e^k} = \mathsf{Com}_{\mathsf{ck}}(0; z_d)$ as

$$c_\ell^{e^n} \cdot \prod_{k=0}^{n-1} c_{*_k}^{e^k} = \mathsf{Com}_{\mathsf{ck}}(0; z_d)$$

for some fixed $c_{*_0}, \dots, c_{*_{n-1}}$, which can be computed from commitments in the statement and the initial message.

Observe that the vectors $(1, e^{(\beta)}, \dots, (e^{(\beta)})^n), \beta = 0, \dots, n$ can be viewed as rows in a Vandermonde matrix because $e^{(0)}, \dots, e^{(n)}$ are all different. The matrix is invertible and we can therefore find a linear combination $(\alpha_0, \dots, \alpha_n)$ of the rows that gives us the vector $(0, \dots, 0, 1)$. Combining the $n+1$ accepting verification equations we therefore get

$$c_\ell = \prod_{\beta=0}^{n} (c_\ell^{(e^{(\beta)})^n} \cdot \prod_{k=0}^{n-1} c_{*_k}^{(e^{(\beta)})^k})^{\alpha_\beta} = \mathsf{Com}_{\mathsf{ck}}(0; \sum_{\beta=0}^{n} \alpha_\beta z_d^{(\beta)})$$

This gives us an extracted opening of c_ℓ to 0. Since $c_i = \mathsf{pk}_i / c$, denote $\mathsf{pk}_\ell = g^{\mathsf{sk}_\ell}$ we have $c = g^{\mathsf{sk}_\ell} h^t$.

As to the PRF part, to show c and v corresponds to the same $\mathsf{sk} = \mathsf{sk}_\ell$, we pick two responses $y_1^{(0)}, y_2^{(0)}, y_1^{(1)}, y_2^{(1)}$ to the challenges $e^{(0)}, e^{(1)}$. We have $g^{y_1^{(0)}} h^{y_2^{(0)}} = m_1 c^{e^{(0)}}$, $g^{y_1^{(1)}} h^{y_2^{(1)}} = m_1 c^{e^{(1)}}$. Divide the two equations we get $c = g^{(y_1^{(0)} - y_1^{(1)})(e^{(0)} - e^{(1)})^{-1}} h^{(y_2^{(0)} - y_2^{(1)})(e^{(0)} - e^{(1)})^{-1}}$. Similarly, we have $u^{y_1^{(0)}} = m_2 v^{e^{(0)}}, u^{y_1^{(1)}} = m_2 v^{e^{(1)}}$. Divide the two equations we get $v = u^{(y_1^{(0)} - y_1^{(1)})(e^{(0)} - e^{(1)})^{-1}}$. Thus we extract $\mathsf{sk} = (y_1^{(0)} - y_1^{(1)})(e^{(0)} - e^{(1)})^{-1}$ and $t = (y_2^{(0)} - y_2^{(1)})(e^{(0)} - e^{(1)})^{-1}$.

To prove that protocol $\Pi_{\mathsf{Ring\text{-}VRF}}$ is perfect special honest verifier zero-knowledge, we build a special honest verifier zero-knowledge simulator that is given a challenge $e \in \{0,1\}^\lambda$. First, it picks the elements of the response uniformly at random as $f_1, \dots, y_2 \leftarrow \mathbb{Z}_q$. It picks $c = g^a h^r$ at random and computes $c_i = \mathsf{pk}_i / c$. It picks $c_{\ell_1}, \dots, c_{\ell_n}, c_{d_1}, \dots, c_{d_{n-1}} \leftarrow \mathsf{Com}_{\mathsf{ck}}(0)$ as random commitments to 0. Then, it computes $c_{a_j} = c_{\ell_j}^{-x} \mathsf{Com}_{\mathsf{ck}}(f_j; z_{a_j})$, $c_{b_j} = c_{\ell_j}^{f_j - x} \mathsf{Com}_{\mathsf{ck}}(0; z_{b_j})$, $c_{d_0} = \prod_i c_i^{\prod_{j=1}^n f_{j,i_j}} \cdot \prod_{k=1}^{n-1} c_{d_k}^{-x^k} \cdot \mathsf{Com}_{\mathsf{ck}}(0; -z_d)$, $m_1 = g^{y_1} h^{y_2} c^{-e}$, $m_2 = u^{y_1} v^{-e}$. Finally, it returns the simulated initial message and response $(c, c_{\ell_1}, \dots, m_2, f_1, \dots, y_2)$.

Now we argue that the simulated proof and the real proof are perfectly indistinguishable. First, in both real proofs and simulated proofs $f_1, \dots, y_2$ are uniformly random in $\mathbb{Z}_q$. Furthermore, by the verification equations, $c_{a_1}, c_{b_1}, \dots, c_{a_n}, c_{b_n}, c_{d_0}, m_1, m_2$ are determined by on $f_1, \dots, y_2$ and $c, c_{\ell_1}, \dots, c_{d_{n-1}}$ both in real and in simulated proofs. Since the commitment scheme Com is perfectly hiding, the adversary cannot distinguish between the real and simulated commitments $c, c_{\ell_1}, \dots, c_{\ell_n}, c_{d_1}, \dots, c_{d_{n-1}}$. Hence, the protocol is perfect zero-knowledge. □

B. ZK Proof for Hybrid Decryption

In this section, we construct a Sigma protocol to show the correctness of decryption w.r.t. the hybrid encryption scheme, and then use Fiat-Shamir heuristic to transform it into a non-interactive version. The non-interactive zero knowledge protocol will be used later in our DKG protocol and handover protocol. To prove the correctness of decryption, actually, the prover only needs to provide $c_2 = (c_1)^{\mathsf{sk}}$. The verifier can then computes $k \leftarrow \mathsf{hash}(c_2)$ and checks the decryption by $m = \mathsf{AES\text{-}GCM}_k^{-1}(u)$ (cf. II-A). The Sigma protocol is depicted in Fig. 10. Similar to Ring VRF, by Fiat-Shamir heuristic we can transform it into a non-interactive zero knowledge proof. We denote the NIZK protocol as $\Pi_{\mathsf{NIZK}_{\mathsf{Dec}}}$.

Theorem 2. *The Sigma protocol in Fig. 10 for relation $\mathcal{R}$ is perfectly complete, 2-special sound and perfectly special honest verifier zero-knowledge.*

Sigma protocol for decryption correctness Π_{Dec}

Statement: $\mathsf{pk} := (g, h), c_1, c_2$

Witness: sk such that $c_2 = c_1^{\mathsf{sk}} \wedge h = g^{\mathsf{sk}}$

Prover:
- Pick random $t \leftarrow \mathbb{Z}_q$ and compute $w_1 := g^t, w_2 := c_1^t$;
- $P \rightarrow V$: (w_1, w_2).

Verifier:
- $e \leftarrow \mathbb{Z}_q$;
  In the non-interactive version, $e = \mathsf{hash}(\mathsf{pk}, c_1, c_2, w_1, w_2)$.
- $V \rightarrow P$: e.

Prover:
- Compute $z := t + e \cdot \mathsf{sk} \pmod{q}$;
- $P \rightarrow V$: z.

Verifier:
- Output 1 if and only if the following holds:
  - $h^e \cdot w_1 = g^z$;
  - $c_2^e \cdot w_2 = c_1^z$.

Fig. 10: Sigma protocol for decryption correctness Π_{Dec}

Proof. To see that the Sigma protocol is perfectly complete observe that $c_2 = c_1^{\mathsf{sk}}$ and $h = g^{\mathsf{sk}}$. Thus we have $g^z = g^t \cdot g^{e \cdot \mathsf{sk}} = g^t \cdot (g^{\mathsf{sk}})^e = h^e \cdot w_1$ and $c_1^z = c_1^t \cdot c_1^{e \cdot \mathsf{sk}} = c_1^t \cdot (c_1^{\mathsf{sk}})^e = c_2^e \cdot w_2$.

To see that the Sigma protocol is 2-special sound, we assume the prover respond to two different challenges e, e' with z, z'. Divide the verification equation we get $g^{z-z'} = h^{e-e'}$ and $c_1^{z-z'} = c_2^{e-e'}$. From $g^{z-z'} = h^{e-e'}$ we extract $\mathsf{sk} = (z - z')(e - e')^{-1}$. Combining it with $c_1^{z-z'} = c_2^{e-e'}$ we can prove that $c_2 = c_1^{\mathsf{sk}}$.

To see that the Sigma protocol is perfectly special honest verifier zero-knowledge we build a special honest verifier zero-knowledge simulator that is given a challenge e. It picks $z \leftarrow \mathbb{Z}_p$, then computes $w_1 = g^z / h^e$ and $w_2 = c_1^z / c_2^e$. We can see that for a real statement, i.e., $c_2 = c_1^{\mathsf{sk}}$, we have $w_2 = w_1^r$ for both real proof and simulated proof, which means real proof and simulated proof have the same distribution. □

C. Distributed Key Generation

For distributed key generation, we modify the distributed key generation protocol by Gennaro and Goldfeder [12] to achieve UC secure. We use a NIZK proof and split the broadcast step into two phase. In phase 1 parties broadcast the ciphertexts; In phase 2 parties broadcast the verification components. This allows the ideal world simulator to compute the verification components reversely to achieve indistinguishability. The protocol $\Pi_{\mathsf{DKG}}^{t,n}$ is depicted in Fig. 11.

D. Anonymous Committee Sortition

In the committee sortition process, we hope to hide the committee members so as to avoid coercion or DoS attack. The committee sortition protocol works as follows. For protocol setup, each party owns his/her secret key sk and there is an agreed-upon random number x from the blockchain. At the beginning of the protocol, each party computes $v = \mathsf{PRF}_{\mathsf{sk}}(x)$

Distributed key generation protocol $\Pi^{t,n}_{\mathsf{DKG}}[\mathbb{G}]$

Setup: Each party $P_i \in \mathcal{P}$ is associated with a ephemeral public key epk_i, and P_i holds the corresponding ephemeral secret key esk_i.

- Upon receiving $(\textsc{KeyGen}, \mathsf{sid}, P_i)$ from $\mathcal{Z}$, $P_i \in \mathcal{P}$ does the following:
  - Select $u_i \leftarrow \mathbb{Z}_q$ and compute $c_i = \mathsf{Com}(g^{u_i})$, where $\mathsf{Com}(g^{u_i}) = \mathsf{hash}(g^{u_i}; r_i)$. Send $(\textsc{Write}, \mathsf{sid}, c_i)$ to $\mathcal{G}_{\mathsf{blockchain}}$. Wait until all parties have broadcast c_i;
  - Choose a random degree-$(t-1)$ polynomial G_i with $G_i(0) = u_i$;
  - For each $k \in [n]$, Set $\sigma_{i,k} = G_i(k)$ and use the k's ephemeral public key to encrypt it with hybrid encryption scheme, setting $\mathsf{ct}_{i,k} = \mathsf{Enc}_{\mathsf{epk}_k}(\sigma_{i,k})$;
  - Let the coefficients of $G_i(x)$ be $a_{i,0}, \ldots, a_{i,t-1}$. Compute $A_{i,0} = g^{a_{i,0}} = g^{u_i}, \ldots, A_{i,t-1} = g^{a_{i,t-1}}$;
  - Send $(\textsc{Write}, \mathsf{sid}, (\mathsf{ct}_{i,1}, \ldots, \mathsf{ct}_{i,n}))$ to $\mathcal{G}_{\mathsf{blockchain}}$;
  - Send $(\textsc{Read}, \mathsf{sid})$ to $\mathcal{G}_{\mathsf{blockchain}}$ and use esk_i to decrypt them to get $\{\sigma_{j,i}\}_{j\in[n]\setminus\{i\}}$;
  - Send $(\textsc{Write}, \mathsf{sid}, ((g^{u_i}; r_i), A_{i,1}, \ldots, A_{i,t-1}))$ to $\mathcal{G}_{\mathsf{blockchain}}(G_i(0) = u_i$ implies $A_{i,0} = g^{u_i})$;
  - Send $(\textsc{Read}, \mathsf{sid})$ to $\mathcal{G}_{\mathsf{blockchain}}$ to get $\{(g^{u_j}; r_j), A_{j,1}, \ldots, A_{j,t-1}\}_{j\in[n]\setminus\{i\}}$;
  - Check if $c_i = \mathsf{hash}(g^{u_i}; r_i)$;
  - Check if the shares are right, i.e., $g^{\sigma_{k,i}} = \prod_{j=0}^{t-1}(A_{k,j})^{i^j}$. If not, the receiver will generate a NIZK proof π to prove that he receives the wrong share(cf. non-interactive version of Fig. 10) and send $(\textsc{Write}, \mathsf{sid}, \pi)$ to $\mathcal{G}_{\mathsf{blockchain}}$.(One valid complaint against P_j will disqualify P_j, and the protocol will abort.);
  - Compute secret key share $\mathsf{psk}_i = \sum_{j\in[n]} \sigma_{j,i}$;
  - Set public key as $\mathsf{gpk} = \prod_{j\in[n]} g^{u_j}$, secret key as $\mathsf{gsk} = \sum_{j\in[n]} u_j$, each party's partial public key as $\mathsf{ppk}_j = g^{x_j} = \prod_{\ell\in[n]} g^{\sigma_{\ell,j}} = \prod_{\ell\in[n]}(\prod_{k=0}^{t-1}(A_{\ell,k})^{j^k})$;
  - Return $(\textsc{KeyGen}, \mathsf{sid}, (\mathsf{psk}_i, \{\mathsf{ppk}_j\}_{j\in[n]}))$ to $\mathcal{Z}$.
- Upon receiving $(\textsc{ReadPk}, \mathsf{sid})$ from $\mathcal{Z}$, return $(\textsc{ReadPk}, \mathsf{sid}, \mathsf{gpk})$ to $\mathcal{Z}$.

Fig. 11: Distributed key generation protocol $\Pi^{t,n}_{\mathsf{DKG}}[\mathbb{G}]$

and check if v is less than a certain threshold T, which represents his/her winning the sortition. The sortition winners will broadcast a Ring VRF proof π along with his/her ephemeral public key epk. Other members watch the blockchain and record the valid ephemeral public keys.

The ideal functionality $\mathcal{F}^{p}_{\mathsf{Anon\text{-}sortition}}$ can be realized by a Ring VRF-based sortition protocol in the $\{\mathcal{F}_{\mathsf{RO}}, \mathcal{G}_{\mathsf{blockchain}}\}$-hybrid model. The simulator $\mathcal{S}$ performs as follows: Upon receiving $(\textsc{Init}, \mathsf{sid}, T[\mathsf{sid}])$ from $\mathcal{F}^{p}_{\mathsf{Anon\text{-}sortition}}$, $\mathcal{S}$ simulates $\mathcal{F}_{\mathsf{RO}}$ so that $v_i < T$ for $i : b^{\mathsf{sid}}_{P_i} = 1$. Upon receiving $(\textsc{Reveal}, \mathsf{sid}, m_i)$ from $\mathcal{F}^{p}_{\mathsf{Anon\text{-}sortition}}$, $\mathcal{S}$ simulates a Ring VRF proof π and sends $(\textsc{Write}, \mathsf{sid}, (\pi, m_i))$ to $\mathcal{G}_{\mathsf{blockchain}}$. Since $\mathcal{F}_{\mathsf{RO}}$ is simulated by $\mathcal{S}$, the real world and ideal world are indistinguishable.

E. Handover

The handover protocol utilizes the Shamir secret sharing scheme to re-share the secret keys from the previous committee to the new committee. In particular, it uses Feldman's VSS to achieve verifiability and accountability. Similar to the DKG protocol, a NIZK proof for decryption correctness is used to identify the malicious behaviors and the broadcast step is split into two phase to ensure UC security. The detail of handover protocol is depicted in Fig. 12.

Note that, our handover protocol achieves identifiable abort. Namely, if a malicious committee member does not follow the protocol, the honest party will raise a complain against him/her and he/she will be disqualified. Then, the remaining parties will re-run the protocol so the the protocol will eventually terminate. In Bool Network, these disqualified committee members will be punished when the handover protocol finishes.

F. Threshold Signature

For threshold signature, we support common signature schemes used in blockchain cryptocurrencies, including BLS [2], Schnorr [22] and ECDSA [14]. To generate a BLS signature, the committee only needs to reconstruct the secret in the exponent. A Schnorr signature can be generated by the additive homomorphism property of Shamir secret sharing. As to the most difficult ECDSA, we adopt the threshold signature protocol proposed by Gennaro and Goldfeder [12], which achieves one round threshold ECDSA with identifiable abort.

V. Security Analysis

In this section, we will formally prove that our DKG protocol $\Pi^{t,n}_{\mathsf{DKG}}[\mathbb{G}]$ and handover protocol $\Pi^{t_1,t_2,c_1,c_2}_{\mathsf{handover}}[\mathbb{G}]$ UC-realize DKG ideal functionality $\mathcal{F}^{t,n}_{\mathsf{DKG}}[\mathbb{G}]$ and handover ideal functionality $\mathcal{F}^{t_1,t_2,c_1,c_2}_{\mathsf{handover}}[\mathbb{G}]$, respectively.

Theorem 3. *Let the hybrid encryption scheme* HEnc *be IND-CPA secure with adversary advantage* $\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{HEnc}}(\mathcal{A}, \lambda)$*. Let the NIZK protocol for decryption correctness* $\Pi_{\mathsf{NIZK}_{\mathsf{Dec}}}$ *be statistically sound with soundness error* $\mathsf{Adv}^{\mathsf{sound}}_{\mathsf{NIZK}_{\mathsf{Dec}}}(\mathcal{A}, \lambda)$ *and perfect zero-knowledge. The protocol* $\Pi^{t,n}_{\mathsf{DKG}}[\mathbb{G}]$ *described in fig. 11 UC-realized* $\mathcal{F}^{t,n}_{\mathsf{DKG}}[\mathbb{G}]$ *against static corruption up to* $t-1$ *parties in the* $\{\mathcal{G}_{\mathsf{blockchain}}, \mathcal{F}_{\mathsf{RO}}\}$*-hybrid model with distinguishing advantage*

$$n^2 \cdot \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{HEnc}}(\mathcal{A}, \lambda) + n(t-1) \cdot \mathsf{Adv}^{\mathsf{sound}}_{\mathsf{NIZK}_{\mathsf{Dec}}}(\mathcal{A}, \lambda) \ .$$

Proof. We first construct a simulator $\mathcal{S}$ such that any non-uniform PPT environment $\mathcal{Z}$ cannot distinguish between the ideal world execution $\mathsf{EXEC}_{\mathcal{F}^{t,n}_{\mathsf{DKG}}[\mathbb{G}],\mathcal{S},\mathcal{Z}}$ and the real world execution $\mathsf{EXEC}^{\mathcal{G}_{\mathsf{blockchain}},\mathcal{F}_{\mathsf{RO}}}_{\Pi^{t,n}_{\mathsf{DKG}}[\mathbb{G}],\mathcal{A},\mathcal{Z}}$. In the ideal world, the parties interact with functionality $\mathcal{F}^{t,n}_{\mathsf{DKG}}[\mathbb{G}]$ and the corrupted parties are controlled by $\mathcal{S}$; in the real world, the parties $\mathcal{P} = \{P_1, \ldots, P_n\}$ run protocol $\Pi^{t,n}_{\mathsf{DKG}}[\mathbb{G}]$ in the $\{\mathcal{G}_{\mathsf{blockchain}}, \mathcal{F}_{\mathsf{RO}}\}$-hybrid world and the corrupted parties are controlled by a dummy adversary $\mathcal{A}$ who simply forwards messages from/to $\mathcal{Z}$.

Simulator. The simulator $\mathcal{S}$ internally runs $\mathcal{A}$, forwarding messages to/from the environment $\mathcal{Z}$. The simulator simulates the following interactions with $\mathcal{A}$:

- Upon receiving $(\textsc{KeyGenNotify}, \mathsf{sid}, P_i)$ from the ideal functionality $\mathcal{F}^{t,n}_{\mathsf{DKG}}[\mathbb{G}]$ about an honest generator P_i, the simulator $\mathcal{S}$ controls P_i to do the following:

The handover protocol $\Pi^{t_1,t_2,c_1,c_2}_{\mathsf{handover}}[\mathbb{G}]$

Setup: Each next-epoch committee member $Q_i \in \mathcal{Q}$ is associated with an ephemeral public key epk_i, and Q_i holds the corresponding ephemeral secret key esk_i. The protocol is parameterized with threshold t_1, t_2, previous-epoch committee size c_1, next-epoch committee size c_2. The shares held by $\mathcal{C}_1$ define a degree-$(t_1 - 1)$ polynomial F_1 with $F_1(0) = \sigma$, where each seat j holds a share $\mathsf{psk}_j = F_1(j)$. All parties agree on the non-zero evaluations points $I = \{1, 2, \dots, c_1\}$ used for a t_1-of-c_1 Shamir secret-sharing scheme.

Upon receiving $(\textsc{Handover}, \mathsf{sid}, \overline{\mathsf{psk}}_i, \{\mathsf{ppk}_j\}_{j\in[c_1]})$ from $\mathcal{Z}$:

- Previous-epoch committee member $P_i \in \mathcal{P}$ does the following:
  - Send $(\textsc{Write}, \mathsf{sid}, \{\mathsf{ppk}^{(i)}_j\}_{j\in[c_1]})$ to $\mathcal{G}_{\mathsf{blockchain}}$;
  - Send $(\textsc{Read}, \mathsf{sid})$ to $\mathcal{G}_{\mathsf{blockchain}}$ and set $\{\mathsf{ppk}_j\}_{j\in[c_1]}$ as the majority of $\{\{\mathsf{ppk}^{(k)}_j\}_{j\in[c_1]}\}_{k\in[c_1]}$;
  - Choose a random degree-$(t_2 - 1)$ polynomial G_i with $G_i(0) = \mathsf{psk}_j$;
  - For each $k \in [c_2]$, Set $\sigma_{i,k} = G_i(k)$ and use the k's ephemeral public key to encrypt it with hybrid encryption scheme, setting $\mathsf{ct}_{i,k} = \mathsf{Enc}_{\mathsf{epk}_k}(\sigma_{i,k})$;
  - Let the coefficients of $G_i(x)$ be $a_{i,0}, \dots, a_{i,t_2-1}$. Compute $A_{i,1} = g^{a_{i,1}}, \dots, A_{i,t_2-1} = g^{a_{i,t_2-1}}$ ($A_{i,0} = \mathsf{ppk}_i$, which is publicly known, so there is no need to broadcast it);
  - Send $(\textsc{Write}, \mathsf{sid}, (\mathsf{ct}_{i,1}, \dots, \mathsf{ct}_{i,c_2}))$ to $\mathcal{G}_{\mathsf{blockchain}}$. Wait until all parties have broadcast $(\mathsf{ct}_{j,1}, \dots, \mathsf{ct}_{j,c_2})$;
  - Send $(\textsc{Write}, \mathsf{sid}, (A_{i,1}, \dots, A_{i,t_2-1}))$ to $\mathcal{G}_{\mathsf{blockchain}}$.
- Next-epoch committee member $Q_i \in \mathcal{Q}$ does the following:
  - Send $(\textsc{Read}, \mathsf{sid})$ to $\mathcal{G}_{\mathsf{blockchain}}$ and use esk to decrypt the ciphertexts to get $\{\sigma_{j,i}\}_{j\in[c_1]}$;
  - Send $(\textsc{Read}, \mathsf{sid})$ to $\mathcal{G}_{\mathsf{blockchain}}$ to get $\{A_{j,k}\}_{j\in[c_1]}$;
  - Check if the shares are right, i.e., $g^{\sigma_{j,k}} = \prod_{n=0}^{t_2-1}(A_{j,n})^{k^n}$, where $A_{j,0} = \mathsf{ppk}_j$. If not, the receiver will broadcast a NIZK proof to prove that he receives the wrong share(cf. non-interactive version of Fig. 10);(One valid complain against P_j will disqualify P_j and the protocol will abort.)
  - Let the indices of the first t_1 valid messages be set $\mathcal{I}$, $\mathcal{I} \subseteq [c_1]$ and $|\mathcal{I}| = t_1$. Compute the share of the global secret corresponding to seat i as $\mathsf{npsk}_i = \sum_{j\in\mathcal{I}} \lambda_j \cdot \sigma_{j,i}$, where $\lambda_j := \prod_{\ell\in\mathcal{I}\setminus\{j\}} \frac{\ell}{\ell-j}$;
  - Compute each new committee member's partial public key as $\mathsf{nppk}_k = \prod\limits_{j\in\mathcal{I}} (g^{\sigma_{j,k}})^{\lambda_j} = \prod\limits_{j\in\mathcal{I}} (\prod_{n=0}^{t_2-1}(A_{j,n})^{k^n})^{\lambda_j}, k = 1, \dots, c_2$, where $\lambda_j := \prod_{\ell\in\mathcal{I}\setminus\{j\}} \frac{\ell}{\ell-j}$;
  - Return $(\textsc{Handover}, \mathsf{sid}, (\mathsf{npsk}_i, \{\mathsf{nppk}_j\}_{j\in[c_2]}))$ to $\mathcal{Z}$.

Fig. 12: The handover protocol $\Pi^{t_1,t_2,c_1,c_2}_{\mathsf{handover}}[\mathbb{G}]$

- Compute $c_i = \mathsf{Com}(g^0)$, where $\mathsf{Com}(g^0) = \mathsf{hash}(g^0; r_i)$. Send $(\textsc{Write}, \mathsf{sid}, c_i)$ to $\mathcal{G}_{\mathsf{blockchain}}$. Wait until all parties have broadcast c_i.
- For $P_k \in \mathcal{P}_h$, set $\sigma_{i,k} = 0$; for $P_k \in \mathcal{P}_c$, select $r_k \leftarrow \mathbb{Z}_q$ and set $\sigma_{i,k} = r_k$. Use the k's ephemeral public key to encrypt it with hybrid encryption scheme, setting $\mathsf{ct}_{i,k} = \mathsf{Enc}_{\mathsf{epk}_k}(\sigma_{i,k})$;
- Send $(\textsc{Write}, \mathsf{sid}, (\mathsf{ct}_{i,1}, \dots, \mathsf{ct}_{i,n}))$ to $\mathcal{G}_{\mathsf{blockchain}}$.

- Once the simulated $\mathcal{G}_{\mathsf{blockchain}}$ receives $(\textsc{Write}, \mathsf{sid}, (\mathsf{ct}_{i,1}, \dots, \mathsf{ct}_{i,n}))$ from all corrupted parties, the simulator $\mathcal{S}$ does the following:
  - Use honest parties' secret keys to decrypt the corresponding ciphertexts to get $\sigma_{i,j}, P_i \in \mathcal{P}_c, P_j \in \mathcal{P}_h$.
  - For all $P_j \in \mathcal{P}_c$, compute $\mathsf{psk}_j = \sum_{i\in[n]} \sigma_{i,j}$ and send $(\textsc{CorruptShares}, \mathsf{sid}, (\{j, \mathsf{psk}_j\}_{P_j\in\mathcal{P}_c}))$ to the ideal functionality $\mathcal{F}^{t,n}_{\mathsf{DKG}}[\mathbb{G}]$.
- Once the simulated $\mathcal{G}_{\mathsf{blockchain}}$ receives $(\textsc{Write}, \mathsf{sid}, (A_{j,1}, \dots, A_{j,t-1}))$ from all corrupted parties, the simulator $\mathcal{S}$ does the following:
  - If $c_i = \mathsf{hash}(g^{u_i}; r_i)$ and $g^{\sigma_{i,k}} = \prod_{j=0}^{t-1}(A_{i,j})^{k^j}$ hold for all $P_i \in \mathcal{P}_c$, set $b = 0$, else set $b = 1$. Send $(\textsc{Abort}, \mathsf{sid}, b)$ to the ideal functionality $\mathcal{F}^{t,n}_{\mathsf{DKG}}[\mathbb{G}]$.
- Upon receiving $(\textsc{KeyGen}, \mathsf{sid}, (\mathsf{psk}_i, \{\mathsf{ppk}_j\}_{j\in[n]}))$ from the ideal functionality $\mathcal{F}^{t,n}_{\mathsf{DKG}}[\mathbb{G}]$, the simulator $\mathcal{S}$ does the following:
  - Compute $\{A_{i,j}\}_{P_i\in\mathcal{P}_h, 0\le j\le t-1}$ that satisfies $g^{\sigma_{i,k}} = \prod_{j=0}^{t-1}(A_{i,j})^{k^j}, P_i \in \mathcal{P}_h, P_k \in \mathcal{P}_c$ and $\mathsf{ppk}_i = \prod_{j\in[n]}(\prod_{k=0}^{t-1}(A_{j,k})^{i^k}), P_i \in \mathcal{P}_h$.
  - For all honest parties $P_i \in \mathcal{P}_h$, send $(\textsc{Write}, \mathsf{sid}, ((A_{i,0}, r_i), A_{i,1}, \dots, A_{i,t-1}))$ to $\mathcal{G}_{\mathsf{blockchain}}$, where simulated $\mathcal{F}_{\mathsf{RO}}$ opens c_i to $(A_{i,0}, r_i)$.
- Once an honest party receives $(\textsc{ReadPk}, \mathsf{sid})$ from the environment $\mathcal{Z}$, compute $\mathsf{gpk} = \prod_{i\in[n]} \mathsf{ppk}_i$, return $(\textsc{ReadPk}, \mathsf{sid}, \mathsf{gpk})$.

Indistinguishability.

The indistinguishability is proven through a series of hybrid worlds $\mathcal{H}_0, \dots, \mathcal{H}_3$.

Hybrid $\mathcal{H}_0$: It is the real world execution $\mathsf{EXEC}^{\mathcal{G}_{\mathsf{blockchain}}, \mathcal{F}_{\mathsf{RO}}}_{\Pi^{t,n}_{\mathsf{DKG}}[\mathbb{G}], \mathcal{A}, \mathcal{Z}}$.

Hybrid $\mathcal{H}_1$: $\mathcal{H}_1$ is the same as $\mathcal{H}_0$ except that in $\mathcal{H}_1$, the messages $(\mathsf{ct}_{i,1}, \dots, \mathsf{ct}_{i,n})$ sent by the honest user P_i are replaced by simulated encryptions of 0.

Claim: If the hybrid encryption scheme HEnc is IND-CPA secure with adversary advantage $\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{HEnc}}(\mathcal{A}, \lambda)$, $\mathcal{H}_0$ and $\mathcal{H}_1$ are indistinguishable with distinguishing advantage at most $n^2 \cdot \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{HEnc}}(\mathcal{A}, \lambda)$.

Proof: The indistinguishability under chosen plaintext attack means that the adversary cannot distinguish encryption of 0 and encryption of a number. There are at most n^2 simulated ciphertexts, so the advantage in $\mathcal{H}_1$ is no more than $n^2 \cdot \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{HEnc}}(\mathcal{A}, \lambda)$.

Hybrid $\mathcal{H}_2$: $\mathcal{H}_2$ is the same as $\mathcal{H}_1$ except that in $\mathcal{H}_2$, when $P_j \in \mathcal{P}_c$ produce a valid complaint against an honest party, $\mathcal{S}$ will abort.

Claim: If the NIZK protocol is statistically sound with soundness error $\mathsf{Adv}^{\mathsf{sound}}_{\mathsf{NIZK}_{\mathsf{Dec}}}(\mathcal{A}, \lambda)$, $\mathcal{H}_1$ and $\mathcal{H}_2$ are indistinguishable with distinguishing advantage at most $n(t-1) \cdot \mathsf{Adv}^{\mathsf{sound}}_{\mathsf{NIZK}_{\mathsf{Dec}}}(\mathcal{A}, \lambda)$.

Proof: By the soundness of the NIZK protocol the probability that $\mathcal{S}$ will give up is negligible. There are at most $n(t-1)$ statements, so the advantage in $\mathcal{H}_1$ is no more than $n(t-1) \cdot \mathsf{Adv}^{\mathsf{sound}}_{\mathsf{NIZK}_{\mathsf{Dec}}}(\mathcal{A}, \lambda)$.

Hybrid $\mathcal{H}_3$: $\mathcal{H}_3$ is the same as $\mathcal{H}_2$ except that in $\mathcal{H}_3$, c_i is opened with the simulated $\mathcal{F}_{\mathsf{RO}}$.

Claim: $\mathcal{H}_2$ and $\mathcal{H}_3$ are perfectly indistinguishable.

Proof: As long as the encoding of $\mathcal{F}_{\mathsf{RO}}$ has not been exhausted, the simulated $\mathcal{F}_{\mathsf{RO}}$ is the same as real $\mathcal{F}_{\mathsf{RO}}$.

The adversary's view of $\mathcal{H}_3$ is identical to the simulated view $\mathsf{EXEC}_{\mathcal{F}_{\mathsf{DKG}}^{t,n}[\mathbb{G}],\mathcal{S},\mathcal{Z}}$. Therefore, no PPT $\mathcal{Z}$ can distinguish the view of the ideal execution from the view of real execution with advantage more than $n^2 \cdot \mathsf{Adv}_{\mathsf{HEnc}}^{\mathsf{IND\text{-}CPA}}(\mathcal{A},\lambda) + n(t-1) \cdot \mathsf{Adv}_{\mathsf{NIZK}_{\mathsf{Dec}}}^{\mathsf{sound}}(\mathcal{A},\lambda)$.

□

Theorem 4. *Let the hybrid encryption scheme* HEnc *be IND-CPA secure with adversary advantage* $\mathsf{Adv}_{\mathsf{HEnc}}^{\mathsf{IND\text{-}CPA}}(\mathcal{A},\lambda)$. *Let the NIZK protocol for decryption correctness* $\Pi_{\mathsf{NIZK}_{\mathsf{Dec}}}$ *be statistically sound with soundness error* $\mathsf{Adv}_{\mathsf{NIZK}_{\mathsf{Dec}}}^{\mathsf{sound}}(\mathcal{A},\lambda)$ *and perfect zero-knowledge. The protocol* $\Pi_{\mathsf{handover}}^{t_1,t_2,c_1,c_2}[\mathbb{G}]$ *described in fig. 12 UC-realized* $\mathcal{F}_{\mathsf{handover}}^{t_1,t_2,c_1,c_2}[\mathbb{G}]$ *against static corruption up to* t_1-1 *previous-epoch parties and up to* t_2-1 *next-epoch parties in the* $\mathcal{G}_{\mathsf{blockchain}}$*-hybrid model with distinguishing advantage*

$$c_1c_2 \cdot \mathsf{Adv}_{\mathsf{HEnc}}^{\mathsf{IND\text{-}CPA}}(\mathcal{A},\lambda) + c_1(t_2-1) \cdot \mathsf{Adv}_{\mathsf{NIZK}_{\mathsf{Dec}}}^{\mathsf{sound}}(\mathcal{A},\lambda) \ .$$

Proof. We first construct a simulator $\mathcal{S}$ such that any non-uniform PPT environment $\mathcal{Z}$ cannot distinguish between the ideal world execution $\mathsf{EXEC}_{\mathcal{F}_{\mathsf{handover}}^{t_1,t_2,c_1,c_2}[\mathbb{G}],\mathcal{S},\mathcal{Z}}$ and the real world execution $\mathsf{EXEC}^{\mathcal{G}_{\mathsf{blockchain}}}_{\Pi_{\mathsf{handover}}^{t_1,t_2,c_1,c_2}[\mathbb{G}],\mathcal{A},\mathcal{Z}}$. In the ideal world, the parties interact with functionality $\mathcal{F}_{\mathsf{handover}}^{t_1,t_2,c_1,c_2}[\mathbb{G}]$ and the corrupted parties are controlled by $\mathcal{S}$; in the real world, the parties $\mathcal{P} = \{P_1,\ldots,P_{c_1}\}$ and $\mathcal{Q} = \{Q_1,\ldots,Q_{c_2}\}$ run protocol $\Pi_{\mathsf{handover}}^{t_1,t_2,c_1,c_2}[\mathbb{G}]$ in the $\mathcal{G}_{\mathsf{blockchain}}$-hybrid world and the corrupted parties are controlled by a dummy adversary $\mathcal{A}$ who simply forwards messages from/to $\mathcal{Z}$.

Simulator. The simulator $\mathcal{S}$ internally runs $\mathcal{A}$, forwarding messages to/from the environment $\mathcal{Z}$. The simulator simulates the following interactions with $\mathcal{A}$:

- Upon receiving $(\textsc{HandoverNotify}, \mathsf{sid}, P_i)$ from the ideal functionality $\mathcal{F}_{\mathsf{handover}}^{t_1,t_2,c_1,c_2}[\mathbb{G}]$ about an honest generator P_i, the simulator $\mathcal{S}$ controls P_i to do the following:
  - Send $(\textsc{Write}, \mathsf{sid}, \{\mathsf{ppk}_j^{(i)}\}_{j\in[c_1]})$ to $\mathcal{G}_{\mathsf{blockchain}}$;
  - Send $(\textsc{Read}, \mathsf{sid})$ to $\mathcal{G}_{\mathsf{blockchain}}$ and set $\{\mathsf{ppk}_j\}_{j\in[c_1]}$ as the majority of $\{\{\mathsf{ppk}_j^{(k)}\}_{j\in[c_1]}\}_{k\in[c_1]}$;
  - For $P_k \in \mathcal{P}_h$, set $\sigma_{i,k} = 0$; for $P_k \in \mathcal{P}_c$, select $r_k \leftarrow \mathbb{Z}_q$ and set $\sigma_{i,k} = r_k$. Use the k's ephemeral public key to encrypt it with hybrid encryption scheme, setting $\mathsf{ct}_{i,k} = \mathsf{Enc}_{\mathsf{epk}_k}(\sigma_{i,k})$;
  - Send $(\textsc{Write}, \mathsf{sid}, (\mathsf{ct}_{i,1},\ldots,\mathsf{ct}_{i,c_2}))$ to $\mathcal{G}_{\mathsf{blockchain}}$.
- Once the simulated $\mathcal{G}_{\mathsf{blockchain}}$ receives $(\textsc{Write}, \mathsf{sid}, (\mathsf{ct}_{i,1},\ldots,\mathsf{ct}_{i,c_2}))$ from all corrupted parties, the simulator $\mathcal{S}$ does the following:
  - Use honest parties' secret keys to decrypt the corresponding ciphertexts to get $\sigma_{i,j}, P_i \in \mathcal{P}_c, Q_j \in \mathcal{Q}_h$.
  - Let the indices of the first t_1 valid messages be set $\mathcal{I}$, $\mathcal{I} \subseteq [c_1]$ and $|\mathcal{I}| = t_1$. Compute corrupted parties' new partial secret key as $\mathsf{npsk}_i = \sum_{j\in\mathcal{I}} \lambda_j \cdot \sigma_{j,i}$, where $\lambda_j := \prod_{\ell\in\mathcal{I}\setminus\{j\}} \frac{\ell}{\ell-j}, Q_i \in \mathcal{Q}_c$;
  - Send $(\textsc{CorruptShares}, \mathsf{sid}, (\{j, \mathsf{npsk}_j\}_{Q_j\in\mathcal{Q}_c}))$ to the ideal functionality $\mathcal{F}_{\mathsf{handover}}^{t_1,t_2,c_1,c_2}[\mathbb{G}]$.
- Once the simulated $\mathcal{G}_{\mathsf{blockchain}}$ receives $(\textsc{Write}, \mathsf{sid}, ((g^{u_i}, r_i), A_{j,1},\ldots,A_{j,t-1}))$ from all corrupted parties, the simulator $\mathcal{S}$ does the following:
  - If $g^{\sigma_{i,k}} = \prod_{n=0}^{t_2-1}(A_{i,n})^{k^n}$ holds for all $P_i \in \mathcal{P}_c \cap \mathcal{I}$, set $b = 0$, else set $b = 1$. Send $(\textsc{Abort}, \mathsf{sid}, b)$ to the ideal functionality $\mathcal{F}_{\mathsf{handover}}^{t_1,t_2,c_1,c_2}[\mathbb{G}]$.
- Upon receiving $(\textsc{Handover}, \mathsf{sid}, \{\mathsf{nppk}_j\}_{j\in[c_2]})$ from the ideal functionality $\mathcal{F}_{\mathsf{handover}}^{t_1,t_2,c_1,c_2}[\mathbb{G}]$, the simulator $\mathcal{S}$ does the following:
  - Compute $\{A_{i,j}\}_{P_i\in\mathcal{P}_h, 1\le j\le t_2-1}$ satisfies $g^{\sigma_{i,k}} = \prod_{j=0}^{t_2-1}(A_{i,j})^{k^j}, P_i \in \mathcal{P}_h, Q_k \in \mathcal{Q}_c$ and $\mathsf{nppk}_k = \prod_{j\in\mathcal{I}}(g^{\sigma_{j,k}})^{\lambda_j} = \prod_{j\in\mathcal{I}}(\prod_{n=0}^{t-1}(A_{j,n})^{k^n})^{\lambda_j}, Q_k \in \mathcal{Q}_h$.
  - For all honest parties $P_i \in \mathcal{P}_h$, send $(\textsc{Write}, \mathsf{sid}, (A_{j,1},\ldots,A_{j,t_2-1}))$ to $\mathcal{G}_{\mathsf{blockchain}}$.

Indistinguishability.

The indistinguishability is proven through a series of hybrid worlds $\mathcal{H}_0,\ldots,\mathcal{H}_2$.

Hybrid $\mathcal{H}_0$: It is the real world protocol execution $\mathsf{EXEC}^{\mathcal{G}_{\mathsf{blockchain}}}_{\Pi_{\mathsf{handover}}^{t_1,t_2,c_1,c_2}[\mathbb{G}],\mathcal{A},\mathcal{Z}}$.

Hybrid $\mathcal{H}_1$: $\mathcal{H}_1$ is the same as $\mathcal{H}_0$ except that in $\mathcal{H}_1$, the messages $(\mathsf{ct}_{j,1},\ldots,\mathsf{ct}_{j,c_2})$ sent by the honest user P_i are replaced by simulated encryptions of 0.

Claim: If the hybrid encryption scheme HEnc is IND-CPA secure with adversary advantage $\mathsf{Adv}_{\mathsf{HEnc}}^{\mathsf{IND\text{-}CPA}}(\mathcal{A},\lambda)$, $\mathcal{H}_0$ and $\mathcal{H}_1$ are indistinguishable with distinguishing advantage at most $c_1c_2 \cdot \mathsf{Adv}_{\mathsf{HEnc}}^{\mathsf{IND\text{-}CPA}}(\mathcal{A},\lambda)$.

Proof: The indistinguishability under chosen plaintext attack means that the adversary cannot distinguish encryption of 0 and encryption of a number. There are at most c_1c_2 simulated ciphertexts, so the advantage in $\mathcal{H}_1$ is no more than $c_1c_2 \cdot \mathsf{Adv}_{\mathsf{HEnc}}^{\mathsf{IND\text{-}CPA}}(\mathcal{A},\lambda)$.

Hybrid $\mathcal{H}_2$: $\mathcal{H}_2$ is the same as $\mathcal{H}_1$ except that in $\mathcal{H}_2$, when $P_j \in \mathcal{P}_c$ produce a valid complaint against an honest party, $\mathcal{S}$ will abort.

Claim: If the NIZK protocol is statistically sound with soundness error $\mathsf{Adv}_{\mathsf{NIZK}_{\mathsf{Dec}}}^{\mathsf{sound}}(\mathcal{A},\lambda)$, $\mathcal{H}_1$ and $\mathcal{H}_2$ are indistinguishable with distinguishing advantage at most $c_1(t_2-1) \cdot \mathsf{Adv}_{\mathsf{NIZK}_{\mathsf{Dec}}}^{\mathsf{sound}}(\mathcal{A},\lambda)$.

Proof: By the soundness of the NIZK protocol the probability that $\mathcal{S}$ will give up is negligible. There are at most $c_1(t_2-1)$ statements, so the advantage in $\mathcal{H}_1$ is no more than $c_1(t_2-1) \cdot \mathsf{Adv}_{\mathsf{NIZK}_{\mathsf{Dec}}}^{\mathsf{sound}}(\mathcal{A},\lambda)$.

The adversary's view of $\mathcal{H}_2$ is identical to the simulated view $\mathsf{EXEC}_{\mathcal{F}_{\mathsf{handover}}^{t_1,t_2,c_1,c_2}[\mathbb{G}],\mathcal{S},\mathcal{Z}}$. Therefore, no PPT $\mathcal{Z}$ can distinguish the view of the ideal execution from the view of real execution with advantage more than $c_1c_2 \cdot \mathsf{Adv}_{\mathsf{HEnc}}^{\mathsf{IND\text{-}CPA}}(\mathcal{A},\lambda) + c_1(t_2-1) \cdot \mathsf{Adv}_{\mathsf{NIZK}_{\mathsf{Dec}}}^{\mathsf{sound}}(\mathcal{A},\lambda)$.

□

VI. Implementation and Benchmarks

1) Test network: We launched a local network of 30 full nodes with a (10,15)-committee to test Bool Network in the real environment. It worked for several days with a number of successful committee switches.

2) Evaluations: To evaluate the performance of the cryptographic protocols, a special set of tests were conducted on a workstation equipped with Intel Xeon(Ice Lake) Platinum 8369B CPU @2.9 GHz and 64GB RAM running Ubuntu 20.04 LTS. We deploy 3 virtual key servers, which are responsible

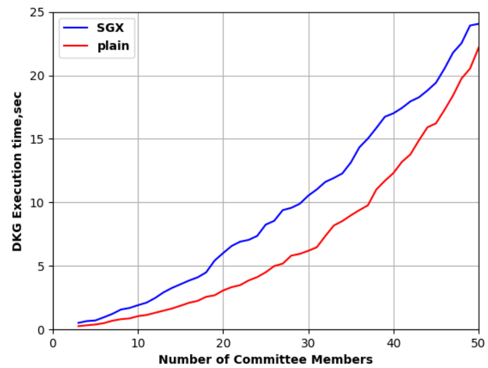

Fig. 13: DKG protocol execution time depending on the number of committee members

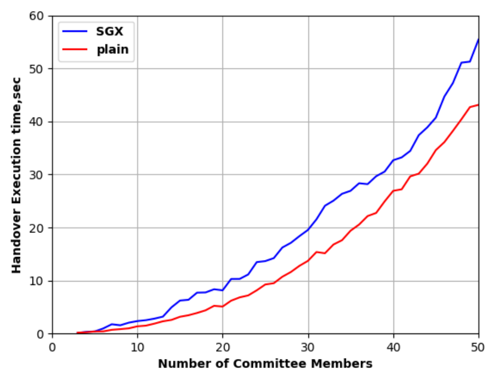

Fig. 14: Handover protocol execution time depending on the number of committee members

for executing the protocol, and 1 manage server, which is responsible for giving instructions, on the workstation. Each key server has 3 threads and we divide the task into pieces to simulate the real execution time.

We benchmarked the running time of distributed key generation protocol, handover protocol and threshold ECDSA w/o Intel SGX for different number of committee members: from 3 to 50. Results were given in Fig. 13, Fig. 14, Fig. 15, respectively.

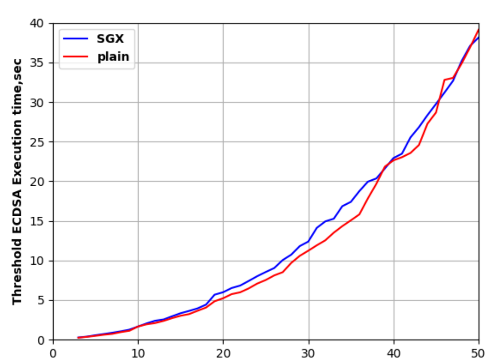

Fig. 15: Threshold signature protocol execution time depending on the number of committee members

For Ring VRF, we benchmarked the prover running time, verifier running time and proof with respect to different ring size: from 3 to 50. Results were given in Fig. 16.

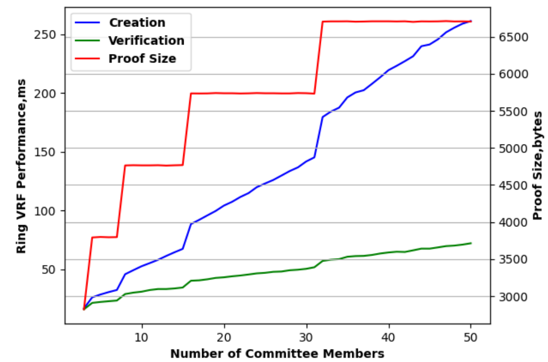

Fig. 16: The prover's running time, verifier's running time and the size of the Ring VRF proof

VII. Conclusion

In this work, we present Bool Network for bitcoin verification layer service. In particular, we propose an efficient Ring-VRF scheme that enables evolving committee with hidden identities. Besides, all the key management process are executed in the TEE for confidentiality and integrity. Compared with existing schemes, our system is more secure and decentralized.

References

[1] Apostolaki, M., Marti, G., Müller, J., Vanbever, L.: Sabre: Protecting bitcoin against routing attacks. arXiv preprint arXiv:1808.06254 (2018)

[2] Barreto, P.S., Kim, H.Y., Lynn, B., Scott, M.: Efficient algorithms for pairing-based cryptosystems. In: Annual international cryptology conference. pp. 354–369. Springer (2002)

[3] Benhamouda, F., Gentry, C., Gorbunov, S., Halevi, S., Krawczyk, H., Lin, C., Rabin, T., Reyzin, L.: Can a public blockchain keep a secret? In: Theory of Cryptography Conference. pp. 260–290. Springer (2020)

[4] Bernstein, D.J., Hamburg, M., Krasnova, A., Lange, T.: Elligator: elliptic-curve points indistinguishable from uniform random strings. In: Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security. pp. 967–980 (2013)

[5] Bernstein, D.J., Lange, T., Schwabe, P.: The security impact of a new cryptographic library. In: International Conference on Cryptology and Information Security in Latin America. pp. 159–176. Springer (2012)

[6] Bracha, G.: Asynchronous byzantine agreement protocols. Information and Computation **75**(2), 130–143 (1987)

[7] Cachin, C., Tessaro, S.: Asynchronous verifiable information dispersal. In: 24th IEEE Symposium on Reliable Distributed Systems (SRDS'05). pp. 191–201. IEEE (2005)

[8] Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: Proceedings 42nd IEEE Symposium on Foundations of Computer Science. pp. 136–145. IEEE (2001)

[9] Canetti, R., Dodis, Y., Pass, R., Walfish, S.: Universally composable security with global setup. In: Theory of Cryptography Conference. pp. 61–85. Springer (2007)

[10] Chen, J., Micali, S.: Algorand: A secure and efficient distributed ledger. Theoretical Computer Science **777**, 155–183 (2019)

[11] Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Conference on the theory and application of cryptographic techniques. pp. 186–194. Springer (1986)

[12] Gennaro, R., Goldfeder, S.: One round threshold ecdsa with identifiable abort. IACR Cryptol. ePrint Arch. **2020**, 540 (2020)

[13] Groth, J., Kohlweiss, M.: One-out-of-many proofs: Or how to leak a secret and spend a coin. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 253–280. Springer (2015)

[14] Johnson, D., Menezes, A., Vanstone, S.: The elliptic curve digital signature algorithm (ecdsa). International journal of information security **1**(1), 36–63 (2001)
[15] Kohlweiss, M., Madathil, V., Nayak, K., Scafuro, A.: On the anonymity guarantees of anonymous proof-of-stake protocols. In: 2021 IEEE Symposium on Security and Privacy (SP). pp. 1818–1833. IEEE (2021)
[16] Lipp, M., Gruss, D., Spreitzer, R., Maurice, C., Mangard, S.: {ARMageddon}: Cache attacks on mobile devices. In: 25th USENIX Security Symposium (USENIX Security 16). pp. 549–564 (2016)
[17] Liu, C., Wang, X.S., Nayak, K., Huang, Y., Shi, E.: Oblivm: A programming framework for secure computation. In: 2015 IEEE Symposium on Security and Privacy. pp. 359–376. IEEE (2015)
[18] Micali, S., Rabin, M., Vadhan, S.: Verifiable random functions. In: 40th annual symposium on foundations of computer science (cat. No. 99CB37039). pp. 120–130. IEEE (1999)
[19] Nayak, K., Fletcher, C.W., Ren, L., Chandran, N., Lokam, S.V., Shi, E., Goyal, V.: Hop: Hardware makes obfuscation practical. In: NDSS. vol. 2017, p. 0 (2017)
[20] Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Annual international cryptology conference. pp. 129–140. Springer (1991)
[21] Rane, A., Lin, C., Tiwari, M.: Raccoon: Closing digital {Side-Channels} through obfuscated execution. In: 24th USENIX Security Symposium (USENIX Security 15). pp. 431–446 (2015)
[22] Schnorr, C.P.: Efficient identification and signatures for smart cards. In: Conference on the Theory and Application of Cryptology. pp. 239–252. Springer (1989)
[23] Tramer, F., Zhang, F., Lin, H., Hubaux, J.P., Juels, A., Shi, E.: Sealed-glass proofs: Using transparent enclaves to prove and sell knowledge. In: 2017 IEEE European Symposium on Security and Privacy (EuroS&P). pp. 19–34. IEEE (2017)
[24] Van Bulck, J., Minkin, M., Weisse, O., Genkin, D., Kasikci, B., Piessens, F., Silberstein, M., Wenisch, T.F., Yarom, Y., Strackx, R.: Foreshadow: Extracting the keys to the intel {SGX} kingdom with transient {Out-of-Order} execution. In: 27th USENIX Security Symposium (USENIX Security 18). pp. 991–1008 (2018)
[25] Xu, Y., Cui, W., Peinado, M.: Controlled-channel attacks: Deterministic side channels for untrusted operating systems. In: 2015 IEEE Symposium on Security and Privacy. pp. 640–656. IEEE (2015)