FPSS: Flexible Proactive Secret Share Scheme

No Author Given

No Institute Given

Abstract. Proactive secret share (PSS) scheme was proposed to preserve a secret that is distributed among a dynamic committee. It allows the periodic hand-off of shares from the old committee to a new committee, to ensure that the *robustness* and *secrecy* can always be guaranteed.
However, when considering applications (such as permissionless blockchains) where a shareholder may leave at any time, the existing PSS schemes do not offer such flexibility as the entire committee needs to be involved interactively to update their shares.
This work proposes FPSS, a **F**lexible **P**roactive **S**ecret **S**hare scheme, to enable efficient membership updates, where only the pair of the leaving and joining shareholders needs to be interactively involved. For the update of a single shareholder, it reduces the communication complexity from $O(n*t)$ to $O(n)$, where t is the threshold to securely reconstruct the secret, and n is the number of a committee. We prove that FPSS achieves *robustness* and *secrecy* after each update. We also provide a proof-of-concept implementation of our protocol.

Keywords: Proactive Secret Share · Flexibility · Blockchain

1 Introduction

Secret share scheme allows a secret owner to distribute a secret among n parties, called a committee, any $t+1$ subset of the n parties can reconstruct the secret. Basically, a secure secret share scheme should satisfy the two properties: 1) *Robustness*. It should always be possible to recover the original secret; 2) *Secrecy*. An adversary who controls at most t shareholders in either the old or new committee should not learn any further information about the secret. PSS allows a secure periodic hand-off, where a secret can be redistributed from the current committee to the next without involving the secret owner, to enable that the robustness and secrecy can always be guaranteed. Such a hand-off phase would be executed among the current and the next committees proactively for every pre-defined time interval. PSS scheme is widely used in many scenarios, such as distributed storage systems [26], or Byzantine Fault Tolerant (BFT) based systems [27].

Problem Statement. The PSS scheme requires periodic share hand-off among committee members who own shares of the same secret. In our scenario, members can leave at any time, and coordinated updates are not preferred.

Strawman Solution. Before leaving, an honest party should transfer his share to another newly joined party. A naive solution is to let the honest party, who is leaving the committee, request an additional phase to hand-off his share with a newly joined party or update his share locally (hand-off his share to himself). We call it an "on-demand" hand-off. Even the update of a single shareholder also requires the entire committee's participation, which is not desirable.

1.1 Contributions

Table 1: Comparison of PSS Schemes.

PSS scheme	Network	Adv	BC?[*]	On-Dmd[†]	Hand-Off			Resilience
					Step	Off-Chain[‡]	On-Chain	
MPSS [21]	p-Sync	Mobile[§]	No	No	8	$O(n^4)$	—[¶]	1/3
COBRA [27]	p-Sync	Adaptive	No	No	4	$O(n^3)$	—	1/3
Shanrang [28]	Async	Mobile	No	No	3	$O(n^3 \log n)$	—	1/4
Yurek et al. [29]	Async	Mobile	No	No	2	$O(n^3)$	—	1/3
Opt-CHURP [18]	Sync	Mobile	Yes	No	3	$O(n^2)$	$O(n)$	1/2
Exp-CHURP-A [18]	Sync	Mobile	Yes	No	3	—	$O(n^2)$	1/2
Exp-CHURP-B [18]	Async	Mobile	Yes	No	3	—	$O(n^3)$	1/3
Benhamouda et al. [9]	Sync	Mobile	Yes	No	2	$O(nt)$ [‖]	$O(n)$	1/4
Goyal et al. [14]	Sync	Adaptive	Yes	No	4 [**]	$O(n^2)$	$O(n)$	1/2
FPSS (This Work)	Sync	Adaptive	Yes	Yes	2	$O(nm)$ [††]	$O(m)$	1/2

[*] Whether the PSS incorporates a blockchain component or not.

[†] Normally, PSS divides a secret's lifetime into epochs, and hand-off will be executed at the beginning of every epoch. "On-Dmd" indicates whether the PSS scheme supports the hand-off to be executed in the middle of an epoch or not.

[‡] We compare the communication cost of the Off-Chain and On-Chain components separately.

[§] "Mobile" and "Adaptive" refer to the *Mobile Adversary* or *Adaptive Adversary*, respectively. We refer the detailed description about our adversary model to Section 3.1.

[¶] The horizontal line denotes that the work did not discuss such an approach.

[‖] t is the threshold, where anyone with $t+1$ valid shares from the same committee can reconstruct the corresponding secret.

[**] 2 rounds for preparation phase and another 2 round for the hand-off phase

[††] m is the number of shareholders leaving the committee concurrently.

This work introduces flexible proactive secret share, FPSS. It adds an additional phase, F-HandOff, to enable effective membership updates. F-HandOff requires only the interaction of the pairs of shareholders who are leaving and joining.

Here are the core innovations of F-HandOff. First, F-HandOff adds *flexibility* for shareholders to hand their shares off to another newly joint one. Second, when multiple shareholders update their shares concurrently, only the pair of the leaving and joining shareholders needs to be interactively involved. The hand-off of FPSS has two cases, *single F-HandOff* and *batched F-HandOff*.

- *Single F-HandOff*. The single F-HandOff involves only two shareholders, the one who is leaving and the other who is joining. It can be executed by any shareholder at any time without interactively communicating with the remaining committee.
- *Batched F-HandOff*. It allows any subset of current committee to leave at the same time without interacting with each other. Communication is only required between the pair of leaving and joining shareholders. Thus, the single F-HandOff cannot be directly used in the batch F-HandOff, since each involved shareholder should have a consensus on the new committee members.

The detailed protocol of FPSS will be presented in Section 3. We also provide the comparison among our proposed FPSS and other existing secret share schemes in Table 1 and the corresponding security discussion in Appendix A. The detailed comparison and related works are referred to Section 6.

Roughly speaking, FPSS employs a blockchain to verify each F-HandOff and as the public accessible storage. The involved shareholders also broadcast messages off-chain to each shareholder in the old and new committees, respectively. Since involved shareholders will communicate via blockchain, any malicious behavior can be detected. The underlying system can incentivize the honest and punish the malicious. While such an incentive mechanism is beyond the scope of this paper, we will not discuss it further. Notice that the two cases of F-HandOff are not tiered. Namely, they are not countermeasures to each other when the fault is detected, but rather the two functionalities for different use cases.

1.2 Applications

The FPSS scheme, like PSS, guarantees that a secret distributed among a committee can be recovered if some conditions are met, while the secret remains confidential to the committee and anyone who cannot meet the conditions. While FPSS provides the additional feature of *flexibility* to enable the efficient hand-off execution. FPSS can also be widely used in a wide range of applications, such as distributed key escrow service [23, 24], multiparty computation (MPC) protocols [12], Byzantine fault tolerant (BFT) related protocols [27], and distributed services [17, 26], etc.

2 Preliminary

2.1 Blockchain

FPSS employs permissionless blockchain as the underlying component. We assume that once the blockchain network receives a transaction, it will be recorded on-chain within Δ time interval. A on-chain transaction will never be modified. The new generated block will be sent to the network and received by other nodes in the network within Δ_b time interval. For the sake of security, the adopted blockchain satisfies the properties of (1) *persistence* — if a transaction tx is confirmed by an honest party, no honest party will ever disagree about the position of tx in the ledger, and (2) *liveness* — if a valid transaction tx is broadcast, it will eventually become confirmed by all honest parties.

2.2 Secret Share Scheme

Secret Share (SS). Assuming that a secret $s = f(0)$, where $f(\cdot)$ is a polynomial of degree t with random coefficients (aside form the constant term) over a finite field. $f(i)$ is know by each node i, where $i \neq 0$ is the unique identifier for nodes.

Proactive Secret Share (PSS). PSS scheme allows a committee $\mathcal{C}$ who holds shares w.r.t to a secret to hand-off their shares to another committee $\mathcal{C}'$. The members in $\mathcal{C}$ and $\mathcal{C}'$ may not always be the same.

Security Definition. A secret share scheme usually should satisfy two properties, *robustness* and *secrecy*. We refer to the description of these two properties in the work

by Goyal et al [14]. *Robustness* requires that it should always be possible to recover the secret. *Secrecy* requires that an adversary should not learn any further information about the secret beyond what has been learned before running the protocol.

2.3 KZG Commitment

KZG commitment [16] allows a prover to generate a commitment to a private polynomial. Let $\mathbb{F}$ be a finite field, $s \in \mathbb{F}$ be a vector, where $s = (s_1, \ldots, s_n)$ satisfies that there is a polynomial $f \in \mathbb{F}[X]$ of degree at most t such that $f(0) = s$ and $f(i) = s_i$ for $i \in \{1, \ldots, n\}$. Each party P_i holds a share s_i and the whole sharing is denoted as $[s]_t$. We present the four algorithms below that we will require in our protocol:

- $\{\langle p, \mathbb{G}, \mathbb{G}_T, e, g\rangle, cpk\} \leftarrow \text{KZG.SETUP}(t, 1^\lambda)$: this algorithm initializes some public parameters, which will be used in the following commitment scheme. It inputs the degree bound t and the unary form of the security parameters, and outputs a commitment key pair (csk, cpk) and a billinaer group $(\langle p, \mathbb{G}, \mathbb{G}_T, e, g\rangle)$. While the commitment secret key csk is safely destroyed, and the commitment public key will be used for the following calculation.
- $\text{KZG.CMT}_f \leftarrow \text{KZG.COMMIT}(f(x), cpk)$: this algorithm generates a commitment for a polynomial. It inputs the polynomial $f(x)$ and the commitment public key cpk, and outputs a commitment KZG.CMT_f on f.
- $\{\langle \text{cmt}_{\text{f(i)}}, \text{w}_{\text{f(i)}}\rangle\} \leftarrow \text{KZG.CRTWTN}(cpk, f(x), i)$: this algorithm generates the witness of the polynomial f on the variable i. It takes and the commitment public key cpk, the polynomial f, and the variable i as inputs, and outputs a witness $w_{f(i)}$ and the commitment of $f(i)$.
- $0/1 \leftarrow \text{KZG.VRFYEVAL}(cpk, \text{KZG.CMT}_f, i, f(i), w_{f(i)})$: this algorithm verifies the commitment on $f(i)$. It takes the commitment public key cpk, the polynomial commitment KZG.CMT_f, the variable i, the commitment of $f(i)$, and the witness $w_f(i)$ as inputs, and outputs 0 if the evaluation is correct, or 1 otherwise.

Let $\mathbb{F}$ be a finite field, $s \in \mathbb{F}$ be a vector, where $s = (s_1, \ldots, s_n)$ satisfies that there is a polynomial $f \in \mathbb{F}[X]$ of degree at most t such that $f(0) = s$ and $f(i) = s_i$ for $i \in \{1, \ldots, n\}$. We provide the DL-based KZG commitment scheme, which will be used in this paper, in Figure 1.

2.4 Verifiable Encryption

The verifiable encryption provides both confidentiality and authenticity for encrypted messages. The scheme works by encrypting a message using a public key and then generating a proof that can be verified to prove the correctness of the encryption.

A verifiable encryption scheme is a encryption scheme with additional proof system.

- $(\text{Enc.PK}, \text{Enc.SK}) \leftarrow \text{ENC.KGEN}(1^{\lambda_2})$. The key generation algorithm of the verifiable encryption scheme. It takes the parameter 1^{λ_2} as input and outputs a pair of private and public key $(\text{Enc.PK}, \text{Enc.SK})$.
- $ct \leftarrow \text{ENC}(\text{Enc.PK}, m)$. The encryption algorithm for encrypting the message m. It takes the public key Enc.PK and message m as inputs, and outputs a ciphertext ct.

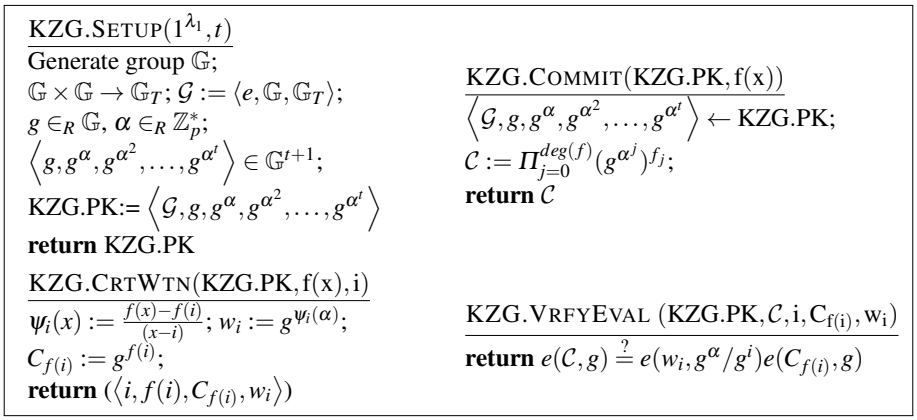

Fig. 1: DL-based KZG Commitment. Let $f(x) \in \mathbb{Z}_p[x]$: $(x-i)$ perfectly divide the polynomial $f(x) - f(i)$ for $i \in \mathbb{Z}_p$.

- $Enc.\pi \leftarrow$ ENC.PRF(Enc.PK, ct, m). The proof algorithm for demonstrating that the ciphertext ct was generated correctly. It takes the public key Enc.PK, the ciphertext ct, and the message m as inputs, and outputs a proof $Enc.\pi$.
- $0/1 \leftarrow$ ENC.VRF($Enc.\pi$, Enc.PK, ct). The corresponding proof system for convincing the verifier that the ciphertext ct was generated correctly and has not been modified since encryption. It takes the proof $Enc.\pi$, public key Enc.PK, and ciphertext ct as inputs, and outputs 0 for rejecting ct or 1 for accepting it. None

Let Γ be a cyclic group of with prime order ρ generated by γ. We assume that γ and ρ are publicly known, and k and k' be further security parameters, where $2^{-k^{\lambda}}$ and $2^{-k'^{\lambda}}$ are the negligible function ($\{0,1\}^k$ is the "challenge space" of the verifier and k' controls the quality of the zero-knowledge property), such that $2^k < min\{p', q', \rho\}$. Let $h_c : \{0,1\}^* \to \{0,1\}^k$ be the hash function. Let $\mathcal{R} := \{(w, \delta) \in W \times \Delta : \gamma^w = \delta\}$ be the relation defined by a generator algorithm $\mathcal{G}'$ which on 1^{λ_3} outputs a description $\Psi = \Psi[\mathcal{R}, W, \Delta]$ of a binary relation $\mathcal{R}$ on $W \times \Delta$. We present the DL-based verifiable encryption by Camenisch et al [11], which will be used in this paper, in Figure 2.

3 FPSS

FPSS, a proactive secret share protocol, is more flexible by adding an F-HandOff algorithm. F-HandOff can be executed by a single shareholder or any subset of shareholders in the same committee concurrently at any time, where only the pair of leaving and joining shareholders need to be interactively involved. This section presents the system model, high-level description of F-HandOff.

3.1 System Model and Security Properties

Notations. We summarise the notation in Table 2 that will be used to explain FPSS.

$\underline{\text{ENC.KGEN}(1^{\lambda_2})}$
$(p',q') \leftarrow \mathbb{P}$;
$p := (2p'+1)$; $q := (2q'+1)$;
$n := pq$, $n' := p'q'$;
$x_1, x_2, x_3 \in_R [n^2/4]$;
$g' \in_R \mathbb{Z}^*_{n^2}$; $g := (g')^{2n}$;
$y_1 := g^{x_1}$; $y_2 := g^{x_2}$; $y_3 := g^{x_3}$;
$hk \in_R \mathcal{H}_\lambda$;
$h := (1+n \bmod n^2) \in \mathbb{Z}^*_{n^2}$;
Enc.SK $:= (hk, n, x_1, x_2, x_3)$;
Enc.PK $:= (hk, n, g, h, y_1, y_2, y_3)$;
return (Enc.SK, Enc.PK)

$\underline{\text{ENC.PRF}(\text{Enc.PK}, ct, m)}$
$(hk, n, g, h, y_1, y_2, y_3) \leftarrow$ Enc.PK;
$(u, e, v) \leftarrow ct$;
$\gamma \leftarrow \mathbb{G}_\rho$, $\delta := \gamma^m$;
$s \in_R [n/4]$, $l := g^m h^s$;
$r', s' \in_R \mathbb{Z}_{-n2^{k+k'-2}, n2^{k+k'-2}}$;
$m' \in_R \mathbb{Z}_{-\rho 2^{k+k'}, \rho 2^{k+k'}}$;
$u' := g^{2r'}$, $e' := y_1^{2r'} h^{2m'}$,
$v' := (y_2 y_3^{\mathcal{H}_{hk}(u,e,L)})^{2r'}$,
$\delta' := \gamma^{m'}$, $l' := g^{m'} h^{s'}$;
$c := h_c(ct || \text{Enc.PK})$;
$\tilde{r} := r' - cr$, $\tilde{s} := s' - cs$, $\tilde{m} := m' - cm$;
$Enc.\pi := (ct, \gamma, L, \delta, l, u', e', v', \delta', l', \tilde{r}, \tilde{s}, \tilde{m})$;
return $Enc.\pi$;

$\underline{\text{ENC}(\text{Enc.PK}, m, L)}$
$r \in_R [n/4]$;
$(hk, n, g, h, y_1, y_2, y_3) \leftarrow$ Enc.PK;
$u := g^r$;
$o := y_1^r h^m$;
$v := abs((y_2 y_3^{\mathcal{H}_{hk}(u,o,L)})^r)$;
$ct := (u, o, v)$;
return ct

$\underline{\text{ENC.VRF}(Enc.\pi, \text{Enc.PK}, \text{ENC.PK})}$
$(ct, \gamma, L, \delta, l, u', e', v', \delta', l', c, \tilde{r}, \tilde{s}, \tilde{m}) \leftarrow Enc.\pi$;
$(u, e, v) \leftarrow ct$; $(hk, n, g, h, y_1, y_2, y_3) \leftarrow$ Enc.PK;
$c := h_c(ct || \text{Enc.PK})$;
If $u' \neq u^{2c} g^{2\tilde{r}}$ or $e' \neq e^{2c} y_1^{2\tilde{r}} h^{2\tilde{m}}$ or
  $v' \neq v^{2c} (y_2 y_3^{\mathcal{H}_{hk}(u,e,L)})^{2\tilde{r}}$ or
  $\delta' \neq \delta^c \gamma^{\tilde{m}}$ or $l' \neq l^c g^{\tilde{m}} h^{\tilde{s}}$ or
  $-n/4 < \tilde{m} < n/4$:
    return 0;
return 1

Fig. 2: DL-based Verifiable Encryption.

Table 2: Notations

Notations	Meaning
$\mathcal{C}, \mathcal{C}'$	the old (current) and new committees respectively, where $\lvert\mathcal{C}\rvert = \lvert\mathcal{C}'\rvert = n$.
$\text{PKs}_\mathcal{C}$	the public key set consisting all the public keys belonging to each of members in $\mathcal{C}$ respectively.
$(\text{PK}_{\mathcal{C}_i}, \text{SK}_{\mathcal{C}_i})$	the key pair of the shareholder $\mathcal{C}_i$.
$\mathcal{P}$ / $\mathcal{P}'$	the leaving / joining shareholders.
$f(\cdot)$, $\zeta(\cdot)$, $\zeta'(\cdot)$	t degree polynomials, where $f(0) = s$, $\zeta(0) = \zeta'(0) = 0$.
$\{\alpha_i\}_\mathcal{C}$	the set of shares owned by each of $\mathcal{C}$ respectively. Such that a number of t subset of $\{\alpha_i\}_\mathcal{C}$ can be used to compute the secret s.
α'_j, α''_j	the re-randomized share generated by the leaving party $\mathcal{P}_j$ in the old commitment $\mathcal{C}$ and by the joining party in the new commitment $\mathcal{C}'$, respectively.
$\{\gamma_i\}$, $\{\gamma'_i\}$	the points located on the polynomial $\zeta(i)$ and $\zeta'(i)$, respectively.

System Model. Our objective is to design a secure and flexible hand-off mechanism that allows honest shareholders to transfer their shares to new members without involving the entire committee. To achieve this goal, we propose the Flexible Proactive Secret

Sharing (FPSS) system, which includes a new hand-off mechanism, F-HandOff, that requires only the leaving and joining shareholders to interact. The F-HandOff process is straightforward: a leaving shareholder $\mathcal{P}$ uploads a message to the blockchain, and the designated party $\mathcal{P}'$ responds with an acceptance message.

Our mechanism enables flexible transfers in dynamic committee scenarios, eliminating the need for full committee involvement. Utilizing the KZG (Section 2.3) and Verifiable Encryption schemes (Section 2.4) enables secure and verifiable hand-offs, assuring accurate transfer of shares to intended recipients.

Blockchain and Network Model. FPSS employs a blockchain, which satisfies the properties of *persistence* and *liveness*, as one of the communication channels. All communications during a flexible hand-off are over both blockchain and the p2p network. The entire flexible hand-off execution does not require the acknowledgment from all recipients.

Adversary Model. We consider FPSS under the adaptive adversary model. Assuming that a computationally bounded adversary $\mathcal{A}$ can adaptively choose at most t shareholders for every Δ_c time interval. That is, anyone can access at most t honest shares in the same committee. Once a shareholder has been corrupted, $\mathcal{A}$ can arbitrarily control his behavior as well as modify the memory. A party is honest before he joined the committee, and once he is assigned a share, $\mathcal{A}$ can decide to corrupt this party or not. For example, if a shareholder $\mathcal{P}_i$ was corrupted by $\mathcal{A}$ in the old committee $\mathcal{C}$, he is still the corrupted shareholder in $\mathcal{C}'$, if he is not left. While if he is corrupted for the first time in $\mathcal{C}'$, he was honest in $\mathcal{C}$.

FPSS inherits the two security properties of PSS, *robustness* and *secrecy*, and introduces an additional property *flexibility*. The formal definitions are given below:

Definition 1 (Robustness). *For any PPT adversary $\mathcal{A}$ with the corruption threshold t, once a secret s is deposited among $n > 2t$ parties by using FPSS, anyone who has $t+1$ honest shares in the same committee can reconstruct a secret s', where $s' = s$.*

Definition 2 (Secrecy). *For any PPT adversary $\mathcal{A}$ with the corruption at most t out of n shares, the probability of $\mathcal{A}$ to calculate a secret s', where $s' = s$ and $n > 2t$, is negligible.*

Definition 3 (Flexibility). *An FPSS is flexible if a share can be successfully handed off among a pair of old and new shareholders, as long as the pair of shareholders correctly execute the hand-off.*

3.2 High-level Description of FPSS

We propose FPSS, which enables each execution of a hand-off interactively involves only the pair of leaving and joining shareholders. It is based on the Shamir secret share protocol [22] with an additional flexible hand-off algorithm, called F-HandOff. FPSS can be deployed in two cases, single hand-off and batched hand-off:

– Single F-HandOff. The single F-HandOff requires only two parties to be interactive: the one who is leaving the committee and the other who is joining the committee. Roughly, each of the two parties contributes n random shares located on a private polynomial that passes through the point $(0,0)$ for each of the committee members, respectively. Fig 3 briefly demonstrates a single F-HandOff. A successful F-HandOff securely hands off a share from the leaving party to the joining party and updates all shares in the committee accordingly.
– Batched F-HandOff. FPSS also allows any subset of the committee to execute the F-HandOff concurrently. While each of the F-HandOff is independent of other concurrent F-HandOffs. To execute a F-HandOff, the leaving party does not need to reach a consensus with other potential leaving party. Additionally, if one F-HandOff fails, it does not affect other concurrent F-HandOff.

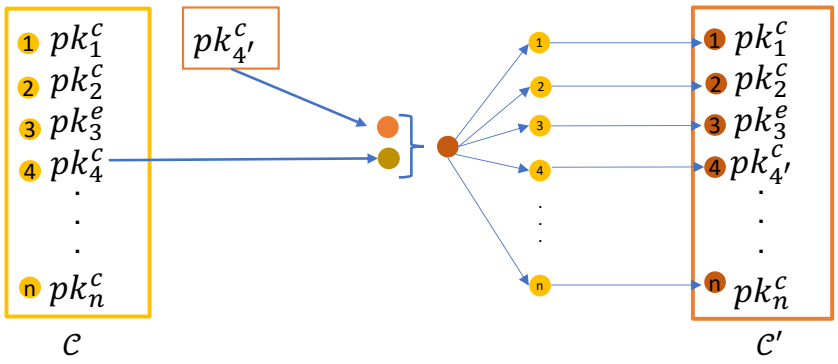

Fig. 3: Overview of F-HandOff

4 Formalization of FPSS

This section describes FPSS in detail. FPSS allows shareholders to leave individually (Section 4.3) and also supports the batched leaving scenario (Section 4.4).

4.1 Setup

Before distribution phase, a trusted third party creates the public key KZG.PK for the KZG commitment scheme and system parameter $\tilde{n} = pq$, where p, q are safe primes, for the verifiable encryption scheme at the start of the protocol (such a trusted party can be replaced by an MPC protocol [13,20] executed among miners or the initial committee).

4.2 Distribution

To distribute a secret s, the secret owner first selects an initial committee $\mathcal{C}^0$ and distributes the secret to its members. Each committee member $\mathcal{P}_i$ is assigned a share α_i of the secret, which is encrypted and recorded on the blockchain for transparency and

immutability. The selection process for the initial committee is assumed to be known by the secret owner and is not considered in this paper. Once the distribution phase is complete, each committee member $\mathcal{P}_i$ holds a unique share α_i of the secret s.

4.3 FPSS - Single F-HandOff

This section describes F-HandOff for the single leaving scenario. That is there is only one shareholder $\mathcal{P}_j \in \mathcal{C}$ who wants to hand-off his share to another party $\mathcal{P}_{j'}$, and a new committee is defined as $\mathcal{C}' = (\mathcal{C}/\mathcal{P}_j) \cup \mathcal{P}_{j'}$.

Once a secret s is distributed among a committee $\mathcal{C}$, each shareholder of the committee $\mathcal{C}_i$ will be assigned a share α_i, and each of their public keys PK is publicly accessible on-chain. Let $\mathcal{P} = \mathcal{C}_j \in \mathcal{C}$ owns α_j be the one, who is leaving the committee, and $\mathcal{P}'$ be the newly joint one. Both $\mathcal{P}$ and $\mathcal{P}'$ then proceed as follows:

1. $\mathcal{P}$ generates a t-degree polynomial $\zeta(\cdot)$ and produces a set of points, $\{\gamma_i\}_{i\in[1,n]}$, where $\gamma_i = \zeta(i)$ and $\zeta(0) = 0$;
2. $\mathcal{P}$ randomizes his share α_j by $\alpha' = \alpha_j + \gamma_j$;
3. $\mathcal{P}$ encrypts the randomized share $\alpha'_{\mathcal{P}'}$ under $\mathcal{P}'$'s public key as well as each of $\{\gamma_i\}_{i\in[1,n]/j}$ under $\mathcal{C}_i$'s public key;
4. $\mathcal{P}$ produces $\Pi_{\mathcal{P}}$ for proving the correctness of these encrypted shares α' and $\{\gamma_i\}$ (the definition of the correctness will be discussed later this Section);
5. $\mathcal{P}$ broadcasts a leaving request, which contains the leaving request indicator, the request ID, the encrypted $\alpha'_{\mathcal{P}'}$, the public key of the designated party, $n-1$ encrypted $\{\gamma_i\}/\gamma_{\mathcal{P}'}$ as well as the encrypted randomized share α', and the corresponding proof $\Pi_{\mathcal{P}}$. Notice that only these encrypted $\{\gamma_i\}/\gamma_{\mathcal{P}'}$ and α' will be recorded on-chain.
6. Once $\mathcal{P}'$ receives the assigned share from $\mathcal{P}$, he generates a t-degree polynomial $\zeta'(\cdot)$ and produces a set of shares, $\{\gamma'_i\}_{i\in[1,n]}$, where $i \neq j$, $\gamma'_i = \zeta'(i)$ and $\zeta'(0) = 0$;
7. $\mathcal{P}'$ encrypts each of $\{\gamma'_i\}$ under $\mathcal{C}_i$'s public key;
8. $\mathcal{P}'$ produces $\Pi_{\mathcal{P}'}$ for proving that these encrypted shares $\{\gamma'_i\}$ are correctly generated (the proof system will be discussed later in this Section);
9. $\mathcal{P}'$ broadcasts a joining request, which contains a signature proving that he is the designated party and the $n-1$ encrypted $\{\gamma'_i\}$ and the corresponding proof for proving the correctness of these encrypted points. Similarly, on-chain record only contains the $n-1$ encrypted $\{\gamma'_i\}$ if the joining request is verified by miners.

The leaving and joining requests always appear on-chain in pairs. That is, once a miner receives a valid leaving request, he will keep this request until he receives the corresponding valid joining request. As a result, the F-HandOff between $\mathcal{P}$ and $\mathcal{P}'$ completes successfully. We use both on-chain and off-chain communications, shareholders in the same committee can check the received share using the on-chain records.

Both $\mathcal{P}$ and $\mathcal{P}'$ do not have to receive acknowledgment messages confirming whether all shareholders accept this request or not until there is a valid reconstruction request or the next periodic hand-off. Thus, F-HandOff can tolerate an extremely long delay for at least $t+1$ honest shareholders to receive their shares, respectively, until there is a valid reconstruction request or the next periodic hand-off.

A successful F-HandOff outputs a new committee $\mathcal{C}' = \mathcal{P}' \cup \{\mathcal{C}/\mathcal{P}\}$ and updates all shares of shareholders in $\mathcal{C}'$, respectively. After a F-HandOff, all honest parties erase their revoked shares.

We then discuss two scenarios when $\mathcal{P}$ or $\mathcal{P}'$ does not follow the protocol correctly.

Discussion 1 *What if $\mathcal{P}$ or $\mathcal{P}'$ does not send the request to leave or join accordingly?*

The F-HandOff is executed on-demand. If the shareholder $\mathcal{P}$ does not send a leaving request, it indicates his intention to stay in the committee. If newly joint shareholder $\mathcal{P}'$ does not respond, $\mathcal{P}$ can designate another party $\mathcal{P}''$ by sending another leaving request to replace the previous one. Since the leaving and joining requests are written to blockchain in pairs, the old leaving request is never recorded. Thus, the new leaving request incurs no additional on-chain cost.

Discussion 2 *What if shareholders did not receive their shares after an F-HandOff?*

This is the "share availability" issue, which can be solved by the on-chain records. That is, any of shareholders can make request to blockchain to retrieve their lost shares. Since the blockchain achieves the *persistence* and *liveness*, at least $t+1$ honest shareholders can retrieve their shares eventually.

Discussion 3 *The compromise of a shareholder's encryption key can result in the vulnerability of all shares encrypted using that key, which poses a significant security risk, especially since all shares are recorded on the blockchain. Given the assumption that the adversary can adapt their corruption strategy to target different shareholders, it is crucial to prevent this vulnerability from occurring.*

It is crucial to prevent this vulnerability from occurring, given the assumption that the adversary can adapt their corruption strategy to target different shareholders.

To address the security risk posed by a compromised shareholder's encryption key, we introduce the forward-secrecy property for the shareholders' public key scheme. This property ensures that even if an adversary compromises a shareholder's private key, they cannot decrypt any ciphertext encrypted using that key before the shareholder updates their keys. To implement this property, we require that honest shareholders update their secret and public key pairs every Δ_c time interval. By doing so, any ciphertext encrypted using a previous key pair remains secure even if the corresponding private key is compromised. This added security measure mitigates the risk of a single compromised shareholder compromising the security of the entire protocol.

Formal single F-HandOff scheme (Figure 5). We then describe the formal single F-HandOff scheme. Recall the notations in Table 2. Let $\mathrm{RPs}_{\mathcal{P}}$ be the set of randomized points and $\{\mathrm{CTR_i}\}_{\mathrm{RPs}_{\mathcal{P}}}$ be the set of encrypted $\mathrm{RPs}_{\mathcal{P}}$.

To leave the committee $\mathcal{C}$, $\mathcal{P}$ should upload a leaving request $\mathrm{LREQ}_{\mathcal{P}} := (\mathrm{LR}_{\mathcal{P}}, \mathrm{ID}_{\mathrm{LR}_{\mathcal{P}}}, \mathrm{PK}_{\mathcal{P}'}, ds, \Pi_{\mathcal{P}})$ to blockchain, where $\mathrm{LR}_{\mathcal{P}}$ is the leaving request indicator, $\mathrm{ID}_{\mathrm{LR}_{\mathcal{P}}}$ denotes the request ID, $\mathrm{PK}_{\mathcal{P}'}$ is the public key of a designated party, ds is a set of encrypted data (defined in Fig 5, line 11) and $\Pi_{\mathcal{P}}$ denotes the proof of the correctness of this leaving request. $\Pi_{\mathcal{P}}$ is outputted by a NIZK proof algorithm, F-HandOff.Prf, which takes a pair of NIZK instance as inputs.

RVC (Req)

1 : **Input**: Req;
2 : **Output**: $1/0$;
3 : **Initiate**: $b = 1$;
4 : **Parse** Req $\rightarrow \{\text{RSIGN}, \text{ID}, \text{JointP or } \sigma, ds, \Pi\}$:
5 : If RSIGN = "LREQ":
6 : If Π satisfies $\mathcal{R}_{\text{LREQ}}$: $b = 0$;
7 : Elif RSIGN = "JREQ":
8 : PK $\leftarrow \mathcal{L}(\text{ID})$;
9 : If PK $=\perp$ or $\text{SIGN.VRFY}_{\text{PK}}(\text{PK}, \sigma, \text{ID}) = \perp$: $b = 0$;
10 : If Π not satisfies $\mathcal{R}_{\text{JREQ}}$: $b = 0$;
11 : Else $b = 0$;
12 : Return b;

Fig. 4: Request Validation Check

The leaving request ($\text{LREQ}_{\mathcal{P}}$) will be verified following the request validation check $\text{RVC}(\text{LREQ}_{\mathcal{P}})$ (see Fig. 4), before it is accepted by the underlying blockchain. Let $i, j \in \mathbb{N}^+$, $i, j \in [1, n]$ and $i \neq j$, the leaving request should satisfy the following statements:

1. for each given encrypted share $(i, \text{ENC}_{\text{PK}_i}(y))$, it should satisfy that $y = \zeta(i)$;
2. for the given point $(j, \text{ENC}_{\text{PK}_j}(y))$, it should satisfy that $y = \alpha_j + \zeta(j)$ and;
3. $\zeta(0) = 0$;

y is private to verifiers. The formal relationship for the preceding proving statement is shown below.

Assuming that $\Pi_{\mathcal{R}_{\mathcal{P}}^{\text{LREQ}}}$ is a proof generated by $\mathcal{P}$ for proving the relation $\mathcal{R}_{\mathcal{P}}^{\text{LREQ}}$. For the NIZK instance $(\chi_{\mathcal{P}}, \omega_{\mathcal{P}})$, where $\chi_{\mathcal{P}} := (\text{CTS}_{\text{PK}_{\mathcal{P}'}, \alpha'_j}, \{\text{CTR}_i\}, \Pi_{\mathcal{P}, \mathcal{R}_{\mathcal{P}}^{\text{LREQ}}})$ and $\omega_{\mathcal{P}} := (\alpha_j, \alpha'_j, \text{RPs}_{\mathcal{P}}, \text{f}(\cdot))$, and the relation $\mathcal{R}_{\mathcal{P}}^{\text{LREQ}}$ defined as follow:

$$\begin{aligned}\mathcal{R}_{\mathcal{P}}^{\text{LREQ}} = \{(\alpha_j, \alpha'_j, \text{RPs}, \text{f}(\cdot)) : \text{KZG.CMT}_{\alpha'_j} = \text{KZG.CMT}_{\alpha_j} + \text{KZG.CMT}_{\{\text{CTR}_i\}[j]} \wedge \\ \text{CTS}_{\text{PK}_{\mathcal{P}'}, \alpha'_j} \Leftrightarrow \text{KZG.CMT}_{\alpha'_j} \wedge \text{for i} \in [1, n] : \{\{\text{CTR}_i\}[i] \Leftrightarrow \text{KZG.CMT}_{\text{RPs}[i]}\} \wedge \\ \text{KZG.CMT}_{\zeta(0)} \Leftrightarrow 0\},\end{aligned} \tag{1}$$

where $\Leftrightarrow$ links two different operations on the same value.

To take over $\mathcal{P}$'s seat in $\mathcal{C}$, $\mathcal{P}'$ should upload a joining request $\text{JREQ}_{\mathcal{P}'} := (\text{JR}_{\mathcal{P}'}, \text{ID}_{\text{LR}_{\mathcal{P}}}, \sigma_{\text{PK}_{\mathcal{P}'}}, ds, \Pi_{\mathcal{P}'})$ to blockchain, where $\text{JR}_{\mathcal{P}'}$ is the joining request indicator, $\text{ID}_{\text{LR}_{\mathcal{P}}}$ is the index of the corresponding leaving request, $\sigma_{\text{PK}_{\mathcal{P}'}}$ is the signature on $\text{ID}_{\text{LR}_{\mathcal{P}}}$ under $\text{PK}_{\mathcal{P}'}$ for the identity authentication, ds is the set of encrypted shares, (defined in Fig 5, line 25) and $\Pi_{\mathcal{P}'}$ is the proof for the validation of the joining request.

The joining request, $\text{JREQ}_{\mathcal{P}'}$, should also satisfy the following statements:

1. for each given point $(i, \text{ENC}_{\text{PK}_i}(y))$, it has $y = \zeta'(i)$;

Single Hand-off

Input: $(\alpha_j, t, \mathcal{C}, \{\mathrm{PK_i}\}_{\mathrm{i}\in[1,\mathrm{n}]}, \mathcal{P}', \mathrm{PK}_{\mathcal{P}'})$

Output : $\mathcal{C}'$

Initiate: $\mathcal{P} = \mathcal{C}_j$, $\mathcal{C}' = \mathcal{C}/\mathcal{P}$, $\mathrm{RPs}_{\mathcal{P}} = \emptyset$, $\{\mathrm{CTR_i}\}_{\mathrm{RPs}_{\mathcal{P}}} = \emptyset$, $\{\mathrm{CTR_i}\}_{\mathrm{RPs}_{\mathcal{P}'}} = \emptyset$, and isValids $= 0$

The leaving party $\mathcal{P}$:

generates a t-degree polynomial $\zeta(\cdot)$, s.t. $\zeta(0) = 0$;

For $i \in [1,n]$, calculates:

$\mathrm{RPs}_{\mathcal{P}}[\mathrm{i}] = \zeta(\mathrm{i})$;

if $i \neq j$: $\{\mathrm{CTR_i}\}_{\mathcal{P}}[\mathrm{i}] = \mathrm{ENC}_{\mathrm{PKs}_{\mathcal{C}}[\mathrm{i}]}(\mathrm{RPs}_{\mathcal{P}}[\mathrm{i}])$;

calculate $\alpha'_j = \alpha_j + \mathrm{RPs}_{\mathcal{P}}[\mathrm{j}]$;

execute $\mathrm{CTS}_{\mathrm{PK}_{\mathcal{P}'},\alpha'_\mathrm{j}} = \mathrm{ENC}_{\mathrm{PK}_{\mathcal{P}'}}(\alpha'_\mathrm{j})$;

execute $\Pi_{\mathcal{P}}$:= F-HandOff.Prf$(\chi_{\mathcal{P}}, \omega_{\mathcal{P}})$ ▶ $\chi_{\mathcal{P}}$ and $\omega_{\mathcal{P}}$ are defined in Equation 1

$ds := (\{\mathrm{CTR_i}\}_{\mathcal{P}}, \mathrm{CTS}_{\mathrm{PK}_{\mathcal{P}'},\alpha'_\mathrm{j}})$

send $\mathrm{LREQ}_{\mathcal{P}} := (LR_{\mathcal{P}}, ID_{LR_{\mathcal{P}}}, \mathrm{PK}_{\mathcal{P}'}, ds, \Pi_{\mathcal{P}})$ to blockchain, and send each of $\{\mathrm{CTR}\}$ and $\mathrm{CTS}_{\mathrm{PK}_{\mathcal{P}'},\alpha'_\mathrm{j}}$ to shareholders in the new committee, respectively;

The joined party $\mathcal{P}'$:

If received $\mathrm{CTS}_{\mathrm{PK}_{\mathcal{P}'},\alpha'_\mathrm{j}}$ from $\mathcal{P}$:

Decrypt $\mathrm{CTS}_{\mathrm{PK}_{\mathcal{P}'},\alpha'_\mathrm{j}}$ and gain α'_j;

generate a t-degree polynomial $\zeta'(\cdot)$, s.t. $\zeta'(0) = 0$;

calculate $\alpha''_j = \alpha'_j + \zeta'(j)$;

For $i \in [1,n], i \neq j$, execute:

$\mathrm{RPs}_{\mathcal{P}'}[\mathrm{i}] = \zeta'(\mathrm{i})$;

$\{\mathrm{CTR_i}\}_{\mathcal{P}'}[\mathrm{i}] = \mathrm{ENC}_{\mathrm{PKs}_{\mathcal{C}'}[\mathrm{i}]}(\mathrm{RPs}_{\mathcal{P}'}[\mathrm{i}])$;

sign $\sigma_{\mathrm{PK}_{\mathcal{P}'}} = \mathrm{SIGN}(\mathrm{SK}_{\mathcal{P}'}, \mathrm{ID}_{\mathrm{LR}_{\mathcal{P}}})$;

execute $\Pi_{\mathcal{P}'}$:= F-HandOff.Prf$(\chi_{\mathcal{P}'}, \omega_{\mathcal{P}'})$ ▶ $\chi_{\mathcal{P}'}$ and $\omega_{\mathcal{P}'}$ are defined in Equation 2.

$ds := \{\mathrm{CTR_i}\}_{\mathrm{RPs}_{\mathcal{P}'}}$

send $\mathrm{JREQ}_{\mathcal{P}'} := (\mathrm{JR}_{\mathcal{P}'}, \mathrm{ID}_{\mathrm{LR}_{\mathcal{P}}}, \sigma_{\mathrm{PK}_{\mathcal{P}'}}, \mathrm{ds}, \Pi_{\mathcal{P}'})$ to blockchain, and send each of $\{\mathrm{CTR}\}_{\mathcal{P}'}$ to shareholders in the new committee, respectively, except for $\mathcal{P}'$ himself;

Blockchain Validation Check:

isValids $= (\mathrm{RVC}(\mathrm{LREQ}_{\mathcal{P}})\&\&\mathrm{RVC}(\mathrm{JREQ}_{\mathcal{P}'}))$;

If isValids:

set $\mathcal{C}' = \mathcal{C}' \cup \mathcal{P}'$, and the F-HandOff succeed;

Else: break;

Fig. 5: FPSS - Single Hand-off

2. $\zeta'(0) = 0$;

where y is private to verifiers. The formal relationship for the preceding proving statement is presented below.

Let $\Pi_{\mathcal{P}',\mathcal{R}_{\mathcal{P}'}^{\text{JREQ}}}$ be the proof generated by $\mathcal{P}'$ under the relation $\mathcal{R}_{\mathcal{P}'}^{\text{JREQ}}$. For the NIZK instance $(\chi_{\mathcal{P}'}, \omega_{\mathcal{P}'})$, where $\chi_{\mathcal{P}'} := (\{\text{CTR}_\text{i}\}_{\text{RPs}_{\mathcal{P}'}}, \Pi_{\mathcal{P}',\mathcal{R}_{\mathcal{P}'}^{\text{JREQ}}})$ and $\omega_{\mathcal{P}'} := (\text{RPs}_{\mathcal{P}'}, \zeta'(\cdot))$, and the relation $\mathcal{R}_{\mathcal{P}'}^{\text{JREQ}}$, where $\mathcal{R}_{\mathcal{P}'}^{\text{JREQ}}$ is defined as:

$$\begin{aligned}\mathcal{R}_{\mathcal{P}'}^{\text{JREQ}} = \{(\text{RPs}_{\mathcal{P}'}, \zeta'(\cdot)) : \text{KZG.CMT}_{\zeta'(0)} \Leftrightarrow 0 \wedge \\ \text{for } i \in [i,n]/j : \{\{\text{CTR}_\text{i}\}_{\text{RPs}_{\mathcal{P}'}}[\text{i}] \Leftrightarrow \text{KZG.CMT}_{\zeta'(\text{i})}\}\end{aligned} \tag{2}$$

4.4 Batched F-HandOff

Section 4.3 introduced the single F-HandOff, enabling a shareholder to transfer his share without involving the rest of committee. In practice, multiple shareholders from the same committee $\mathcal{C}$ might leave simultaneously. This section presents the batched F-HandOff, explaining how FPSS handles such concurrent scenarios.

Intuitively, the batched F-HandOff is consisted of multiple single F-HandOffs. But the real procedure is more complicate than the intuition. Before introducing the batched F-HandOff, we explain why the single F-HandOff cannot be directly executed by different parties in the same committee. For the ease of understanding, our explanation uses a simple example. Let $\mathcal{C}$ be a committee of 3 members, where $\mathcal{C} = \{\mathcal{P}_a, \mathcal{P}_b, \mathcal{P}_c\}$. Assuming that a dealer deposits a secret s among $\mathcal{C}$, such that each of $\mathcal{P}_a$, $\mathcal{P}_b$ and $\mathcal{P}_c$ holds shares α_a, α_b and α_c respectively. Any two of them can reconstruct a value s', where $s' = s$. $\mathcal{P}_a$ and $\mathcal{P}_b$ propose their leaving requests individually at the time t. $\mathcal{P}_a$ decides to hand-off his share α_a to $\mathcal{P}'_a$, and $\mathcal{P}_b$ chooses $\mathcal{P}'_b$ to take his share α_b. Thus, from $\mathcal{P}_a$'s view, the new committee is $\mathcal{C}'_a = \{\mathcal{P}'_a, \mathcal{P}_b, \mathcal{P}_c\}$, while for $\mathcal{P}_b$, the new committee $\mathcal{C}'_b$ is consist of $\{\mathcal{P}_a, \mathcal{P}'_b, \mathcal{P}_c\}$. Since the two requests are proposing concurrently without negotiation, the newest committee $\mathcal{C}'$ should be the intersection of both $\mathcal{C}'_a$ and $\mathcal{C}'_b$, such that $\mathcal{C}' = \{\mathcal{P}'_a, \mathcal{P}'_b, \mathcal{P}_c\}$. The share of each party in $\mathcal{C}'$ are:

- $\alpha_{a'} = \alpha_a + \gamma_{a->a'} + \gamma'_{a'->a'} + \gamma'_{b'->a'}$;
- $\alpha_{b'} = \alpha_b + \gamma_{b->b'} + \gamma'_{b'->b'} + \gamma'_{a'->b'}$;
- $\alpha_c = \alpha_c + \gamma_{a->c} + \gamma_{b->c} + \gamma'_{a'->c} + \gamma'_{b'->c}$;

where the share $\gamma_{a->a'}$ is generated by $\mathcal{P}_a$ and encrypted under $\text{PK}_{\mathcal{P}'_a}$, and so on. Namely, $\mathcal{P}'_a$ losses the share $\gamma_{b->a}$, while $\mathcal{P}'_b$ losses the point $\gamma_{a->b}$, since they are not able to access the decryption keys $\text{SK}_{\mathcal{P}_a}$ and $\text{SK}_{\mathcal{P}_b}$, respectively. Due to the inappropriate hand-off, the new shares $\alpha_{\mathcal{P}'_a}$ or $\alpha_{\mathcal{P}'_b}$ cannot contribute to reconstruct a valid secret. In the demonstrated case above, no one can obtain a valid secret from $\mathcal{C}'$ in the Reconstruction phase, which breaks the *robustness* property.

First, we introduce two solutions, but they have drawbacks.

First Attempt. The incorrect assign issue can be revised by adding another response step. In this step, the leaving parties $\mathcal{P}_a$ and $\mathcal{P}_b$ should disclose their received shares to $\mathcal{P}'_a$ and $\mathcal{P}'_b$, respectively. A successful F-HandOff requires that each of $\mathcal{P}_a$ and $\mathcal{P}_b$ encrypts the two decrypted shares $\gamma_{b->a}$ and $\gamma_{a->b}$ under the public keys $\text{PK}_{\mathcal{P}'_a}$ and $\text{PK}_{\mathcal{P}'_b}$, respectively. However, it is vulnerable to single point of failure. If any of $\mathcal{P}_a$ and

$\mathcal{P}_b$ fails to disclose their received points to $\mathcal{P}'_a$ and $\mathcal{P}'_b$ respectively, the entire hand-off failed as well, albeit the failure is auditable.

Second Attempt. Since γ_b and γ_a are encrypted under $\mathcal{P}_a$ and $\mathcal{P}_b$'s public keys, respectively, we can require that each leaving party $\mathcal{P}$ discloses his secret key to the designated party $\mathcal{P}'$ within the leaving request. However, this attempt leads to a share leakage issue. That is, once $\mathcal{P}'$ knows $\mathrm{SK}_{\mathcal{P}}$, he can also decrypt all history shares owned by $\mathcal{P}$'s, which breaks the *secrecy* requirement.

Our approach. Comparing the two attempts above, it is hard to solve the single point of failure vulnerability in the first attempt. Since it is too strong to assume that there is no single point of failure during the protocol execution. We resolve the share leakage issue in the second attempt by using two key-pairs to guarantee the *secrecy* property and introduce our approach as follows:
Each shareholder $\mathcal{P}$ in the committee $\mathcal{C}$ has two key-pairs. One is called the long-term key-pair $(\mathrm{SK}^l_{\mathcal{P}}, \mathrm{PK}^l_{\mathcal{P}})$ for receiving shares assigned to him. The other is ephemeral key-pair $\{\mathrm{SK}^e_{\mathcal{P}}, \mathrm{SK}^e_{\mathcal{P}}\}$ for receiving the random share from leaving parties and the corresponding joining parties. Thus, when $\mathcal{P}$ is leaving the committee, he should encrypt his $\mathrm{SK}^e_{\mathcal{P}}$ by using $\mathrm{PK}^l_{\mathcal{P}'}$, and include the encrypted $\mathrm{SK}^e_{\mathcal{P}}$ in the leaving request. Thus, $\mathcal{P}'$ can decrypt all the random shares sending to $\mathcal{P}$ but not the original share owned by $\mathcal{P}$.

Each of the batch F-HandOffs is similar to the single F-HandOff with the two key-pair design. To avoid repetition, we omit the batched F-HandOff description here, and provide the formal batched F-HandOff in Figure 6.

Discussion 4 (Single Failure in Batched F-HandOff) *Supposing that there are two concurrent leaving requests,* $\mathrm{LREQ}_{\mathcal{P}_a}$ *and* $\mathrm{LREQ}_{\mathcal{P}_b}$*, and only one joining party responses a joining request, for example* $\mathrm{JREQ}_{\mathcal{P}'_a}$.

To prevent being affected by such a single failure, all shares contained in a request are encrypted under the corresponding leaving party's ephemeral public key.

If both $\mathcal{P}'_{a'}$ and $\mathcal{P}'_{b'}$ succeed to upload a joining request $\mathrm{BJREQ}_{a'}$ and $\mathrm{BJREQ}_{b'}$ on-chain, both F-HandOff executions are succeed. Since $\mathcal{P}_a$ and $\mathcal{P}_b$ also release their ephemeral private key to $\mathcal{P}'_{a'}$ and $\mathcal{P}'_{b'}$ in the leaving request BLREQ_a and BLREQ_b respectively, each of $\mathcal{P}'_{a'}$ and $\mathcal{P}'_{b'}$ can decrypt these shares accordingly. However, if $\mathcal{P}'_{b'}$ fails to response his joining request $\mathrm{BJREQ}_{b'}$, the F-HandOff between $\mathcal{P}_b$ and $\mathcal{P}'_{b'}$ failed. That is $\mathcal{P}_b$ is still a shareholder in the next committee $\mathcal{C}' = \{\mathcal{P}'_{a'}, \mathcal{P}_b, \mathcal{P}_c\}$, and $\mathcal{P}_b$ can decrypt these shares $\alpha_{a->b'}$ and $\alpha_{a'->b'}$ and update his share accordingly. So such a failed concurrent HandOff between $\mathcal{P}_b$ and $\mathcal{P}'_{b'}$ does not affect the concurrent F-HandOff between $\mathcal{P}_a$ and $\mathcal{P}'_{a'}$.

Discussion 5 (Confidential Issue of Batched F-HandOff) *A failed F-HandOff leads to the disclosure of* $SK^e_{\mathcal{P}}$*. Once* $\mathcal{P}$ *discloses* $SK^e_{\mathcal{P}}$ *to* $\mathcal{P}'$*, all messages encrypted under*$SK^e_{\mathcal{P}}$ *can be accessed by* $\mathcal{P}'$ *even if this F-HandOff failed.*

There are two potential solutions. One is that $\mathcal{P}$ can update his $(\mathrm{SK}^e_{\mathcal{P}}, \mathrm{PK}^e_{\mathcal{P}})$ on-chain if his F-HandOff fails. The other is that if the failure caused by some non-malicious reasons, such as the unstable network issue of $\mathcal{P}'$, $\mathcal{P}$ can execute the hand-off phase again with the same $\mathcal{P}'$.

Formal batched F-HandOff protocol (Figure 6). We then describe the scenario that two shareholders in the same committee are leaving concurrently. That is $\mathcal{P}_a$ and

$\mathcal{P}_b$ will hand their shares off to $\mathcal{P'}_a$ and $\mathcal{P'}_b$, respectively. While there is a case that $\mathcal{P'}_b$ did not response a joining request, but $\mathcal{P'}_a$ did. Only the hand-off between $\mathcal{P}_b$ and $\mathcal{P'}_b$ failed, $\mathcal{P}_a$ and $\mathcal{P'}_a$ can still succeed in executing the hand-off.

To leave the committee $\mathcal{C}$, $\mathcal{P}$ uploads a batched leaving request $\text{LREQ}^b_{\mathcal{P}} := (\text{LR}_{\mathcal{P}}, \text{LRID}_{\mathcal{P}}, \text{PK}^{\text{l}}_{\mathcal{P}'}, \text{ds}, \Pi_{\mathcal{P},\mathcal{R}^{\text{BLREQ}}_{\mathcal{P}}})$ to blockchain. Different from it in the single F-HandOff, $\text{PK}^{l}_{\mathcal{P}'}$ indicates the long-term public key of $\mathcal{P}'$. The ds is the set of these encrypted shares (defined in Fig 6,line 15). The prove of a batched leaving request $\Pi_{\mathcal{P},\mathcal{R}^{\text{BLREQ}}_{\mathcal{P}}}$ is same as the single leaving request with an additional statement:

- for the given ephemeral public key $\text{PK}^{e}_{\mathcal{P}}$ and ciphertext $\text{CTSK}_{\text{PK}^{\text{l}}_{\mathcal{P}'},\text{SK}^{\text{e}}_{\mathcal{P}}}$, $\text{SK}^{e}_{\mathcal{P}}$ is the secret key associated with $\text{PK}^{e}_{\mathcal{P}}$, where $\text{CTSK}_{\text{PK}^{\text{l}}_{\mathcal{P}'},\text{SK}^{\text{e}}_{\mathcal{P}}}$ is the ciphertext of $\text{SK}^{e}_{\mathcal{P}}$.

Assuming that $\Pi^{b}_{\mathcal{R}^{\text{BLREQ}}_{\mathcal{P}}}$ is a proof generated by $\mathcal{P}$ in the batched F-HandOff case for proving the relation $\mathcal{R}^{\text{BLREQ}}_{\mathcal{P}}$. For the NIZK instance $(\chi^{b}_{\mathcal{P}}, \omega^{b}_{\mathcal{P}})$, where $\chi^{b}_{\mathcal{P}} := (\text{CTSK}_{\text{PK}^{\text{l}}_{\mathcal{P}'},\text{SK}^{\text{e}}_{\mathcal{P}}}, \text{CTS}_{\text{PK}^{\text{l}}_{\mathcal{P}'},\alpha'_{\text{j}}}, \{\text{CTR}_{\text{i}}\}_{\mathcal{P}}, \text{PK}^{\text{e}}_{\mathcal{P}}, \Pi_{\mathcal{P},\mathcal{R}^{\text{BLREQ}}_{\mathcal{P}}})$ and $\omega_{\mathcal{P}} := (\text{SK}^{e}_{\mathcal{P}}, \alpha_j, \alpha'_j, \text{RPs}_{\mathcal{P}}, \zeta(\cdot))$, the relation $\mathcal{R}^{\text{BLREQ}}_{\mathcal{P}}$ is defined as:

$$
\begin{aligned}
\mathcal{R}^{\text{BLREQ}}_{\mathcal{P}} = \{(\text{SK}^{e}_{\mathcal{P}}, \alpha_j, \alpha'_j, \text{RPs}_{\mathcal{P}}, \zeta(\cdot)) : \text{CTSK}_{\text{PK}^{\text{l}}_{\mathcal{P}'},\text{SK}^{\text{e}}_{\mathcal{P}}} \Leftrightarrow \text{PK}^{\text{e}}_{\mathcal{P}} \wedge \\
\text{KZG.}\textsc{cmt}_{\alpha'_j} \Leftrightarrow \text{KZG.}\textsc{cmt}_{\alpha_j} + \text{KZG.}\textsc{cmt}_{\{\text{CTR}_{\text{i}}\}[\text{j}]} \wedge \\
\text{KZG.}\textsc{cmt}_{\alpha'_j} \Leftrightarrow \text{CTS}_{\text{PK}_{\mathcal{P}'},\alpha'_{\text{j}}} \wedge \text{KZG.}\textsc{cmt}_{\zeta(0)} \Leftrightarrow 0 \wedge \\
\text{for } i \in [1,n]/j : \{\{\text{CTR}_{\text{i}}\}_{\mathcal{P}}[\text{i}] \Leftrightarrow \text{KZG.}\textsc{cmt}_{\text{RPs}_{\mathcal{P}}[\text{i}]}\}\}.
\end{aligned}
\tag{3}
$$

Identical to the single F-HandOff, the party that is designated in the leaving request, uploads to blockchain the joining request $\text{BJREQ}_{\mathcal{P}'} := (\text{JR}_{\mathcal{P}'}, \text{LRID}_{\text{LR}_{\mathcal{P}}}, \sigma_{\text{PK}^{l}_{\mathcal{P}'}}, ds, \Pi_{\mathcal{P}',\mathcal{R}^{\text{BJREQ}}_{\mathcal{P}'}})$.

Let $\Pi_{\mathcal{P}',\mathcal{R}^{\text{BJREQ}}_{\mathcal{P}'}}$ be the proof generated by $\mathcal{P}'$ under the relation $\mathcal{R}^{\text{BJREQ}}_{\mathcal{P}'}$. For the NIZK instance $(\chi_{\mathcal{P}'}, \omega_{\mathcal{P}'})$, where $\chi_{\mathcal{P}'} := (\{\text{CTR}_{\text{i}}\}_{\mathcal{P}'}, \Pi_{\mathcal{P}',\mathcal{R}^{\text{JREQ}}_{\mathcal{P}'}})$ and $\omega_{\mathcal{P}'} := (\text{RPs}_{\mathcal{P}'}, \zeta'(\cdot))$, the relation $\mathcal{R}^{\text{BJREQ}}_{\mathcal{P}'}$ is defined as:

$$
\begin{aligned}
\mathcal{R}^{\text{BJREQ}}_{\mathcal{P}'} = \{(\text{RPs}_{\mathcal{P}'}, \zeta'(\cdot)) : \zeta'(0) = 0 \wedge \\
\text{for } i \in [i,n]/j : \{\{\text{CTR}_{\text{i}}\} \Leftrightarrow \text{KZG.}\textsc{cmt}_{\text{RPs}_{\mathcal{P}'}[\text{i}]}\}\}
\end{aligned}
\tag{4}
$$

None

We refer the detail construction of the relation $\mathcal{R}^{\text{LREQ}^b}_{\mathcal{P}}$ (Equation 1), $\mathcal{R}^{\text{JREQ}}_{\mathcal{P}'}$ (Equation 2), $\mathcal{R}^{\text{BLREQ}}_{\mathcal{P}}$ (Equation 3), and $\mathcal{R}^{\text{BJREQ}}_{\mathcal{P}'}$ (Equation 4) in Section 4.6.

Remark. FPSS introduces F-HandOff, in both Section 4.3 and Section 4.4. As F-HandOff employs the underlying blockchain to verify each of leaving and joining requests, if the pair of leaving and joining shareholders are honest, F-HandOff will succeed eventually. As we assume that the underlying blockchain satisfies the *liveness* property (see Section 2.1), F-HandOff is *flexible*.

We assume that there are at least $t+1$ honest shareholders in the committee, and the formal security is defined in Theorem 1 below.

Batched Hand-off

Initiate: $\mathcal{C}' = \emptyset, \mathcal{P}_a = \mathcal{C}_j, \mathcal{P}_b = \mathcal{C}_{j'}, \mathcal{P}'_a = \mathcal{C}'_j, \mathcal{P}'_b = \mathcal{C}'_{j'}$

Input: $(\alpha_{\mathcal{P}_a}, \alpha_{\mathcal{P}_b}, t, \mathcal{C}, \mathrm{PKs}^{\mathrm{e}}_{\mathcal{C}}, \mathrm{PK}^{\mathrm{l}}_{\mathcal{P}_\mathrm{a}}, \mathrm{PK}^{\mathrm{l}}_{\mathcal{P}_\mathrm{b}}, \mathrm{PK}^{\mathrm{l}}_{\mathcal{P}'_\mathrm{a}}, \mathrm{PK}^{\mathrm{e}}_{\mathcal{P}'_\mathrm{a}}, \mathrm{PK}^{\mathrm{l}}_{\mathcal{P}'_\mathrm{b}}, \mathrm{PK}^{\mathrm{e}}_{\mathcal{P}'_\mathrm{b}})$

Output : $\mathcal{C}'$

The leaving party $\mathcal{P}_a$:

generates a t-degree polynomial $\zeta(\cdot)$, s.t. $\zeta(0) = 0$;

For $i \in [1, n]$, calculates:

$\mathrm{RPs}_{\mathcal{P}_\mathrm{a}}[\mathrm{i}] = \zeta(\mathrm{i})$;

if $i \neq j$: $\{\mathrm{CTR}_\mathrm{i}\}_{\mathcal{P}_\mathrm{a}}[\mathrm{i}] = \mathrm{ENC}_{\mathrm{PKs}^{\mathrm{e}}_{\mathcal{C}}[\mathrm{i}]}(\mathrm{RPs}_{\mathcal{P}_\mathrm{a}}[\mathrm{i}])$;

calculate $\alpha'_j = \alpha_j + \mathrm{RPs}_{\mathcal{P}_\mathrm{a}}[\mathrm{j}]$;

execute $\mathrm{CTS}_{\mathrm{PK}^{\mathrm{l}}_{\mathcal{P}'_\mathrm{a}}, \alpha'_\mathrm{j}} = \mathrm{ENC}_{\mathrm{PK}_{\mathcal{P}'^{\mathrm{s}}_\mathrm{a}}}(\alpha'_\mathrm{j})$;

execute $\mathrm{CTSK}_{\mathrm{PK}^{\mathrm{l}}_{\mathcal{P}'_\mathrm{a}}, \mathrm{PK}^{\mathrm{e}}_{\mathcal{P}_\mathrm{a}}} = \mathrm{ENC}_{\mathrm{PK}^{\mathrm{l}}_{\mathcal{P}'_\mathrm{a}}}(\mathrm{SK}^{\mathrm{e}}_{\mathcal{P}_\mathrm{a}})$;

execute $\Pi_{\mathcal{P}_a} := \text{F-HandOff}^b.\mathrm{Prf}(\chi^{\mathrm{b}}_{\mathcal{P}_\mathrm{a}}, \omega^{\mathrm{b}}_{\mathcal{P}_\mathrm{a}})$; ▶ $\chi^b_{\mathcal{P}}$ and $\omega^b_{\mathcal{P}}$ are defined in Equation 3

$ds_a := (\{\mathrm{CTR}_\mathrm{i}\}_{\mathcal{P}_\mathrm{a}}, \mathrm{CTS}_{\mathrm{PK}^{\mathrm{l}}_{\mathcal{P}'_\mathrm{a}}, \alpha'_\mathrm{j}}, \mathrm{CTSK}_{\mathrm{PK}^{\mathrm{l}}_{\mathcal{P}'_\mathrm{a}}, \mathrm{PK}^{\mathrm{e}}_{\mathcal{P}_\mathrm{a}}})$;

send $\mathrm{LREQ}^b_{\mathcal{P}_a} := (\mathrm{LR}_{\mathcal{P}_\mathrm{a}}, \mathrm{LRID}_{\mathcal{P}_\mathrm{a}}, \mathrm{PK}^{\mathrm{l}}_{\mathcal{P}'_\mathrm{a}}, \mathrm{ds}_\mathrm{a}, \Pi_{\mathcal{P}_\mathrm{a}})$ to blockchain;

The leaving party $\mathcal{P}_b$:

we omit the detail here;

send $\mathrm{LREQ}^b_{\mathcal{P}_b} := (\mathrm{LR}_{\mathcal{P}_\mathrm{b}}, \mathrm{LRID}_{\mathcal{P}_\mathrm{b}}, \mathrm{PK}^{\mathrm{l}}_{\mathcal{P}'_\mathrm{b}}, \mathrm{ds}_\mathrm{b}, \Pi_{\mathcal{P}_\mathrm{b}})$ to blockchain;

send each of CTR and $(\mathrm{CTSK}_{\mathrm{PK}^{\mathrm{l}}_{\mathcal{P}'_\mathrm{a}}, \mathrm{PK}^{\mathrm{e}}_{\mathcal{P}_\mathrm{a}}}, \mathrm{CTS}_{\mathrm{PK}^{\mathrm{l}}_{\mathcal{P}'_\mathrm{a}}, \alpha'_\mathrm{j}})$ to shareholders in the new committee, respectively;

The joined party $\mathcal{P}'_a$:

If received $(\mathrm{CTSK}_{\mathrm{PK}^{\mathrm{l}}_{\mathcal{P}'_\mathrm{a}}, \mathrm{PK}^{\mathrm{e}}_{\mathcal{P}_\mathrm{a}}}, \mathrm{CTS}_{\mathrm{PK}^{\mathrm{l}}_{\mathcal{P}'_\mathrm{a}}, \alpha'_\mathrm{j}})$ from $\mathcal{P}_a$:

Decrypt $\mathrm{CTSK}_{\mathrm{PK}^{\mathrm{l}}_{\mathcal{P}'_\mathrm{a}}, \mathrm{PK}^{\mathrm{e}}_{\mathcal{P}_\mathrm{a}}}$ and $\mathrm{CTS}_{\mathrm{PK}^{\mathrm{l}}_{\mathcal{P}'_\mathrm{a}}, \alpha'_\mathrm{j}}$, and gain $\mathrm{PK}^{e}_{\mathcal{P}_a}$ and α'_j;

generate a t-degree polynomial $\zeta'(\cdot)$, s.t. $\zeta'(0) = 0$;

calculate $\alpha''_j = \alpha'_j + \zeta'(j)$;

For $i \in [1, n], i \neq j$, execute:

$\{\mathrm{CTR}_\mathrm{i}\}_{\mathrm{RPs}_{\mathcal{P}'}}[\mathrm{i}] = \mathrm{ENC}_{\mathrm{PKs}^{\mathrm{r}}_{\mathcal{C}}[\mathrm{i}]}(\gamma'_\mathrm{i})$;

sign $\sigma_{\mathrm{PK}^{l}_{\mathcal{P}'_a}} = \mathrm{SIGN}(\mathrm{SK}^{l}_{\mathcal{P}'}, \mathrm{LRID}_{\mathrm{LR}_{\mathcal{P}}})$;

execute $\Pi_{\mathcal{P}'_a} := \text{F-HandOff}^b.\mathrm{Prf}(\chi^{\mathrm{b}}_{\mathcal{P}'_\mathrm{a}}, \omega^{\mathrm{b}}_{\mathcal{P}'_\mathrm{a}})$; ▶ see $\chi^b_{\mathcal{P}'_a}$ and $\omega^b_{\mathcal{P}'_a}$ in Equation 4

$ds_{a'} := \{\mathrm{CTR}_\mathrm{i}\}_{\mathrm{RPs}_{\mathcal{P}'}}$;

send $\mathrm{JREQ}^b_{\mathcal{P}'_a} := (\mathrm{LR}_{\mathcal{P}'_a}, \mathrm{JRID}_{\mathcal{P}_\mathrm{a}}, \sigma_{\mathrm{PK}^{\mathrm{l}}_{\mathcal{P}'_\mathrm{a}}}, \mathrm{ds}_{\mathrm{a}'}, \Pi_{\mathcal{P}'_\mathrm{a}})$ to blockchain, and send each of $\{\mathrm{CTR}\}_{\mathcal{P}'_\mathrm{a}}$ to shareholders in the new committee, respectively, except for $\mathcal{P}'_a$ himself;

The joined party $\mathcal{P}'_b$:

No Joining request sent by $\mathcal{P}'_b$;

Blockchain Validation Check:

$\mathrm{isValids}_a = (\mathrm{RVC}(\mathrm{LREQ}^b_{\mathcal{P}_a}) \&\& \mathrm{RVC}(\mathrm{JREQ}_{\mathcal{P}'_a}))$;

If $\mathrm{isValids}_a$:

set $\mathcal{C}' = \mathcal{C}/\mathcal{P}_a \cup \mathcal{P}'_a$;

only the F-HandOff between $\mathcal{P}_a$ and $\mathcal{P}'_a$ succeed;

Fig. 6: FPSS - Batched Hand-off

Theorem 1. *A FPSS is secure, if it satisfies the robustness and secrecy.*

We present the proof of Theorem 1 in Appendix A.

4.5 Reconstruction

In an ideal scenario, reconstruction can be fully executed off-chain. When a requester asks for the reconstruction of a secret s, shareholders of the current committee can directly send their newest shares to the requester. Once the requester receives at least $t+1$ honest shares, they can execute Lagrange Interpolation to reconstruct the secret s. The communication cost of this off-chain reconstruction process is proportional to the number of shares being exchanged, which is $O(n)$.

However, in order to avoid receiving malicious shares, we also provide an on-chain approach. When a requester asks for the reconstruction of a secret s, the shareholders encrypt their secret keys using the requester's secret key and record it on-chain. As we assume that the adversary can compromise at most t shareholders in the same committee, at least $t+1$ shareholders must release their secret keys. Once the requester receives at least $t+1$ secret keys from the shareholders of the current committee, they can decrypt at least $t+1$ of the newest honest shares and execute Lagrange Interpolation to reconstruct the secret s. The on-chain approach has an additional cost proportional to the number of shares being stored on-chain, which is also $O(n)$.

4.6 Proof of Correct F-HandOff

This section presents the detailed construction for proving the validity of a F-HandOff request, which allows for the secure F-HandOff of shares between subsets of honest shareholders. We present the proof system for both single and batched F-HandOff requests. To provide an overview of the proof system, We first describe its key components and how they work together to ensure security and correctness.

Overview. The objective of the proposed proof system is to convince any third party that a given encrypted point (i, ct_i) satisfies that $\mathrm{CTS_i} = \mathrm{ENC}_{\mathrm{PK_i}}(\alpha_\mathrm{i})$ and $\alpha_i = \zeta(i)$, where ζ is a private polynomial and PK_i is the encryption key of the i^{th} shareholder. To achieve this objective, the proposed proof system leverages the KZG commitment and Verifiable Encryption (VE) scheme (see Sections 2.3 and 2.4, respectively) to verify that an encrypted value corresponds to a valid evaluation of the private polynomial at a specific point.

By using KZG commitments to commit to the coefficients of the private polynomial and VE scheme to encrypt the values of the polynomial at specific points, the proof system can verify that a given ciphertext corresponds to a valid encryption of the polynomial evaluated at the specified point. The KZG commitment provides a commitment to the polynomial coefficients, which can be used to generate the expected evaluation of the polynomial at the given point. This evaluation is then compared to the decrypted value to ensure correctness, allowing for efficient and secure verification of encrypted values without revealing the private polynomial or the specific evaluation point.

Single F-HandOff.

We specify the proving system of the single F-HandOff relations (see Equation 1 and Equation 2).

Let $\mathbb{G}$ be a bilinear group of prime order p and generator g for KZG commitment scheme. Let d be a trapdoor generated by a trusted third party. This system setup is generated in the SETUP phase (see Section 4.1).

For the leaving party.

1. **KZG Proving Phase**:

First, the leaving party generates a KZG proof of knowledge of the t polynomial $\zeta(x)$. The KZG proof consists of a set of evaluation proofs of the polynomial at specific points in the field. The KZG commitment scheme used in the proof provides a commitment to the polynomial coefficients, which can be computed as $\text{KZG.CMT}_{\zeta} = g^{\zeta(d)}$. The KZG commitment KZG.CMT_{ζ} can be used to verify that $\zeta(i) = \gamma_i$ without revealing the polynomial. Therefore, proving that $\zeta(0) = 0$ simply follows from the KZG commitment scheme.

However, for the evaluation of $\zeta(i)$ where $i \in [1,n]$, the leaving party must also provide a commitment to the evaluation point. This is to ensure that the evaluation remains hidden and secure. The commitment of $\zeta(i)$ can then be used in the verification phase to verify that the re-randomized share satisfies the required equation. By providing these commitments, the leaving party can ensure the correctness and security of the reshare process.

Second, to prove that the re-randomized share ${\alpha'}_j$ assigned to the joining party satisfies ${\alpha'}_j = \alpha_j + \zeta(j)$, the leaving party must provide a proof. Since KZG commitments have homomorphic properties, if the leaving party's share α_j and the re-randomized share ${\alpha'}_j$ satisfy ${\alpha'}_j = \alpha_j + \zeta(j)$, the commitment can be computed as:

$$\text{KZG.CMT}_{{\alpha'}_i} = \text{KZG.CMT}_{\alpha_i} \cdot \text{KZG.CMT}_{\zeta(j)}.$$

Since we require on-chain recording of the commitment for each updated share. Verification can be done by comparing the commitment $\text{KZG.CMT}_{\alpha_i}$ with the on-chain record.

2. **Encryption Phase**: Next, the leaving party encrypts the shares using a verifiable encryption (VE) scheme, which provides both confidentiality and authenticity for encrypted shares. The encryption proofs for both $\gamma_i = f(i)$ and ${\alpha'}_j$ are identical. So we use γ_i as an example in the following description. Specifically, the leaving party encrypts each of the points $ct_i = \text{ENC}_{\text{PK}_i}(\gamma_i)$, and creates the zero-knowledge proof π_i^{VE} that proves the $\text{KZG.CMT}_{\zeta(\cdot)}$ is a commitment to the polynomial $\zeta(\cdot)$, such that $\gamma_i = \zeta(i)$ without revealing γ_i or $\zeta(\cdot)$.

3. **Verification Phase**: The verifier uses the KZG and VE schemes to verify that ct_i is the encryption of $\gamma_i = \zeta(i)$ as follows:

- KZG Verification: The verifier checks the KZG proof generated in the first step to ensure that the commitments g^{γ_i} correspond to the evaluations $\zeta(i) = \gamma_i$ of the polynomial at the specific points i.
- VE Verification: The verifier checks the validity of the encryption by using the commitments g^{γ_i}, ciphertext ct_i and π_i^{VE}.

For more details and implementation considerations, refer to the preliminary section (Section 2.3 and 2.4) and the original KZG paper [16] and the Camenisch-Shoup verifiable encryption scheme paper [11].

By combining KZG and VE, we can verify that a ciphertext ct_i is the encryption of a point $\zeta(i)$ without revealing the private polynomial $\zeta(\cdot)$ and the evaluation point $\zeta(i)$ to the verifier.

For the joining party.

The proof required by the joining party is simpler than the leaving party's proof, as the joining party only needs to prove that the polynomial they generated satisfies $\zeta'(0) = 0$, and the evaluation of each point $\gamma'_i = \zeta'(i)$, where $i \in [1,n]$ and $i \neq j$. Since the description of the proof generation process is already covered by the leaving party's proof above, we will omit it here to avoid redundancy.

Batched F-HandOff

The proof system for batched leaving is similar to that of Single F-HandOff, with an additional step for verifying the ephemeral secret key. Since the leaving party needs to release their ephemeral secret key to the joining party for accessing the points assigned to them, the leaving party encrypts their ephemeral secret key using the joining party's long term secret key and generates an encryption proof for the correctness of their ephemeral secret key. This proof can be generated using the verifiable encryption scheme directly. By including this step, the proof system ensures that the joining party has access to the correct ephemeral secret key, allowing them to securely access the point assigned to the leaving party from among concurrent leaving requests.

For the batched joining proof, it is identical to the leaving joining proof, we omit here for avoid redundancy.

5 Performance

This section introduces our proof-of-concept implementation of FPSS and the corresponding evaluation analysis.

5.1 Implementation

Our implementation is based on Churp [25] and the Coinbase's advanced cryptography library, kryptology [4], GNU multiple precision arithmetic Library (GMP) [6], and the Pairing-Based Cryptography Library (PBC) [5] for the cryptographic primitives, and gRPC [3] for network infrastructure.

We use the following setting same as Churp. For polynomial arithmetic, we used the polynomial ring $\mathbb{F}_p[x]$ for 256-bit prime p. For the KZG commitment scheme, we used type A pairing on an elliptic curve $y^2 = x^3 + x$ over $\mathbb{F}_p$ for a 512-bit q. The order of the EC group is also p. We also use SHA-256 for hashing. For the public key encryption scheme, we use the group $\mathbb{Z}_n$ for 2048-bit composite group order n. For the verifiable encryption scheme, we used a cyclic group $\mathbb{G}_p$ of the same prime order p.

FPSS can be deployed on either a permissioned or a permissionless blockchain. We use a blockchain simulator as Churp did. That is, we abstract away the specific choice and simulate one using a trusted node. It will incur additional latency when deploying FPSS on a Blockchain. Our performance analysis is based on the batched F-HandOff, since it can also be used for a single shareholder update.

5.2 Evaluation

Setup. Our experiments are run on a macOS with processor 2.6 GHz 6-Core Intel Core i7 and memory 16GB 2400 MHz DDR4.

On-chain communication complexity. The definition of on-chain cost is adopted from CHURP [18]. That is the number of bits written to the blockchain. FPSS employs blockchain validators (a.k.a miners) for verifying the on-chain requests. As FPSS requires all ciphertexts to be recorded on-chain, which is about $1.5 * 2 * n$ kb on-chain per F-HandOff, where n is the size of the committee. FPSS can be deployed on either permissionless (such as Filecoin [2]) and permissioned (such as Corda [1]) blockchain. While choosing permissioned blockchain can reduce the on-chain cost.

Off-chain data size. F-HandOff has two rounds, which incurs an $O(m)$ communication complexity on-chain and an $O(n * m)$ communication complexity off-chain, where n is the committee size and m is the number of leaving shareholders. The off-chain data transmitted per leaving node includes: a KZG commitment of a polynomial, n KZG proof (consisting of n commitment of polynomial points and the corresponding witnesses) and $n+1$ ciphertext (consisting of n encrypted polynomial points and the corresponding proof, and an ciphertext on ephemeral secret key and its proof). The off-chain data transmitted per joining node includes: a KZG commitment of polynomial, $n-1$ KZG proof (consisting of $n-1$ commitment of polynomial points and the corresponding witnesses) and $n-1$ ciphertext on shares to each shareholders in the new committee, except for himself.

Fig. 7 shows that in a large committee, FPSS reduces off-chain communication for shareholders to update their shares. Thus, for the update of a single shareholder, the data required to be transmitted off-chain is about $6n+6$ kB. As a result, when running a 1000-shareholder committee, it takes about 6 MB of data for FPSS to be transmitted off-chain and 226 MB for CHURP for each single shareholder update, which is reduced to about $37\times$ off-chain data transmission. While FPSS is not always more efficient than CHURP. For example, when the number of shareholders needing to update their shares is greater than 100 in a 200-shareholder committee, CHURP requires less off-chain data to be transmitted than FPSS. In conclusion, a committee with a majority of members who change frequently could use CHURP to update the shares of the committee. If we consider the scenario where not more than half of the committee members may leave at any time, FPSS would be the better choice to reduce off-chain data transmission.

Network transmission latency. For a single F-HandOff, the leaving party will broadcast a leaving request to blockchain and the encrypted shares to shareholders, respectively, in the new committee; as soon as the joining party receives a share from the leaving party, he will broadcast a joining request to the blockchain and the encrypted shares to the entire new committee except for himself. According to our Blockchain model in Section 3.1, the underlying blockchain satisfies the *liveness* property. Namely, if a request is valid, it will eventually be recorded on-chain. In addition, F-HandOff does not require acknowledgment messages to be responded from all receivers to the sender (the leaving or joining party). Let D_L be the transmission latency, which is the time that an encrypted share (about 1.5 kb) is sent from the leaving party to the joining party, and let D_J be the latency, which is the time that the joining request (about $3n+3$ kb) is sent from the joining party to the blockchain. In the optimal scenario, the data

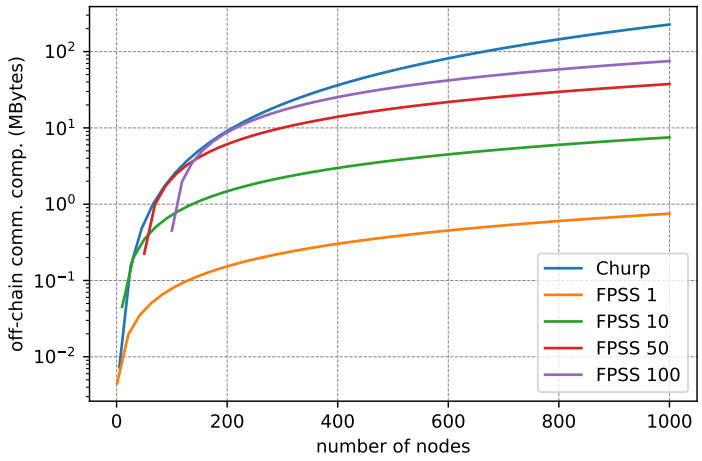

Fig. 7: Off-Chain Communication Complexity for FPSS and Churp. We depict four lines for FPSS with the number of shareholders needing to update their shares being 1, 10, 50, and 100, respectively.

transmission latency of a single F-HandOff is D_L+D_J. Since each F-HandOff is independent of other concurrent ones, and multiple concurrent F-HandOffs can be executed in parallel. If every participant uses the Ethernet cabling, the LAN speeds are typically 1000 mbps and WAN are 150 mbps [7]. For a committee with 100 shareholders, the data transmission latency is about 0.3 ms to transmit 0.3 MB data in a LAN setting and about 2 ms in a WAN setting.

Microbenchmarks. We also evaluate the cost of core verifications in our implementation of the KZG polynomial commitment scheme and the verifiable encryption scheme. The results are presented in Table 3. For the KZG scheme, the setup refers to the time required to generate a public key for a polynomial with a given degree. For the VE scheme, the setup refers to the time of generating the system parameter n=pq, where p and q are safe primes. They are all the one time cost across all committed polynomials.

Table 3: Cost of KZG and VE Scheme

	Degree		
	6	26	49
(KZG) Setup	0.06 s	0.2 s	0.4 s
(KZG) Poly Commit	0.06 ms	0.3 ms	0.7 ms
(KZG) Point Commit	0.02 ms	0.09 ms	0.1 ms
(KZG) Point Eval	3.7 ms	3.4 ms	3.5 ms

	Num of Nodes		
	10	50	100
(VE) Setup	0.02 s		
(VE) EncryptAndProve	0.98 s	4.7 s	9.5 s
(VE) Cipher Verify	0.72 s	3.5 s	7.0 s

6 Related Work

Proactive secret share schemes have been studied by plenty of works [8, 10, 15, 27, 28, 28, 29]. They provide either a stronger security model or a more efficient algorithm for resharing secrets from the old committee to the new one.

Secret share schemes on Blockchain. Recent years, the Blockchain rises a lot of attention for it permissionless, public accessibility, and immutability functionalities. Many researches would like to adopted the blockchain as the underlying infrastructure to improve their techniques, including some dynamic secret share schemes [9, 14, 18]. For example, instead of sending a message to a receiver directly, they also consider to use blockchain as the communication channel. That is, once a participant in the scheme posts his message to the blockchain, everyone in the network received it. Or some works [9, 14] use blockchain for player replacement. Benhamouda et al. [9] uses PoS blockchain to construct the committee selection scheme and the anonymous public key encryption primitive. Each member of the new committee is anonymous to the old committee. Their player selection will be a prepare phase for hand-off, which incurs on-chain cost. This scheme can tolerate at most $1/4$ of malicious parties under the mobile adversary setting. Goyal et al. [14] enables the dynamic secret share scheme to be deployed on either PoS or PoW blockchain and can tolerant of at most $1/2$ malicious parties under the adaptive adversary model. This scheme requires that both all the old and new committees to participate in the hand-off phase. CHURP [18] uses the bivariant polynomial to do share secret, and they introduce a new proactivization protocol with also $1/2$ resilience. They prefer to use blockchain as the communication channel, or off-chain as the alternative due to the reason that written to blockchain is expensive. Also, all participants in their protocol should participant in a hand-off phase. CALYPSO [17] introduces a data management system, such that data owner can manage their data access control right via such system. Their system built upon the secret share scheme, while among a static committee.

Highlight our advancements. Compared to other schemes in Table 1, the lowest communication cost of the listed schemes is $O(nt)$ off-chain and $O(n)$ on-chain, even if the hand-off is requested by a single shareholder. FPSS reduces this cost from $O(nt)$ to $O(mn)$ off-chain or $O(n)$ to $O(m)$ on-chain, where m is the number of shareholders leaving the old committee. For each single shareholder update, the off-chain communication cost becomes $O(n)$, and the on-chain cost becomes $O(1)$.

7 Conclusion

PSS scheme periodically execute hand-off involving the entire committee. FPSS could be used as a general framework that adds F-HandOff on top of PSS schemes [9, 14, 18] to be executed on-demand. While this requires more in-depth research in the future and is not discussed in this paper.

We consider such a scenario that shareholders may leave at any time in some distributed systems, such as permissionless blockchains. The existing PSS schemes do not offer such flexibility, which requires the entire committee to be interactively involved in the share update. We propose FPSS to enable the flexible hand-off, which involves only the pair of leaving and joining shareholders' involvement.

References

1. Corda. https://www.r3.com/products/corda/
2. Filecoin. https://filecoin.io/
3. The high performance remote procedure call (rpc) framework. https://pkg.go.dev/google.golang.org/grpc
4. Kryptology: Coinbase's open source cryptography library. https://github.com/coinbase/kryptology/tree/master/pkg/verenc/camshoup
5. The pairing-based cryptography library. https://github.com/ncw/gmp
6. the gnu multiprecision library (gmp). https://github.com/Nik-U/pbc
7. WAN vs. LAN: What Is the Difference? https://www.truecable.com/blogs/cable-academy/wan-vs-lan
8. Abraham, I., Jovanovic, P., Maller, M., Meiklejohn, S., Stern, G.: Bingo: Adaptively secure packed asynchronous verifiable secret sharing and asynchronous distributed key generation. Cryptology ePrint Archive, Paper 2022/1759 (2022), https://eprint.iacr.org/2022/1759
9. Benhamouda, F., Gentry, C., Gorbunov, S., Halevi, S., Krawczyk, H., Lin, C., Rabin, T., Reyzin, L.: Can a public blockchain keep a secret? In: TCC (1). Lecture Notes in Computer Science, Springer (2020)
10. Cachin, C., Kursawe, K., Lysyanskaya, A., Strobl, R.: Asynchronous verifiable secret sharing and proactive cryptosystems (2002)
11. Camenisch, J., Shoup, V.: Practical verifiable encryption and decryption of discrete logarithms. In: CRYPTO. Lecture Notes in Computer Science, vol. 2729, pp. 126–144. Springer (2003)
12. Eldefrawy, K., Ostrovsky, R., Park, S., Yung, M.: Proactive secure multiparty computation with a dishonest majority. In: SCN. Lecture Notes in Computer Science, vol. 11035, pp. 200–215. Springer (2018)
13. Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Secure distributed key generation for discrete-log based cryptosystems. In: EUROCRYPT. Lecture Notes in Computer Science, Springer (1999)
14. Goyal, V., Kothapalli, A., Masserova, E., Parno, B., Song, Y.: Storing and retrieving secrets on a blockchain. In: Public Key Cryptography (1). Lecture Notes in Computer Science, Springer (2022)
15. Herzberg, A., Jarecki, S., Krawczyk, H., Yung, M.: Proactive secret sharing or: How to cope with perpetual leakage. In: CRYPTO. Lecture Notes in Computer Science, vol. 963, pp. 339–352. Springer (1995)
16. Kate, A., Zaverucha, G.M., Goldberg, I.: Constant-size commitments to polynomials and their applications. In: ASIACRYPT. Lecture Notes in Computer Science, vol. 6477, pp. 177–194. Springer (2010)
17. Kokoris-Kogias, E., Alp, E.C., Gasser, L., Jovanovic, P., Syta, E., Ford, B.: CALYPSO: private data management for decentralized ledgers. Proc. VLDB Endow. **14**(4), 586–599 (2020)
18. Maram, S.K.D., Zhang, F., Wang, L., Low, A., Zhang, Y., Juels, A., Song, D.: CHURP: dynamic-committee proactive secret sharing. In: CCS. pp. 2369–2386. ACM (2019)
19. Qin, X., Pan, S., Mirzaei, A., Sui, Z., Ersoy, O., Sakzad, A., Esgin, M.F., Liu, J.K., Yu, J., Yuen, T.H.: Blindhub: Bitcoin-compatible privacy-preserving payment channel hubs supporting variable amounts. IEEE S&P (2023)
20. Rabin, T., Ben-Or, M.: Verifiable secret sharing and multiparty protocols with honest majority (extended abstract). In: STOC. ACM (1989)
21. Schultz, D.A., Liskov, B., Liskov, M.D.: MPSS: mobile proactive secret sharing. ACM Trans. Inf. Syst. Secur. **13**(4), 34:1–34:32 (2010)

22. Shamir, A.: How to share a secret. Commun. ACM (1979)
23. Sui, Z., Liu, J.K., Yu, J., Au, M.H., Liu, J.: Auxchannel: Enabling efficient bi-directional channel for scriptless blockchains. In: AsiaCCS. pp. 138–152. ACM (2022)
24. Sui, Z., Liu, J.K., Yu, J., Qin, X.: Monet: A fast payment channel network for scriptless cryptocurrency monero. In: ICDCS. IEEE (2022)
25. Churp Team: Churp. `https://github.com/CHURPTeam/CHURP` (2019)
26. Traverso, G.: Long-Term Confidential Secret Sharing-Based Distributed Storage Systems. Ph.D. thesis, TU Darmstadt, Germany (2019)
27. Vassantlal, R., Alchieri, E., Ferreira, B., Bessani, A.: COBRA: dynamic proactive secret sharing for confidential BFT services. In: IEEE Symposium on Security and Privacy. pp. 1335–1353. IEEE (2022)
28. Yan, Y., Xia, Y., Devadas, S.: Shanrang: Fully asynchronous proactive secret sharing with dynamic committees. ePrint Arch. (2022)
29. Yurek, T., Xiang, Z., Xia, Y., Miller, A.: Long live the honey badger: Robust asynchronous DPSS and its applications. IACR Cryptol. ePrint Arch. p. 971 (2022)

A Security Discussion

This section provides the proof sketch of the security properties, *robustness* and *secrecy*. that FPSS required.

A.1 FPSS Robustness Proof

We adopt the proof idea of Robustness from the work by Goyal et al. [14]. For the single hand-off of FPSS, the Robustness can be proved by Lemma 1, 2, 3.

Lemma 1 (Secret Integrity). *Given a secret s and its commitment* cmt_s*, with overwhelming probability, once s is successfully distributed among a committee $\mathcal{C}$ by uploading a distribution transaction on-chain, the secret s will never be modified for each single hand-off.*

Proof. Since we assume the underlying blockchain satisfies *persistence* (Section 2.1), a transaction can be recorded on-chain only if it is valid. Thus, if a distributed transaction is recorded on-chain, each shareholder $\mathcal{P}$ in the committee $\mathcal{C}$ owes a valid share $\alpha_i = f(i)$ of the secret s, where $f(0) = s$.

Assuming that $\{\alpha_i\}_{i\in[1,n]}$ is a set of shares owing by shareholders in the committee $\mathcal{C}$ respectively. Once a single hand-off is executed successfully, namely a pair of leaving and joining requests are recorded on-chain, the set of shares becomes $\{\alpha'_i\}_{i\in[1,n]}$. where $\alpha'_i = \alpha_i + \zeta(i) + \zeta'(i)$, $\zeta(0) = 0$, and $\zeta'(0) = 0$. Thus, the value $s' = f(0) + \zeta(0) + \zeta'(0)$ equals to the original secret s committed in cmt_s.

Lemma 2 (Honest Share Integrity). *Given a secret s and its commitment* cmt_s*, with overwhelming probability, after each of the single F-HandOff, each honest party in the new committee $\mathcal{C}'$ obtain the correct share of the given s. For any subset $\mathcal{U} \subset \mathcal{C}$, the share provided by parties in $\mathcal{U}$ can only be reconstructed to either the given secret s or $\perp$, where $\perp$ denotes a failure in the reconstruction. In particular, if $\mathcal{U}$ contains at least $t+1$ honest parties, the shares provided by parties in $\mathcal{U}$ can always be reconstructed (by anyone with a valid secret reconstruction request) to the given secret s.*

Proof. Since we assume that the underlying blockchain satisfies the persistence properties (see Section 2.1), a transaction can be recorded on-chain only if it is valid (accepted by the honest party).

For the newly joined party $\mathcal{C}'_j$, his receives a share $(\alpha_j + \zeta(j))$, where α_j is a valid share of the secret s in the old committee and $\zeta(0) = 0$. He will compute the new share α'_j by adding $\zeta'(j)$, where $\zeta'(0) = 0$.

For each of the remaining parties $\mathcal{C}'_i$, where $i \in [1,n]/j$ and $\mathcal{C}_i = \mathcal{C}'_i$, he will receive two points $\zeta(i)$ and $\zeta'(i)$. Thus, their updated share is $\alpha'_i = \alpha_i + \zeta(i) + \zeta'(i)$, where α_i is a valid share of the old committee.

If the leaving or joining request is recorded on-chain, the correctness of these corresponding shares is verified by blockchain validators.

Since there is at least $t+1$ honest shareholders in the current committee $\mathcal{U}$, the secret s committed in cmt_s can always be reconstructed by $t+1$ valid shares from the current committee.

Lemma 3 (Robustness of Single Hand-Off). *Given a secret s and its commitment* cmt_s*, with overwhelming probability, the valid requester in* Reconstruction can always reconstruct the s committed in cmt_s.

Proof. From the above two Lemmas, Lemma 1 and Lemma 2, we conclude that the current committee contains at least $t+1$ honest parties with $t+1$ valid shares respectively. So that any honest party with valid secret request can reconstruct the secret from the current committee. Thus, the lemma is proved.

Lemma 4 (Robustness of Batched Hand-Off). *Given a secret s and its commitment* cmt_s*, with overwhelming probability, the valid requester in* Reconstruction can always reconstruct the s committed in cmt_s under the Batched Hand-Off case.

Proof. The proof of batched hand-off is similar to single hand-off, we omitted here.

A.2 FPSS Secrecy Proof

The proof of Secrecy is the same for both the single hand-off and batched hand-off.

Lemma 5 (Secrecy). *If the adversary can corrupt at most t shareholders in the same committee, FPSS satisfies the secrecy property.*

Proof. Let $\mathcal{S}$ be a simulator and $\mathcal{A}$ be the probabilistic polynomial-time (PPT) adaptive adversary. The goal of adversary $\mathcal{A}$ is to output a guess b the distributed secret s. The simulator's goal is to output a guess b' that is indistinguishable from the adversary's guess b. Let $\mathcal{L}(\Delta)$ be the functionality of Blockchain. Since robustness proof already proves that the secret can always been recovered after several successful F-HandOff, we do not consider the verification of F-HandOff requests in the secrecy proof again. We simplify the Blockchain Functionality (see Figure 8 [19]) required in following proof.

(Simplified) Ideal Functionality $\mathcal{L}(\Delta)$

This functionality stores all the published transactions in the ledger. Once a valid transaction is posted, it takes at most Δ rounds to be added in the ledger.

Post a transaction: Upon $(\text{post}, \text{tx}) \overset{\tau_0}{\leftarrow} P$, publish tx on the ledger $\mathcal{L}$ in round $\tau_1 \leq \tau_0 + \Delta$.

Fig. 8: Simplified Blockchain Functionality

All transactions published in the ledger $\mathcal{L}$, can be publicly accessed, $\mathcal{A}$ can access all encrypted shares on-chain. Let $\hat{\varepsilon}$ be the advantage of guessing the secret s in the following FPSS-Secrecy-Game. We will show that the simulator $\mathcal{S}$ has the same advantage in guessing the secret s correctly as $\mathcal{A}$. The FPSS-Secrecy-Game is defined as follows:
Setup Game:

- The challenger runs the setup algorithm at time τ to generate the public parameters and shares them with all parties participating in this protocol;
- The challenger chooses a secret value s, and divides them into n shares, such that $[\alpha_1, \ldots, \alpha_n]$;
- The challenger updates all shareholders' secret and private key pairs at time τ, and encrypts these shares by using each shareholder's public key and distributes these encrypted shares to these shareholders, respectively. So that any t+1 shares can be used to reconstruct s.
- The challenger records the commitment of s and all encrypted $[\alpha_1, \ldots, \alpha_n]$ on $\mathcal{L}$.

Single F-HandOff Game

- The adversary $\mathcal{A}$ chooses a set of up to t corrupted parties within Δ_c time interval and obtains their plaintext shares of s at time τ.
- At the time τ, the simulator $\mathcal{S}$ randomly chooses a shareholder $\mathcal{P}_i$, where $i \in [1, n]$, to be the leaving party. There are two cases:
  1. if $\mathcal{P}_i$ is uncompromised:
     - $\mathcal{S}$ generates a polynomial $\zeta(x)$ of degree t such that $\zeta(0) = 0$, and chooses n points on the polynomial, such that $\zeta(1), \ldots, \zeta(n)$;
     - $\mathcal{S}$ updates $\mathcal{P}_i$'s share by $\alpha'_i = \alpha_i + \zeta(i)$, where α_i is $\mathcal{P}_i$'s original share.
     - $\mathcal{S}$ encrypts α'_i by using $\mathcal{P}'$'s encryption key and $n-1$ points, $\zeta(j)$, where $j \in [1, n], j \neq i$, by using other shareholders' encryption keys, respectively.
     - $\mathcal{S}$ sends a transaction including the encrypted share α'_i and these encrypted points $\zeta(j)$, where $j \in [1, n], j \neq i$, to $\mathcal{L}$.
  2. if $\mathcal{P}_i$ is compromised:
     - $\mathcal{S}$ generates a polynomial $\zeta(x)$ of degree t such that $\zeta(0) = 0$, and chooses n points on the polynomial, such that $\zeta(1), \ldots, \zeta(n)$.
     - $\mathcal{S}$ updates $\mathcal{P}_i$'s share by computing $\alpha'_i = \alpha_i + \zeta(i)$.
     - $\mathcal{S}$ encrypts α'_i by using $\mathcal{P}'$'s encryption key and $n-1$ points, $\zeta(j)$, where $j \in [1, n], j \neq i$, by using other shareholders' encryption key, respectively.

- $\mathcal{S}$ sends a transaction including the encrypted share α'_i and these encrypted points $\zeta(j)$, where $j \in [1,n], j \neq i$, to $\mathcal{L}$.
- $\mathcal{S}$ sends the plaintext α'_i to $\mathcal{A}$.

– Each shareholder $\mathcal{P}_j$, where $j \neq i$, receives the corresponding point $\zeta(j)$, and updates his share by computing $\alpha'_j = \alpha_j + f(j)$;
– $\mathcal{P}'_i$ receives the updated share α'_i from $\mathcal{P}_i$. There are three cases:
  1. if $\mathcal{P}'_i$ is uncompromised:
     - $\mathcal{S}$ generates a polynomial $\zeta'(x)$ of degree t such that $\zeta'(0) = 0$, and chooses n points on the polynomial, such that $\zeta'(1), \ldots, \zeta'(n)$. We do not distinguish whether $\mathcal{P}_i$ is compromised or not under this case, since $\mathcal{P}_i$ does not affect $\mathcal{P}'_i$'s behaviour.
     - $\mathcal{S}$ updates $\mathcal{P}'$'s share by computing $\alpha''_i = \alpha'_i + \zeta'(i)$.
     - $\mathcal{S}$ encrypts α''_i by using $\mathcal{P}'$'s encryption key and $n-1$ points, $\zeta'(j)$, where $j \in [1,n], j \neq i$, by using other shareholders' encryption keys, respectively.
     - $\mathcal{S}$ sends a transaction including the encrypted share α''_i and these encrypted points $\zeta'(j)$, where $j \in [1,n], j \neq i$, to $\mathcal{L}$.
  2. if $\mathcal{P}'_i$ is compromised, but $\mathcal{P}_i$ is uncompromised:
     - $\mathcal{S}$ generates a polynomial $\zeta'(x)$ of degree t such that $\zeta'(0) = 0$, and chooses n points on the polynomial, such that $\zeta'(1), \ldots, \zeta'(n)$. We do not distinguish whether $\mathcal{P}_i$ is compromised or not under this case, since $\mathcal{P}_i$ does not affect $\mathcal{P}'_i$'s behaviour.
     - $\mathcal{S}$ updates $\mathcal{P}'$'s share by computing $\alpha''_i = \alpha'_i + \zeta'(i)$.
     - $\mathcal{S}$ sends α''_i to $\mathcal{A}$.

     If $\mathcal{A}$ already corrupts a set of t shareholders, $\mathcal{P}_i$ is not included, the simulator $\mathcal{S}$ halts. That is once $\mathcal{P}'$ is corrupted, $\mathcal{A}$ will have $t+1$ shares in the new committee. While this case is contradict to our assumption, such that $\mathcal{A}$ can corrupt at most t shareholders for every Δ_c interval.
  3. if both $\mathcal{P}'_i$ and $\mathcal{P}_i$ are all compromised:
     - $\mathcal{S}$ generates a polynomial $\zeta'(x)$ of degree t such that $\zeta'(0) = 0$, and chooses n points on the polynomial, such that $\zeta'(1), \ldots, \zeta'(n)$.
     - $\mathcal{S}$ updates $\mathcal{P}'$'s share by computing $\alpha''_i = \alpha'_i + \zeta'(i)$.
     - $\mathcal{S}$ encrypts α''_i by using $\mathcal{P}'$'s encryption key and $n-1$ points, $\zeta'(j)$, where $j \in [1,n], j \neq i$, by using other shareholders' encryption keys, respectively.
     - $\mathcal{S}$ sends a transaction including the encrypted share α''_i and these encrypted points $\zeta'(j)$, where $j \in [1,n], j \neq i$, to $\mathcal{L}$.
     - $\mathcal{S}$ sends α''_i to $\mathcal{A}$.

– $\mathcal{S}$ sends the corresponding points $\zeta'(j)$ to each shareholder $\mathcal{P}_j$ where $j \neq i$, who updates their share by computing $\alpha''_j = \alpha'_j + \zeta'(j)$.
– $\mathcal{A}$ continues to corrupt additional shareholders (if any) and receives their shares.

Guess Game. Adversary $\mathcal{A}$ outputs a guess for the value of the secret s, and the simulator outputs the same guess.
Proof of Indistinguishability. We claim that the simulator $\mathcal{S}$ is indistinguishable from the actual game.

Case 1: Both $\mathcal{P}_i$ and $\mathcal{P}'_i$ is not corrupted. In this case, $\mathcal{S}$ proceeds as in the proof for the case when $\mathcal{P}_i$ is not compromised and $\mathcal{P}'_i$ is not compromised, which has already been shown to be indistinguishable from the actual game.

Case 2&3: $\mathcal{P}_i$ is corrupted and $\mathcal{P}'_i$ is corrupted, or $\mathcal{P}_i$ is corrupted but $\mathcal{P}'_i$ is not corrupted. In this cases, $\mathcal{S}$ proceeds as described above. Since $\mathcal{S}$ chooses at most t shareholders to corrupt uniformly at random and receives their shares, the shares held by the adversary $\mathcal{A}$ are indistinguishable from those in the actual game. Moreover, the shares distributed to the shareholders not chosen to perform the hand-off are also indistinguishable from the actual game, since they are generated using the same polynomial as in the actual game. The simulator $\mathcal{S}$ then proceeds to generate a new share for the new shareholder $\mathcal{P}'_i$ using the same polynomial as in the actual game.

Case 4: $\mathcal{P}_i$ is not corrupted, but $\mathcal{P}'_i$ is corrupted. In this case, if $\mathcal{S}$ already chooses t shareholders except for $\mathcal{P}_i$ to corrupt uniformly at random and receives their shares, the program halts. Otherwise, since all corrupted shareholders including $\mathcal{P}'_i$ are chosen uniformly at random, the shares held by the adversary $\mathcal{A}$ are indistinguishable from those in the actual game.

Since shareholders update their secret and public key pairs every Δ_c time interval and the encryption scheme used in the protocol is semantically secure, the encrypted shares of the secret s and points of the polynomials, $\zeta(\cdot)$ and $\zeta'(\cdot)$, recorded on $\mathcal{L}$ are indistinguishable from random strings. Therefore, the updated shares α'_i and α''_i received by $\mathcal{P}'_i$ is indistinguishable from a valid share of a random string.

Similarly, the points $\zeta'(j)$ received by the shareholders not chosen to perform the hand-off are indistinguishable from random strings, since they are generated using the same polynomial as in the actual game. Therefore, the updated share α''_j held by each of these shareholders is indistinguishable from a valid share of a random string.

Hence, the shares held by the adversary $\mathcal{A}$ are indistinguishable from those in the actual game, and the guess output by $\mathcal{S}$ is the same as the guess output by $\mathcal{A}$. Therefore, the simulator $\mathcal{S}$ is indistinguishable from the actual game.

Let $\mathcal{F}_{sec}$ be the FPSS-Secrecy-Game. Since the advantage of the adversary $\mathcal{A}$ in the $\mathcal{F}_{sec}$ is at most $\hat{\varepsilon}$, the simulator $\mathcal{S}$ is indistinguishable from the actual game, we have:

$$|Pr[\mathcal{F}_{sec}(\mathcal{A})] = 1| - Pr[\mathcal{F}_{sec}(\mathcal{S})] = 1| \leq \hat{\varepsilon}$$

, which implies that the FPSS protocol is secure against an adaptive adversary that can corrupt up to t shareholders.