Proactive Secret Sharing in Dynamic Environments

Andrew Low
Deepak Krishna
Fan Zhang
Lun Wang
Yupeng Zhang
Ari Juels
Dawn Song

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2019-62
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2019/EECS-2019-62.html

May 17, 2019

Acknowledgement

This project is collaborative work with Deepak Krishna, Fan Zhang, Lun Wang, Yupeng Zhang, Ari Juels, and Dawn Song. I contributed significantly to this project as well as a data capsules project over the course of 2018 and 2019.

I would like to thank my advisor, Dawn Song, for her guidance during my undergraduate and graduate careers. I would also like to thank Joey Gonzalez for his advice and teaching on systems and AI. I would also like to thank Vern Paxson, Paul Pearce, Austin Murdock, John Kuriyan, and Deepti Karandur for their advice and mentorship during my research journey. Lastly, I would like to thank Rohith Krishna, Steven Sum, Kevin Lian, and all of VCG for their unwavering support.

Proactive Secret Sharing in Dynamic Environments

by

Andrew Low

A thesis submitted in partial satisfaction of the

requirements for the degree of

Master in Science

in

Electrical Engineering and Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Dawn Song, Chair
Professor Joey Gonzalez

Spring 2019

Abstract

Proactive Secret Sharing in Dynamic Environments

by

Andrew Low

Master in Science in Electrical Engineering and Computer Science

University of California, Berkeley

Professor Dawn Song, Chair

We introduce CHURP (CHUrn-Robust Proactive secret sharing). CHURP enables secure secret sharing in dynamic settings, where the committee of nodes storing a secret changes over time. Designed for blockchains, CHURP has lower communication complexity than previous schemes: $O(n)$ on-chain and $O(n^2)$ off-chain in the optimistic case of no node failures.

CHURP includes several technical innovations: An efficient new proactivization scheme, a technique (using asymmetric bivariate polynomials) for efficiently changing secret-sharing thresholds, and a hedge against setup failures in an efficient polynomial commitment scheme. We report on the functionality and implementation of CHURP, and present performance improvements.

To Chin-Chai, Amy, and Michelle

Contents

Acknowledgments

This project is collaborative work with Deepak Krishna, Fan Zhang, Lun Wang, Yupeng Zhang, Ari Juels, and Dawn Song. I contributed significantly to this project as well as a data capsules project over the course of 2018 and 2019.

I would like to thank my advisor, Dawn Song, for her guidance during my undergraduate and graduate careers. I would also like to thank Joey Gonzalez for his advice and teaching on systems and AI. I would also like to thank Vern Paxson, Paul Pearce, Austin Murdock, John Kuriyan, and Deepti Karandur for their advice and mentorship during my research journey. Lastly, I would like to thank Rohith Krishna, Steven Sum, Kevin Lian, and all of VCG for their unwavering support.

Chapter 1

Introduction

Secure private key storage is a fundamental challenge in blockchains and decentralized systems. Private keys control a plethora of valuable digital assets, from cryptocurrencies to online identities. Private key loss often leads to serious and irreversible consequences.

However, users struggle to store their private keys reliably and securely. Users often store their keys on local devices, where they are vulnerable to phishing attacks, malware, and device loss or failure. Bitcoin private keys are also a valuable target to malicious actors, which incentivizes further attacks. Consequently, an estimated four million Bitcoin have been lost forever due to lost or stolen keys [32].

Many users thus choose to store their cryptocurrency with exchanges such as Coinbase, which holds an estimated 10% of all circulating Bitcoin [1]. However, the centralized nature of these systems contradicts the decentralization that defines blockchain systems.

Secret sharing provides an alternative to centralized private key stores. In a secret sharing scheme, a committee of n nodes hold shares of a secret s. One well-known implementation is Shamir's secret sharing scheme, in which a secret is encoded as the y-intercept of polynomial $P(x)$ of degree d [35]. Points along the polynomial are then distributed among the committee nodes. An adversary must compromise at least $d + 1$ nodes in order to learn the secret s, and must sabotage at least $n - d$ nodes to render the secret unrecoverable.

Proactive secret sharing [22] provides additional security guarantees. PSS periodically proactivizes its shares without altering the secret. Each node generates refreshed shares of the secret s that are independent of the old shares, which are discarded. This makes the scheme more robust; attackers must now compromise at least $d + 1$ nodes within the time period in order to recover the secret.

Proactive secret sharing schemes thus allow entities to store their private keys securely on committees, without relying on centralized systems. A number of blockchain systems [2, 8, 12, 24, 40], employ such schemes to store private keys and other critical data. However, these systems still face an important challenge: node churn.

In permissionless blockchains such as Ethereum, nodes can join and leave arbitrarily. In permissioned blockchains, each node must be registered with a known identity, but permissioned nodes can still stop communicating or drop from the protocol. Maintaining a

committee of nodes over an extended time scale is thus problematic; as nodes go offline the committee eventually shrinks to the point where the secret is unrecoverable. Thus, blockchain protocols for secret sharing must support dynamic committees.

Today, there are no adequate secret sharing protocols that support dynamic committees. Existing approaches focus on static committees [22, 6], assume weak or passive adversaries [33, 11], or incur prohibitive communication costs [38, 39, 34, 36, 29]. In this paper, we address this critical gap by introducing a new dynamic-committee proactive secret-sharing protocol called CHURP (CHUrn-Robust Proactivization).

CHURP Overview

The CHURP protocol enables a secret s to be stored on a dynamic committee, i.e. one in which nodes can drop arbitrarily. CHURP operates over fixed time intervals we term epochs. To counteract node churn, CHURP refreshes the committee periodically with new nodes and proactivizes the secret shares of the new committee. During each epoch, a designated committee of size n with (t, n)-sharing of a secret s maintains the user's secret. When the protocol initiates a transition to a new epoch, the old committee executes a handoff protocol to a new, randomly selected committee of size n with (t, n)-sharing of s. CHURP is secure against an active, adaptive adversary that compromises $t < n/2$ nodes in each of the old and new committees. CHURP is also able to adjust n and t between epochs.

CHURP utilizes two forms of communication: a public bulletinboard abstraction and a direct point-to-point system. The public bulletinboard is consistent and available to all live nodes, whereas point-to-point communication may fail at any time. Although we use a bulletinboard abstraction to describe the CHURP protocol, in practice we expect blockchains to fill this role.

Our main achievement in CHURP is its low communication complexity: optimistic per-epoch communication in a blockchain setting of $O(n)$ on-chain and $O(n^2)$ off-chain. We aim to minimize on-chain communication complexity because it comes with the additional costs of placing transactions on the blockchain. Cheating nodes cause pessimistic on/off-chain communication of $O(n^2)/O(n^3)$. Both off-chain costs are substantially lower compared to other schemes.

Overall, CHURP implements a simple abstraction. It simulates a trusted, available, third-party that stores cryptographic secrets for secure use in many applications.

Technical Challenges

CHURP addresses three major technical challenges in order to achieve its low communication complexity. The first challenge is that existing schemes [22] incur high communication costs during proactivization, up to $O(n^3)$ off-chain. Unlike other protocols, CHURP utilizes a bivariate secret sharing scheme, which stores the secret using a bivariate polynomial $B(x, y)$. CHURP introduces a new proactivization protocol for bivariate shares that incurs

$O(n^2)$ off-chain cost. The protocol is based on efficient bivariate 0-sharing, which generates a randomized bivariate polynomial $B(x, y)$ with $B(0, 0) = 0$ to proactivize the secret shares.

The second challenge is that during the handoff to a new committee, an adversary could compromise t nodes in the old committee and t nodes in the new committee, or $2t$ nodes in total. An attacker with 2t shares would be able to break the (t, n) secret sharing scheme. Prior work addresses this issue with 'blinding' approaches with costly communication [34]. To keep communication costs low, CHURP instead utilizes a technique called dimension-switching. Dimension-switching utilizes an asymmetric bivariate polynomial, eg a polynomial $B(x, y)$ with degree t in one dimension and degree 2t in the other. Under dimension-switching, the committee temporarily switches to $(2t, n)$ secret sharing during handoff to tolerate 2t compromised shares. The scheme reverts to a (t, n) scheme once the handoff is complete.

Finally, most PSS schemes commit to degree t polynomials using classical schemes [13, 31] that produce polynomial commits of size $O(t)$. CHURP uses an alternative polynomial commit proposed by Kate, Zaverucha, and Goldberg (KZG) [23] that produces polynomial commits of constant size ($O(1)$). KZG commitments have been used before [3], but CHURP introduces a KZG hedge. KZG assumes a trusted setup and a non-standard hardness assumption. If these fail, CHURP remains secure - but degrades to a weaker adversarial threshold of $t < n/3$.

Implementation and Experiments

We implement CHURP and evaluate it both on a single datacenter as well as on multiple data centers across the globe. Our experiments show significant reductions in communication costs, with a 2300x reduction in off-chain communication compared to state of the art dynamic-committee PSS schemes'[34] on a committee of size 100.

Outline and Contributions

After introducing the functional, adversarial, and communication models in Chapter 3, we present our main contributions:

- *CHurn Robust Proactive secret sharing (CHURP):* A novel dynamic committee PSS protocol with lower communication complexity compared to state-of-the-art PSS schemes.
- *Novel secret sharing techniques:* We introduce a new zero-sharing protocol for efficient proactivization in bivariate secret sharing settings. We also present dimension-switching, a technique to efficiently safeguard the secret during committee handoffs. Lastly, we also introduce a set of KZG hedges to safeguard the protocol's correctness in the event of failures in the KZG scheme.
- *Implementation and experiments:* We report on our implementation of CHURP and present performance measurements demonstrating practicality.

We discuss related work in Chapter 2. We will release CHURP as an open-source tool.

Chapter 2

Related Work

Bivariate Polynomials

A bivariate polynomial is comprised of two variables, commonly written as $B(x, y)$. Most classical secret sharing schemes utilize univariate polynomials to encode and distribute secrets. However, CHURP utilizes bivariate polynomials, which enable the dimension-switching and efficient 0-sharing techniques we introduce later. Bivariate polynomials have been explored extensively in secret-sharing literature, to build VSS protocols [6, 14], for secret sharing with unconditional security [36, 29], etc. Few works [33, 11], however, have considered application to dynamic committees for secret sharing, and these have been limited to passive adversaries. CHURP's use of dimension-switching provides security against active, adaptive adversaries.

Zero-sharing, the technique of generating a zero-hole ($P(0) = 0$) polynomial, has been used for proactive security since the work of [22]. However, prior works [33, 11, 29] have naively extended these to the bivariate case leading to high communication costs. Instead, CHURP applies known share re-sharing techniques [15, 10] to build an efficient bivariate zero-sharing protocol.

KZG Polynomial Commitments

PSS schemes utilize polynomial commitments to verify the consistency of polynomials, which are typically kept secret. Most protocols commit to secret degree-t polynomials using classical commitment schemes that generate commitments of size $O(t)$. However, Kate, Zavarucha, and Goldberg [23] introduce an alternative polynomial commitment scheme which is able to generate polynomial commitments of size $O(1)$. This improvement is integral in reducing the communication complexity of CHURP.

However, the constant size KZG polynomial commitments come at the expense of security guarantees. Use of KZG requires a trusted setup as well as a non-standard hardness assumption. CHURP adopts KZG commitments and assumes a correct setup in the optimistic case. However, if the setup is performed incorrectly or the KZG assumptions are

compromised, CHURP is able to detect this and switches to a pessimistic variant of the protocol that is more robust to malicious actors.

Blockchain

A blockchain is a sequence of cryptographically linked records, which store various types of information. Blockchains have been adapted for a variety of purposes, from electronic payment systems, i.e. Bitcoin [26], to systems that support smart contracts, i.e. Ethereum [5]. Each block typically includes the hash of the previous blocks, which guarantees the consistency and integrity of the data stored on the blockchain. Blockchains also provide availability: blocks are replicated across many participating nodes, ensuring that the data stored on the blockchain is always available.

CHURP utilizes a public bulletinboard component in its protocol design. Nodes can publish information to the bulletinboard, where it becomes available to all other nodes. We assume that the bulletinboard is available and that the bulletinboard's data is consistent: all nodes that attempt to retrieve a certain record will receive the same record from the bulletinboard. In practice, this bulletinboard abstraction is satisfied by blockchain systems, which provide similar consistency and availability guarantees.

Proactive Secret Sharing

Proactive security, the idea of refreshing secrets to withstand compromise, was first proposed by Ostrovsky and Yung [30] for multi-party computation (MPC). It was adapted for secret sharing by Herzberg et al, whose techniques continue to be used in subsequent works, e.g. [16, 21, 9, 25, 34, 4, 28], and in CHURP.

All the above schemes assume a synchronous network model and a computationally bounded adversary; CHURP does too, given its blockchain setting. PSS schemes have also been proposed in asynchronous settings [6, 39, 34] and unconditional settings [36, 29]. Nikov and Nikova [27] provide a survey of the different techniques used in PSS schemes along with some attacks (which CHURP addresses via its dimension-switching techniques).

Chapter 3

Model and Assumptions

In a secret-sharing scheme, a committee of nodes shares a secret s. We let C denote the committee and denote the individual committee nodes by $\{\mathcal{C}_i\}_{i=1}^{n}$. Each member node $\mathcal{C}_i$ holds a distinct share of the secret s_i. CHURP periodically proactivizes shares, i.e. changes the shares s_i without reconstructing or altering the secret. CHURP is designed for a dynamic committee, which allows member nodes to join or leave the protocol arbitrarily. Proactivization and dynamic committees together prevent leakage of s to an adversary that gradually compromises nodes.

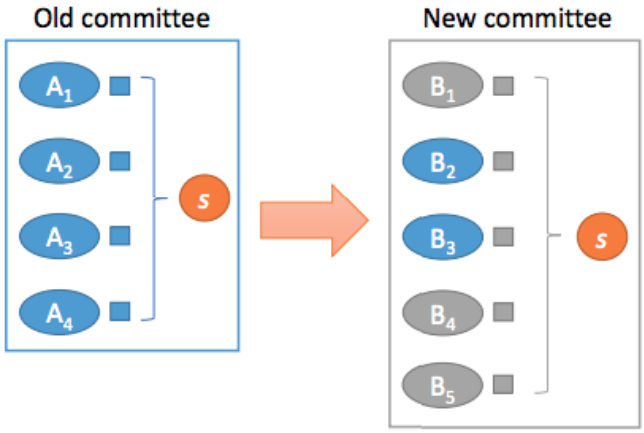

Figure 3.1: Handoff between two committees a dynamic proactive secret-sharing epoch. The secret s remains fixed. Committees may intersect, e.g., in this example, $B_2 = A_2$ and $B_3 = A_3$.

CHURP periodically executes a handoff protocol to proactivize the secret shares and transfer them to a new, randomly selected committee. The new committee may intersect the old committee. Fig. 3.1 depicts the handoff process.

Functional Model

Epoch: Time in CHURP, as in other proactive secret sharing schemes, is divided into fixed intervals of predetermined length called epochs. During each epoch, a single designated committee controls and maintains the secret s. For a given epoch e, we denote the committee of that epoch as $\mathcal{C}^{(e)}$, which has size $N^{(e)}$. $\mathcal{C}^{(e)}$ shares s using a $(t, N^{(e)})$ threshold scheme

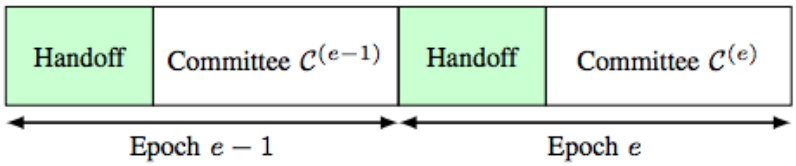

Figure 3.2: Time is divided into fixed intervals termed epochs. Each epoch begins with a handoff phase in which the old committee from the previous epoch $\mathcal{C}^{(e-1)}$ transfers the secret shares to the new committee $\mathcal{C}^{(e)}$. This is followed by a period of normal committee operation.

Handoff: As shown in Fig. 3.2, each epoch begins with a handoff phase in which the old committee $\mathcal{C}^{(e-1)}$ transfers the secret to the new committee $\mathcal{C}^{(e)}$ and proactivizes the secret. Until the handoff phase is complete, the old committee controls the secret and maintains regular operations involving s.

Churn: In the dynamic committee setting of CHURP, nodes are free to leave the protocol at any time but can only rejoin the protocol during a handoff. Let C_{left} denote the set of nodes that leave the protocol before handoff during epoch $e-1$. Let C_{alive} denote the set of nodes that perform the handoff during the transition to the next epoch e. We define the churn rate a denote a bound such that $|C_{alive}| \geq \mathcal{C}^{(e-1)} * (1-a)$. We later define a lower bound on committee size using a.

Keys: We assume that each node in CHURP maintains a public/private key pair, and that the public keys of all nodes are known by all other nodes. This is a common setting for distributed secret sharing systems.

Adversarial Model

CHURP assumes an adversary A that is both active and adaptive. The adversary is able to corrupt up to a threshold of t nodes per committee, and may corrupt these nodes at any time. Once a node is corrupted, we assume that it remains corrupted until the end of the epoch. Note that corrupted nodes may be "released" by the adversary at the end of an epoch and are no longer considered corrupted. Corrupted nodes are under an attacker's full control and can act arbitrarily.

Nodes may agree out-of-band to adjust the threshold t during the next handoff. We term the thresholds of two consecutive epochs $e-1$ and e as t_{e-1} and t_e, respectively. An

adversary may thus corrupt up to t_{e-1} nodes in $\mathcal{C}^{(e-1)}$ and t_e nodes in the $\mathcal{C}^{(e)}$. **The protocol for adjusting t is presented in Appendix D.** For simplicity of exposition, however, we assume that t remains fixed across epochs in the following sections.

During handoff, both $\mathcal{C}^{(e-1)}$ and $\mathcal{C}^{(e)}$ are active. Thus, an attacker may control up to $2t$ nodes during the handoff phase. Since committees may intersect, an attacker may control a node $\mathcal{C}_i$ that is a member of both the old and new committees. Alternatively, A may control $\mathcal{C}_i$ in one committee but not both, indicating fresh node corruption or node recovery.

Communication Model

Our primary goal is to minimize communication complexity in CHURP. We define two types of communication: on-chain and off-chain. On-chain communication measures all messages that are posted to the public bulletinboard, which is accessible by all nodes. Off-chain communication mainly consists of direct point-to-point communication between nodes. We optimize for on-chain complexity and off-chain complexity, in that order. We prioritize on-chain complexity because we utilize blockchains to serve as the public bulletinboard, and blockchains incur high costs for on-chain operations. We also consider the round complexity of the protocol design as well.

On-Chain: Prior proactive secret sharing schemes such as MPSS [34] rely on PBFT [7] for consensus. However, PBFT is slow and can incur high communication costs. Instead, we assume the availability of a blockchain (or other bulletinboard abstraction) to all nodes in the committee. The advantages of this design are twofold. First, abstracting away the consensus layer allows for simpler, more modular, and easier-to-understand protocols. Second, it makes sense to capitalize on the availability of blockchains today as opposed to re-engineering their functionality.

In our model, nodes can either post a message or retrieve any number of messages from the blockchain. After a node posts a message to the blockchain, the message is published within a time frame T. We assume that blockchain access is synchronous and that published messages are retrievable by any node. We also assume that the channel is reliable; i.e. messages posted are not lost. Such behavior can be ensured with very high probability if the transaction fees are high enough.

Permissionless Blockchains: While CHURP will work with permissioned blockchains, we focus on permissionless blockchains such as Ethereum. On such chains, users pay for writes but reads are free. We thus optimize on-chain communication by measuring the number of on-chain writes, e.g.$O(n)$ on-chain communication complexity means $O(n)$ bits are written to the chain.

Off-Chain: Nodes may also communicate directly with other nodes without use of the blockchain via point-to-point techniques. We assume that each node is able to establish a channel with every other node. We also assume that these channels are reliable, i.e. no messages are dropped, and that these channels are synchronous, i.e. we establish an upper bound on the time it takes the message to traverse the channel. We note communication

interruptions do not break CHURP. If node $\mathcal{C}_i$'s communication fails, the other nodes will simply assume the node has left the protocol.

Off-chain P2P channels can be implemented in different ways depending on the deployment environment. In a decentralized setting, however, nodes are assumed to not have P2P communication to protect from targeted attacks and anonymity compromise. In such cases, nodes can use anonymous channels such as Tor to preserve anonymity with additional setup cost and engineering complexity. Alternatively, off-chain channels can be implemented as an overlay on top of existing blockchain infrastructure.

We measure off-chain communication complexity as the total number of bits transmitted in point-to-point channels. In general, when we refer to protocols' cost in this work, we mean their communication complexity, on-chain or off-chain, as the case may be.

Chapter 4

Effective Bivariate Zero-sharing

In an ordinary (t, n)-threshold Shamir secret sharing scheme, shares of secret s are points on a univariate polynomial $P(x)$ such that $P(0) = s$. To enable its two key techniques, CHURP instead utilizes an asymmetric $\langle t, 2t\rangle$ bivariate polynomial $B(x, y)$ with $B(0, 0) = s$. A share of $B(x, y)$ is itself a univariate polynomial, given by $B(x, i)$ or $B(i, y)$ where i is the node index.

Proactivization schemes aim to refresh the secret shares without altering the secret. This commonly involves generating a fresh, random polynomial with a zero valued secret, i.e. $B(0, 0) = 0$. We term such a polynomial a zero-hole polynomial and the generation of this polynomial as zero-sharing. This zero-hole polynomial is added to the polynomial that encodes the secret s, altering each of the shares but leaving the secret unchanged.

While previous approaches to proactivization involve costly $O(n^3)$ off-chain communication during zero-sharing, we introduce a new protocol that incurs $O(n^2)$ off-chain complexity. This improvement is crucial to CHURP's overall efficiency.

Recall that in the context of bivariate polynomials, zero-sharing involves a committee C generating a $\langle t, 2t\rangle$ random bivariate polynomial $Q(x, y)$ such that $Q(0, 0) = 0$. Each node $\mathcal{C}_i$ holds a share $Q(i, y)$.

Prior works have naively extended zero-sharing techniques for univariate polynomials to the bivariate case. Such approaches require each committee node $\mathcal{C}_i$ to generate a zero-hole bivariate polynomial $Q_i(x, y)$, and distribute shares of $Q_i(x, y)$ to all other nodes. Each node transmits $O(n)$ univariate polynomials, resulting in $O(n^2)$ communication per node and $O(n^3)$ communication total.

CHURP utilizes a new technique we call BivariateZeroShare, which incurs $O(tn)$ off-chain communication cost in the optimistic case. In the pessimistic case, i.e. if nodes cheat, other protocols are invoked. Even in the pessimistic case, our techniques incur no more communication complexity than previous schemes: $O(n^3)$ in the dynamic setting and $O(n^2)$ in the static Herzberg setting.

BivariateZeroShare consists of two steps. First, a subset $\mathcal{U} \subseteq \mathcal{C}$ of 2t+1 nodes executes a zero-sharing subprotocol UnivariateZeroShare. At the end of this step, each node in $\mathcal{U}$ holds a share s_j of a univariate polynomial $P(x)$. In the second step, each node in $\mathcal{U}$ encodes its

$(2t, 2t+1)$-UnivariateZeroShare

Input: t, set of $2t+1$ nodes $\{\mathcal{U}_j\}_{j\in[2t+1]}$

Output: Each node $\mathcal{U}_j$ outputs a share $s_j = P(j)$ for randomly generated degree-$2t$ polynomial $P(y)$ with $P(0) = 0$

node $\mathcal{U}_j$

Generate a random $2t$-degree polynomial P_j s.t. $P_j(0) = 0$

Send a point $P_j(i)$ to node $\mathcal{U}_i$ for each $i \in [2t+1]$

Wait to receive points $\{P_i(j)\}_{i\in[2t+1]}$ from all other nodes

Let $P = \sum_{i\in[2t+1]} P_i$, compute share $P(j) = \sum_{i\in[2t+1]} P_i(j)$

Figure 4.1: $(2t, 2t+1)$ UnivariateZeroShare between $2t+1$ nodes. A zero-hole univariate polynomial P of degree $2t$ is generated.

share s_j in another secret-sharing layer and distributes shares of s_j to the full committee. Each committee node $\mathcal{C}_i$ thus obtains a share $Q(i, y)$ of bivariate polynomial $Q(x, y)$.

We formally present BivariateZeroShare in Fig. 4.2. For ease of presentation, we now describe the honest-but-curious version of the protocol. The full, adversary-resistant protocol is described in Chapter 6.

1. **Sharing $P(x)$:** BivariateZeroShare first chooses a subset $\mathcal{U} \subseteq \mathcal{C}$ of $2t+1$ nodes. This can be done by ordering all nodes by their public keys lexicographically, and choosing the first $2t+1$ nodes.

   The nodes of $\mathcal{U}$ then execute the UnivariateZeroShare protocol described in Figure 3. We note that this subprotocol is not new, it has been used for proactivization in [22]. Each node $\mathcal{U}_j$ generates a degree-$2t$ univariate zero-hole polynomial $P_j(x)$. The sum $P(x) = \sum_{j=1}^{2t+1} P_j(x)$ is itself a degree-2t zero-hole polynomial. $\mathcal{U}_j$ then redistributes points on its local polynomial $P_j(x)$, enabling every node $\mathcal{U}_i$ to reconstruct its share $s_i = P(i)$.

2. **Resharing $P(x)$:** Nodes in $\mathcal{U}$ now reshare $P(x)$ among all the nodes in the committee $\mathcal{C}^{(e)}$, resulting in a sharing of the desired bivariate polynomial $Q(x, y)$.

   Each node $\mathcal{U}_j$ generates a random degree-t polynomial $R_j(x)$ to encode the node's share of $P(x)$, i.e. $R_j(0) = s_j$. Together, the $2t+1$ degree-t polynomials $\{R_j(x)\}$ uniquely define a degree-$\langle t, 2t\rangle$ bivariate polynomial $Q(x, y)$ such that $Q(x, j) = R_j(x)$ and $Q(0, 0) = 0$.

   Node $\mathcal{U}_j$ then sends $R_j(i) = Q(i, j)$ to every other node $\mathcal{C}_i$ in the full committee. Using the received points, each committee member $\mathcal{C}_i$ interpolates to compute its share - a degree-$2t$ polynomial $Q(i, y)$. We note that $Q(0, 0) = 0$ because the zero coefficients of $R_j(x)$ are composed of shares generated during the zero-sharing step UnivariateZeroShare. Since each node in $\mathcal{U}$ transmits n points, the overall communication cost is $O(tn)$ off-chain.

BivariateZeroShare is used as a subroutine in CHURP with a few modifications. As explained before, it can also reduce the communication complexity of Herzberg's PSS scheme, i.e. the static committee setting, by a factor of $O(n)$.

(t, n)-BivariateZeroShare

1 : **Input:** t, n, set of nodes $\{\mathcal{C}_i\}_{i\in[n]}$ $(2t < n)$

2 : **Output:** Each node $\mathcal{C}_i$ outputs a share $Q(i, y)$ for randomly generated degree-$\langle t, 2t\rangle$ bivariate polynomial $Q(x, y)$ with $Q(0, 0) = 0$

3 : Order $\{\mathcal{C}_i\}_{i\in[n]}$ based on lexicographic order of their public keys

4 : Choose first $2t + 1$ nodes, w.l.o.g., $\mathcal{U} = \{\mathcal{C}_j\}_{j\in[2t+1]}$

5 : Invoke $(2t, 2t + 1)$-UnivariateZeroShare among $\{\mathcal{U}_j\}_{j\in[2t+1]}$ to generate shares $\{s_j\}_{j\in[2t+1]}$

6 : node $\mathcal{U}_j$:

7 : Generate a random t-degree polynomial R_j s.t $R_j(0) = s_j$

8 : Send a point $R_j(i)$ to node $\mathcal{C}_i$ for each $i \in [n]$

9 : Denote the bivariate polynomial $Q(x, y)$ where $\{Q(x, j) = R_j(x)\}_{j\in[2t+1]}$

10 : node $\mathcal{C}_i$:

11 : Wait to receive points $\{R_j(i)\}_{j\in[2t+1]} = \{Q(i, j)\}_{j\in[2t+1]}$

12 : Interpolate to reconstruct a $2t$-degree polynomial $Q(i, y)$

13 : Output share $Q(i, y)$

Figure 4.2: (t, n) BivariateZeroShare between n nodes. A zero-hole bivariate polynomial $Q(x, y)$ of degree $\langle t, 2t\rangle$ is generated

Chapter 5

CHURP Protocol Overview

CHURP is a suite of tiered protocols with different tradeoffs between trust assumptions and communication complexity. By default, CHURP attempts to execute the optimistic path - the most efficient but also the most trusting protocol. When adversarial behavior is detected, CHURP falls back to lower, more robust, protocol tiers. The three tiers of CHURP and their relationship are shown in Fig. 5.1.

The top tier, Opt-CHURP, is the default protocol. It is optimistic and highly efficient: if no node misbehaves, Opt-CHURP only incurs $O(n)$ on-chain and $O(n^2)$ off-chain communication cost. As a design choice, Opt-CHURP does not identify faulty nodes but rather just detects faulty behavior, upon which execution falls back to a lower-tier protocol.

The second tier is Exp-CHURP-A, the main pessimistic path of CHURP. Unlike Opt-CHURP, Exp-CHURP-A is able to identify and expel misbehaving nodes using proofs of correctness. Exp-CHURP-A trades performance for robustness: execution is guaranteed to complete so long as the adversarial threshold $t < n/2$, but the protocol incurs $O(n^2)$ on-chain and $O(n^3)$ off-chain costs.

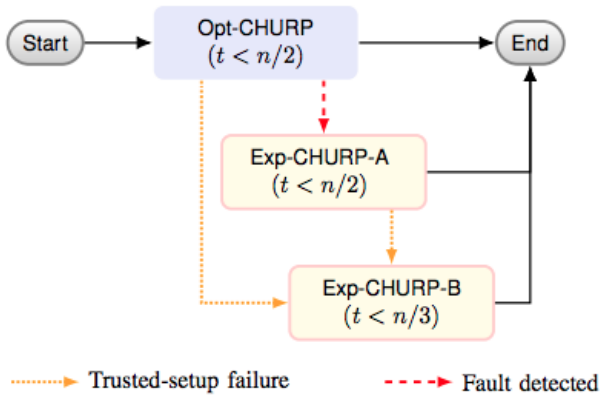

Figure 5.1: CHURP Protocol tiers. Opt-CHURP is the default protocol of CHURP. Exp-CHURP-A and Exp-CHURP-B are run only if a fault occurs in Opt-CHURP

Protocol	On-chain, Off-chain	Threshold	Optimistic
Opt-CHURP	$O(n), O(n^2)$	$t < n/2$	Yes
Exp-CHURP-A	$O(n^2), O(n^3)$	$t < n/2$	No
Exp-CHURP-B	$O(n^2), O(n^3)$	$t < n/3$	No
Opt-Schultz-MPSS	$O(n), O(n^4)$	$t < n/3$	Yes
Schultz-MPSS	$O(n^2), O(n^4)$	$t < n/3$	No

Table 5.1: On-chain cost and Off-chain costs of all protocols for the dynamic setting. An optimistic protocol ends successfully only if no faulty behavior is detected.

Both Opt-CHURP and Exp-CHURP-A utilize KZG constant-size polynomial commitments to achieve $t < n/2$. However, the KZG scheme requires a trusted setup to generate public keys with a trapdoor. The trapdoor must be destroyed after setup; otherwise soundness is lost. KZG introduces the only trusted setup required in CHURP, and thus represents its main protocol-level vulnerability. KZG also relies on the non-standard t-Strong Diffie-Hellman hardness assumption.

To hedge against soundness failure in KZG (either through a falsified trust assumption or a compromised trust setup), we introduce an additional verification step (StateVerif) to check KZG soundness. StateVerif can be executed at the end of both Opt-CHURP and Exp-CHURP-A. StateVerif is also efficient, incurring $O(n)$ on-chain complexity.

If StateVerif detects that KZG has been compromised, it triggers the third tier protocol, Exp-CHURP-B. Exp-CHURP-B does not use KZG commitments and has the same cost as Exp-CHURP-A, namely $O(n^2)$ on-chain and $O(n^3)$ off-chain. However, Exp-CHURP-B tolerates a lower adversarial threshold, $t < n/3$. The details of StateVerif are presented in Chapter 6.

In summary, the three tiers of CHURP are:

1. **Opt-CHURP:** The default protocol of CHURP. It incurs $O(n)$ on-chain and $O(n^2)$ off-chain communication complexity under the optimal resilience bound $t < n/2$
2. **Exp-CHURP-A:** Invoked if Opt-CHURP fails. It incurs $O(n^2)$ on-chain and $O(n^3)$ off-chain communication complexity under the optimal resilience bound $t < n/2$
3. **Exp-CHURP-B:** Invoked if a soundness breach is detected in KZG by StateVerif. It incurs the same cost as Exp-CHURP-A, but under the resilience bound $t < n/3$.

Dimension-Switching

CHURP uses an bivariate polynomial $B(x, y)$ of asymmetric degree-$\langle t, 2t \rangle$. By this, we mean that it is degree-t in x and degree-$2t$ in y. The secret is stored within B at $B(0, 0) = s$.

This structure enables the dimension-switching technique in CHURP. Nodes can switch between sharing in the degree-t dimension of $B(x, y)$ and the degree-$2t$ dimension. The result

is a change from $\langle t, n\rangle$ sharing to $\langle 2t, n\rangle$ sharing, or vice versa. Dimension-switching provides an efficient way to address a key challenge during committee handover. During a handover, an adversary could potentially control $2t$ nodes, necessitating a $\langle 2t, n\rangle$ sharing scheme. In between handovers, however, we opt for the more efficient $\langle t, 2t\rangle$ sharing scheme.

Notation and Invariants

We now review the notation and invariants used to explain CHURP. Notation is summarized in Table 5.2

- **KZG Commitments:** As discussed earlier, KZG commitments allow a prover to commit to a polynomial $P(x)$ and later prove correct evaluation $P(i)$ to a verifier. Further details can be found in [23]
- **CHURP Invariants:** We say the system arrives at a steady state after completing a successful handoff. The following invariants stipulate the desired properties of a steady state. We use invariants to explain the protocol and reason about its security.

    1. **Inv-Secret:** The secret s is the same across handoffs.
    2. **Inv-State:** Each node $\mathcal{C}_i$ holds a full share $B(i, y)$ and a proof to the correctness thereof. Specifically, the full share $B(i, y)$ is a degree-$2t$ polynomial, and can be represented by $2t + 1$ points $\{B(i, j)\}_{j=1}^{2t+1}$. The proof is a set of witnesses $\left\{W_{B(i,j)}\right\}_{j=1}^{2t+1}$.
    3. **Inv-Comm:** KZG commitments to reduced shares $(\{B(x, j)\}_{j=1}^{2t+1})$ are available to all nodes.

The first invariant Inv-Secret ensures the secret remains unchanged, a core-functionality of CHURP. Inv-Comm and Inv-State ensure the correctness of the protocol. For example, during the Share Reduction phase of the handoff, nodes in the old and new committee switch their dimension of sharing, from full shares to reduced shares. Using the commitments (specified by Inv-Comm) and the witnesses (specified by Inv-State), new committee members can verify the correctness of the reduced shares, and thus the correctness of dimension-switching.

Note that to realize Inv-Comm, hashes of the KZG polynomial commitments are stored on-chain for consensus while the actual commitments are transmitted off-chain between nodes.

Overview

We now present an overview of the optimistic execution of CHURP. We illustrate the details of the optimistic and pessimistic paths in the next chapter.

Notation	Description
$\mathcal{C}^{(e-1)}, \mathcal{C}^{(e)}$	Old, New committee
$B(x, y)$	Bivariate polynomial used to share the secret
$\langle t, k \rangle$	Degree of $\langle x, y \rangle$ terms in B
$RS_i(x) = B(x, i)$	Reduced share held by $\mathcal{C}_i$
$FS_i(y) = B(i, y)$	Full share held by $\mathcal{C}_i$'s
$C_{B(x,j)}$	KZG commitment to $B(x, j)$
$W_{B(i,j)}$	Witness to evaluation of $B(x, j)$ at i
$Q(x, y)$	Bivariate proactivization polynomial
$\mathcal{U}'$	Subset of nodes chosen to participate in handoff
λ_i	Lagrange coefficients

Table 5.2: Notation

At the end of a given epoch $e-1$, before a handoff occurs, the current committee $\mathcal{C}^{(e-1)}$ is in what we term a steady state. Specifically, the committee $\mathcal{C}^{(e-1)}$ holds a $\langle t, n \rangle$-sharing of the secret $s = B(0, 0)$. This sharing uses the degree-t dimension of $B(x, y)$, as noted above. Node $C_i^{(e-1)}$ holds share $s_i = B(i, y)$ and can compute $B(x, 0)$ for $x = i$. The shares s_i are actually a share of $\langle t, n \rangle$-sharing of $B(0, 0)$. We refer to these shares in steady state as full shares.

During the handoff in epoch e, nodes in the old and new committees $\mathcal{C}^{(e-1)}$ and $\mathcal{C}^{(e)}$ switch their sharing of s to the degree-$2t$ dimension of $B(x, y)$, resulting in what we term reduced shares.

Formally, node C_j^e holds reduced share $s_j = B(x, j)$, and can compute $B(0, y)$ for $y = j$. Thus, s_j is a share in a $\langle 2t, n \rangle$-sharing of $B(0, 0)$. The shares s_j have reduced power in the sense that $2t + 1$ shares (as opposed to $t + 1$ shares in steady state) are required to recover the secret s. Thus, during handoff an adversary cannot reconstruct the secret even if the adversary controls t nodes across $\mathcal{C}^{(e-1)}$ and $\mathcal{C}^{(e)}$.

After share reduction, the polynomial $B(x, y)$ is proactivized. A zero-hole bivariate polynomial $Q(x, y)$ is generated such that $Q(0, 0) = 0$ using the BivariateZeroShare protocol described above. $Q(x, y)$ is then added to $B(x, y)$, resulting in a new polynomial $B'(x, y)$. Nodes update their reduced shares accordingly. Because $Q(x, y)$ is zero-hole, the secret s remains unchanged, i.e. $s = B'(0, 0)$.

Shares of $B'(x, y)$ are now independent of those for $B(x, y)$, which were held by the old committee. This makes it safe to perform full share distribution, i.e. to switch to the degree-t dimension of $B'(x, y)$. This involves distributing full shares to the new committee $\mathcal{C}^{(e)}$. The handoff completes with the new committee $\mathcal{C}^{(e)}$ holding $\langle t, n \rangle$-sharing of a secret s using $B'(x, y)$.

To summarize, the three phases of the CHURP handoff are:

1. **Share reduction:** A switch is made from the degree-t dimension of $B(x, y)$ to the degree-$2t$ dimension. As a result, each node in the new committee C_j^e obtains a reduced share $B(x, j)$.

2. **Proactivization:** The new committee generates a zero-hole polynomial $Q(x, y)$, and each node C_j^e obtains a reduced share $Q(x, j)$. Each node C_j^e updates its share to obtain a reduced share of the proactivized polynomial, $B'(x, j) = B(x, j) + Q(x, j)$. Proactivization ensures that new committee shares are independent of those held by the old committee.

3. **Full share distribution:** Full shares $B'(i, y)$ are generated from the reduced shares $\{B'(x, i)\}$, switching from the degree-$2t$ dimension back to the degree-t dimension of $B'(x, y)$. The protocol thus returns to its steady state.

Note that during handoff, the old committee $\mathcal{C}^{(e-1)}$ can still perform operations using s, even if some nodes have left the protocol. Thus, there is no operational discontinuity of CHURP.

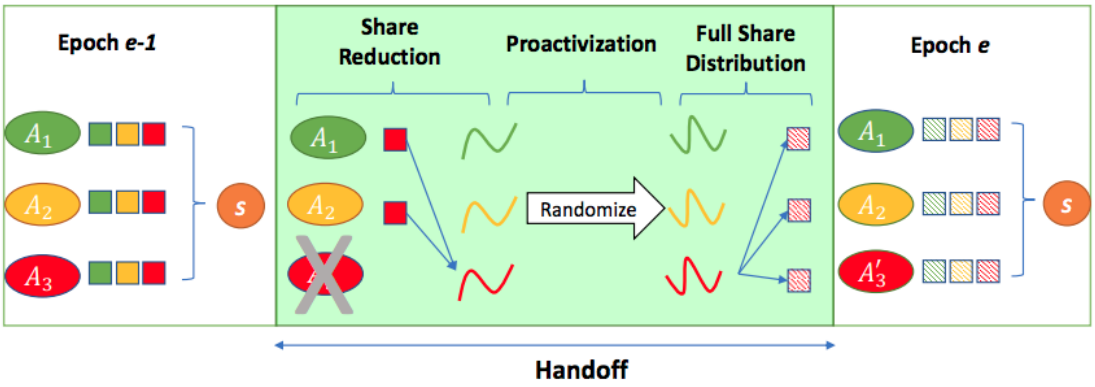

Figure 5.2: An example of the handoff protocol. Curves denote univariate polynomials (reduced shares), while squares denote points on these polynomials.

An Example

In Fig. 5.2, we show a simple example of the handoff protocol, assuming all nodes are honest. The old committee consists of three nodes $\mathcal{C}^{(e-1)} = A_1, A_2, A_3$. A_3 leaves at the end of the epoch, and a new node A_3' joins. The new committee is thus $C^e = A_1, A_2, A_3'$. The underlying polynomial $B(x, y)$ is thus of degree $\langle 1, 2\rangle$. Node A_i's share is $B(i, y)$ or three points: $B(i, 1)$, $B(i, 2)$, and $B(i, 3)$. The figure depicts all three phases of the handoff, as follows:

- **Share Reduction:** To start the handoff, each node in the new committee constructs its reduced share $B(x, j)$ from points received from the old committee. As shown, A'_3 receives points $B(1, 3)$ and $B(2, 3)$ from A_1 and A_2, respectively, from which $B(x, 3)$ can be constructed. Similarly, A_1 and A_2 construct $B(x, 1)$ and $B(x, 2)$ respectively.

- **Proactivization:** Having reconstructed the reduced shares $\{B(x, j)\}_j$, nodes in the new committee collectively generate a zero-hole bivariate polynomial $Q(x, y)$ of degree-$\langle t, 2t\rangle$, with the constraint that each node j only learns $Q(x, j)$. Reduced shares are updated as $B'(x, y) = B(x, y) + Q(x, y)$.

- **Full Share Distribution:** Nodes in the new committee get their updated full shares from the updated reduced shares. In A_1's case, A_1 knows $B'(x, 1)$ and sends $B'(2, 1)$ to A_2 and $B'(3, 1)$ to node A_3. Other nodes do the same. Hence, A_1 receives $B'(1, 2)$ and $B'(1, 3)$ from A_2 and A'_3, respectively. A_1 now has the necessary three points $B'(1, j)$ for $j \subseteq \{1, 2, 3\}$ to interpolate and recover its full share $B'(1, y)$

Chapter 6

Protocol Details

We now present the protocol details of the optimistic and pessimistic paths of CHURP.

6.1 Setup

The setup phase of CHURP executes once and sets the system to a proper initial steady state. To start, an initial committee C_0 is selected. C_0 performs the KZG setup the secret is shared among members of C_0. Using their shares, members of C_0 can generate commitments to install the three invariants.

The setup of KZG can be performed by a trusted party or a committee assuming at least one party is honest. The secret to be managed by CHURP can be generated by a trusted party or in a distributed fashion, e.g., [17]. We leave committee selection out-of-scope for this paper. Readers can refer to [18] for a discussion.

6.2 Optimistic Path (Opt-CHURP)

As described above, Opt-CHURP transfers shares of some secret s from an old committee $C = \mathcal{C}^{(e-1)}$ to a new committee $C' = \mathcal{C}^{(e)}$. CHURP supports both committee size and threshold changes, eg from $\langle n, t\rangle$ to $\langle n', t'\rangle$ in any epoch. For ease of exposition here, we allow n to change across epochs but assume a constant threshold t.

Opt-CHURP consists of three phases: Opt-ShareReduce, Opt-Proactivize, and Opt-ShareDistribution. The first phase, Opt-ShareReduce, performs dimension-switching to tolerate an adversary capable of corrupting $2t$ nodes across the old and new committees. When Opt-ShareReduce completes, nodes in the new committee hold the reduced shares. The second phase, Opt-Proactivize, proactivizes the reduced shares so that they are independent of the old shares. The last phase, Opt-ShareDistribution, reverses the dimension-switching and restores the full-shares to the new committee, thus returning to steady state.

At the beginning of Opt-CHURP, each node in C' requests the set of KZG commitments from any node in C, eg, C_1. Recall that by the invariant Inv-Comm, each node in C holds

the KZG commitments to the current reduced shares, $\left\{C_{B(x,j)}\right\}_{j=1}^{2t+1}$, while the corresponding hashes are on-chain. The received commitments are verified using the on-chain hashes. Optimistically, each node in C' receives the correct set of commitments. If a node receives corrupt ones, we switch to the pessimistic path Exp-CHURP-A where the commitments are published on-chain. The phases of Opt-CHURP are as follows:

Share Reduction (Opt-ShareReduce)

The protocol starts by choosing a subset $U' \subseteq C'$ of $2t+1$ members (possible because $C' > 2t$). The nodes in U' are denoted $\{U'_i\}_{i=1}^{2t+1}$. Some members of the old committee may have left the protocol at this point. Let $C_{alive} \subseteq C$ denote the subset of nodes that are present. Without loss of generality let this subset be $\{C_i\}_{i=1}^{|C_{alive}|}$.

By the invariant Inv-State, each live node $\mathcal{C}_i$ in the old committee holds a full share $B(i, y)$. Each node $\mathcal{C}_i$ distributes points on its full share to members of U', allowing the computation of reduced shares $B(x, j)$. This makes the dimension-switch from the degree-t dimension of $B(x, y)$ to the degree-$2t$ dimension.

Specifically, $\mathcal{C}_i$ sends $B(i, j)$ to U'_j, which interpolates the received points to recover $B(x, j)$. Note that in the optimistic path we require all $2t+1$ nodes to participate. If any adversarial nodes fail to do so, we fall back to the pessimistic path.

The received points are accompanied by witnesses allowing for verification using the KZG commitments received previously. Since $t+1$ correct points are sufficient to reconstruct the reduced share and our adversarial threshold is $t < n/2$, we need at least $n = 2t+1$ points ($|C_{alive}| > 2t$) to guarantee liveness. The size of C_{alive} is governed by the bounded churn rate a, i.e. $|C_{alive}| >= |C| * (1-a)$. Thus the condition for liveness, $|C_{alive}| > 2t$, places a lower bound on the committee size, $|C| * (1-a) > 2t$, or $|C| > \frac{2t}{1-a}$.

The protocol Opt-ShareReduce is formally specified in Fig. 6.1. At the end of Opt-ShareReduce, dimension-switching is complete and each node U'_j has a reduced share $B(x, j)$.

Communication Complexity: Each node in U' receives $O(n)$ points, so Opt-ShareReduce incurs $O(nt)$ off-chain cost.

Proactivization (Opt-Proactivize)

In this phase, U' proactivizes the bivariate polynomial $B(x, y)$ to generate new shares independent of the old ones held by members of C. The polynomial $B(x, y)$ is updated using a random zero-hole bivariate polynomial $Q(x, y)$, i.e. $Q(0, 0) = 0$. The result is an updated polynomial $B'(x, y) = B(x, y) + Q(x, y)$. Since $Q(x, y)$ is zero-hole, the secret remains unchanged, preserving the invariant Inv-Secret.

We achieve this by adapting the BivariateZeroShare technique described in Chapter 4 to handle active adversaries. Recall that BivariateZeroShare consists of two steps. First, a univariate zero-sharing subroutine generates shares of the number 0. These shares are then shared in a second step, resulting in a sharing of zero-hole $Q(x, y)$ among C'.

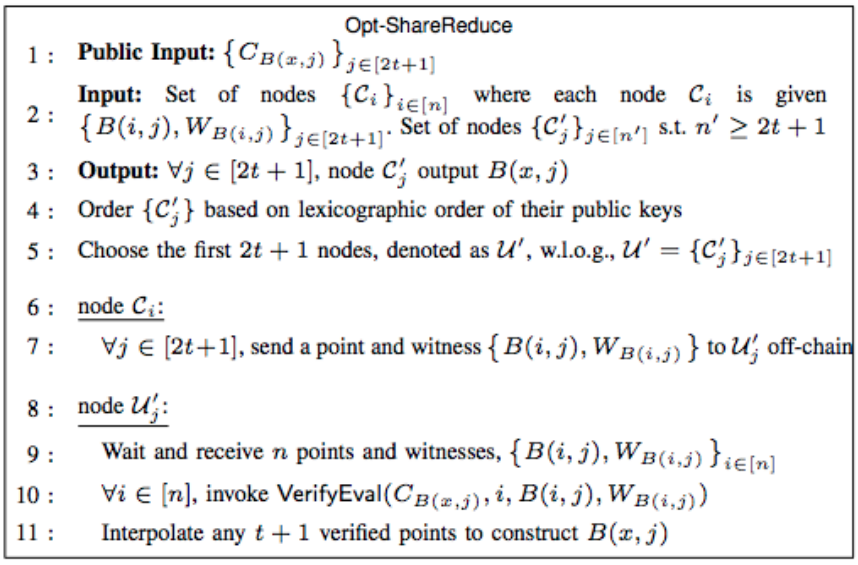

Figure 6.1: Opt-ShareReduce between the committees C and C'

By the end of the previous phase (ShareReduction), each node $U'_j \subseteq U'$ holds a reduced share $B(x,j)$. By the end of the current phase, we update these reduced shares with the shares $Q(x,j)$ generated from $Q(x,y)$.

The protocol starts by invoking the zero-sharing subroutine UnivariateZeroShare introduced previously, which is the first step of BivariateZeroShare. Specifically, $(2t, 2t+1)$-UnivariateZeroShare is run among U' to generate shares s_j at each U'_j. To handle active adversaries, U'_j sends a commitment to the share, g^{s_j}, to all other nodes in U' (where g is a publicly known generator). Lagrange coefficients $\{\lambda_j^{2t}\}_j$ can be precomputed to interpolate and verify if the shares form a 0-sharing, i.e. $\sum_{j=1}^{2t+1} \lambda_j^{2t} s_j = 0$. Translating to commitments, all nodes check the following:

$$\prod_{j=1}^{2t+1} (g^{s_j})^{\lambda_j^{2t}} = 1 \ . \tag{6.1}$$

Then, U'_j generates a random degree-t univariate polynomial $R_j(x)$ that encodes the node's share s_j, i.e. $R_j(0) = s_j$. Together, the $2t+1$ polynomials uniquely define a zero-hole bivariate polynomial $Q(x,y)$ such that $\{Q(x,j) = R_j(x)\}_{j=1}^{2t+1}$. Each node U'_j then updates its reduced share with $R_j(x)$, i.e. $B'(x,j) = B(x,j) + R_j(x)$. Points on $B'(x,j)$ are then distributed to the entire committee C' in the next phase of Opt-CHURP.

Each U'_j sends constant-size information to other nodes off-chain enabling verification of the above step. Let $Z_j(x) = R_j(x) - s_j$ denote a zero-hole polynomial, the commitment to $Z_j(x)$, C_{Z_j}, and a witness to the evaluation at zero are distributed enabling verification of the statement: $Z_j(0) = 0$; equivalent to $R_j(0) = s_j$. The commitment to the updated reduced share $B'(x,j)$ is also distributed. Since the $B'(x,j) = B(x,j) + Z_j + s_j$, the homomorphic

property of the commitment scheme allows other nodes to verify if $C_{B'(x,j)} = C_{B(x,j)} * C_{Z_j} * C_{s_j}$, where $C_{s_j} = g^{s_j}$ and the other two commitments were received previously.

In total, each U'_j generates the following set of commitment and witness information during Opt-Proactivize, $\left\{g^{s_j}, C_{Z_j}, W_{Z_j(0)}, C_{B'(x,j)}\right\}$. While this set is transmitted off-chain to all nodes in the full committee C', a hash of it is published on-chain. The received commitments can then be verified using the published hash, thereby ensuring that everyone receives the same commitments. Note that the set of commitments is sent to C' instead of just the subset U' to preserve the invariant Inv-Comm, i.e. ensure that all nodes hold KZG commitments to the updated reduced shares.

The verification mechanisms used in this protocol are sufficient to detect any faulty behavior, but do not identify the offending nodes. Thus, the adversary can disrupt the protocol without revealing his/her nodes. For example, it could send corrupt commitments to nodes selectively. Although the published hash reveals this, a verifiable accusation cannot be made since the commitments were sent off-chain. Another example would be a corrupt node sending points from a non-zero-hole polynomial in the UnivariateZeroShare protocol. Again, we detect such a fault but cannot identify which nodes are faulty. Therefore, detection of a fault simply leads to a switch to the pessimistic path, Exp-CHURP-A. While Exp-CHURP-A is capable of identifying misbehaving nodes, note that we do not retroactively identify the faulty nodes from Opt-CHURP.

The protocol Opt-Proactivize is formally specified in Fig. 6.2. By the end of this, if no faults are detected, each U'_j holds the updated reduced shares $B'(x, j)$. The invariants Inv-Secret and Inv-Comm hold as $s = B'(0, 0)$ and all of C' hold the KZG commitments respectively. In the next phase, we preserve the other invariant Inv-State.

Communication complexity: Each node in U' publishes a hash on-chain and transmits $O(t)$ data off-chain. Hence, Opt-Proactivize incurs $O(t)$ on-chain and $O(t^2)$ off-chain cost.

Full Share Distribution (Opt-ShareDist)

In the final phase, full shares are distributed to all members of the new committee, thus preserving the Inv-State invariant. A successful completion of this phase marks the end of the handoff.

By the end of the previous phase, each node U'_j in the chosen subset of nodes U' holds a new reduced share $B'(x, j)$. Now, U'_j distributes points on $B'(x, j)$, allowing computation of full shares $B'(i, y)$ by each member of the new committee C'_i. This makes the dimension-switch from the degree-$2t$ dimension of $B'(x, y)$ to the degree-t dimension. Specifically, each C'_i receives $2t + 1$ points $\{B'(i, j)\}_{j=1}^{2t+1}$, which can be interpolated to recover the full share $B'(i, y)$. This is made verifiable by sending a witness along with the points.

Since the point distribution is off-chain, a faulty node can send corrupt points without getting identified. Similar to the previous phase, when adversarial behavior is detected CHURP switches to the pessimistic Exp-CHURP-A.

The protocol Opt-ShareDist is formally specified in Fig. 6.3. If all nodes receive correct points, the optimistic path concludes successfully. The remaining invariant Inv-State

is fulfilled as each node in C' receives a full share, and hence the system returns to the steady state. After a successful completion of CHURP, we require that members of the old committee C delete their old full shares and members of U' delete their new reduced shares.

Communication complexity: Each node in C' receives $2t + 1$ points, thus Opt-Sharedist incurs $O(nt)$ off-chain cost.

Opt-Proactivize

1 : **Public Input:** $\{C_{B(x,j)}\}_{j\in[2t+1]}$

2 : **Input:** Set of nodes $\{\mathcal{C}'_i\}_{i\in[n']}$. Let $\mathcal{U}' = \{\mathcal{C}'_j\}_{j\in[2t+1]}$, each node $\mathcal{U}'_j$ is given $B(x, j)$

3 : **Output:** $\mathcal{U}'_j$ outputs `success` and $B'(x, j)$ for a degree-$\langle t, 2t\rangle$ bivariate polynomial $B'(x, y)$ with $B'(0, 0) = B(0, 0)$ (or) `fail`

4 : **Public Output:** $\left\{C_{B'(x,j)}\right\}_{j\in[2t+1]}$

5 : Invoke $(2t, 2t + 1)$-UnivariateZeroShare among the nodes $\{\mathcal{U}'_j\}_{j\in[2t+1]}$ to generate shares $\{s_j\}_{j\in[2t+1]}$

6 : node $\mathcal{U}'_j$:

7 : Generate random t-degree polynomial $R_j(x)$ such that $R_j(0) = s_j$

8 : Denote the bivariate polynomial $Q(x, y)$ where $\{Q(x, j) = R_j(x)\}_{j\in[2t+1]}$

9 : Denote the bivariate polynomial $B'(x, y) = B(x, y) + Q(x, y)$

10 : node $\mathcal{U}'_j$:

11 : Compute $B'(x, j) = B(x, j) + Q(x, j)$ and $Z_j(x) = R_j(x) - s_j$

12 : Send $\left\{g^{s_j}, C_{Z_j}, W_{Z_j(0)}, C_{B'(x,j)}\right\}$ off-chain to all nodes in $\mathcal{C}'$, where $C_{Z_j} = \mathsf{Commit}(Z_j)$; $W_{Z_j(0)} = \mathsf{CreateWitness}(Z_j, 0)$; $C_{B'(x,j)} = \mathsf{Commit}(B'(x, j))$

13 : Publish hash of the commitments on-chain $H_j = H(g^{s_j}||C_{Z_j}||W_{Z_j(0)}||C_{B'(x,j)})$

14 : node $\mathcal{C}'_i$:

15 : $\forall j \in [2t + 1]$, retrieve on-chain hash H_j, also receive $\left\{g^{s_j}, C_{Z_j}, W_{Z_j(0)}, C_{B'(x,j)}\right\}$ off-chain

16 : $\forall j \in [2t + 1]$, if $H_j \neq H(g^{s_j}||C_{Z_j}||W_{Z_j(0)}||C_{B'(x,j)})$ or $\mathsf{VerifyEval}(C_{Z_j}, 0, 0, W_{Z_j(0)}) \neq$ True or $C_{B'(x,j)} \neq C_{B(x,j)} \times C_{Z_j} \times g^{s_j}$, output `fail`

17 : Using Lagrange coefficients in Eq. (1), if $\prod_{j=1}^{2t+1}(g^{s_j})^{\lambda_j^{2t}} \neq 1$ output `fail`

18 : node $\mathcal{U}'_j$:

19 : Output `success` and $B'(x, j)$

Figure 6.2: Opt-Proactivize updates the reduced shares

Opt-ShareDist

Public Input: $\left\{C_{B'(x,j)}\right\}_{j\in[2t+1]}$

Input: Set of nodes $\{\mathcal{C}'_i\}_{i\in[n']}$. Let $\mathcal{U}' = \{\mathcal{C}'_j\}_{j\in[2t+1]}$, each node $\mathcal{U}'_j$ is given $B'(x,j)$

Output: $\forall i \in [n']$, $\mathcal{C}'_i$ outputs `success` and $B'(i,y)$ (or) `fail`

node $\mathcal{U}'_j$:

  $\forall i \in [n']$, send a point and witness off-chain $\left\{B'(i,j), W'_{B(i,j)}\right\}$ to $\mathcal{C}'_i$ where $W'_{B(i,j)} = \mathsf{CreateWitness}(B'(x,j), i)$

node $\mathcal{C}'_i$:

  Wait and receive points and witnesses $\left\{B'(i,j), W'_{B(i,j)}\right\}_{j\in[2t+1]}$

  $\forall j \in [2t+1]$, invoke $\mathsf{VerifyEval}(C_{B'(x,j)}, i, B'(i,j), W'_{B(i,j)})$

  If all $2t+1$ points are correct, interpolate to construct $B'(i,y)$

  Output `success` and the full share $B'(i,y)$

  In all other cases, output `fail`

Figure 6.3: Opt-ShareDist uses the updated reduced shares to distribute full shares in C'

Communication and Security

Each of the three phases in Opt-CHURP incur no more than $O(n)$ on-chain and $O(n^2)$ off-chain cost assuming $O(t) = O(n)$. In terms of round complexity, it completes three rounds (one for each phase) that does not depend on the committee size. The details of CHURP's pessimistic paths are discussed in the next section. Table 5.1 compares the on-chain and off-chain costs of the three paths of CHURP and Schultz-MPSS [34], the latter will be explained in more detail in Chapter 7.

The adversary can learn at most $2t$ reduced shares and t full shares of $B^e(x,y)$ for a given epoch e. Before Opt-CHURP ends, the adversary learns t full shares and t reduced shares of epoch e from t corrupted nodes in $\mathcal{C}^{(e)}$. In the subsequent handoff between committees $\mathcal{C}^{(e)}$ and C_{e+1}, the adversary learns at most another t reduced shares for epoch e, during the dimension-switch of the Opt-ShareReduce phase, leading to a total worst-case leakage of $2t$ reduced shares and t full shares of $B^e(x,y)$. It can be shown that the above leakage is not enough to compute $B^e(x,y)$. Since the shares are proactivized in the handoff, subsequent corruptions do not leak any more information about $B^e(x,y)$.

Exp-CHURP-A

This path is invoked when adversarial behavior is detected in Opt-CHURP. The first phase of Exp-CHURP-A is the same as Opt-ShareReduce, and is not re-executed if Opt-ShareReduce ends successfully.

In Exp-Proactivize, we use a different zero-sharing protocol, allowing honest parties to

avoid re-execution of the protocol in case of corruption - they can simply discard the shares generated by adversarial nodes. Messages are encrypted under the receiver's public key and posted on-chain, so that misbehaving nodes can be identified in case of corruption.

If any adversary in U' is expelled in this phase, we ask members in the old committee to publish the shares and witnesses sent to adversarial nodes during Opt-ShareReduce on the chain. In this way, all honest parties have access to the reduced shares that belong to adversarial nodes, which allows them to reconstruct the full shares in the next phase.

In Exp-ShareDist, to allow identification of malicious nodes, members post all messages on-chain in contrast to the optimistic path. Exp-Proactivize and Exp-ShareDist are presented in Fig. 6.4 and Fig. 6.5. The overall on-chain and off-chain complexities of Exp-CHURP-A are $O(n^2)$ and $O(n^3)$, respectively.

Exp-Proactivize

1 : **Public Input:** $\{C_{B(x,j)}\}_{j\in[2t+1]}$

2 : **Input:** Set of $2t+1$ nodes $\{\mathcal{U}'_j\}_{j\in[2t+1]}$. Each node $\mathcal{U}'_j$ is given $B(x,j)$

3 : **Output:** $\mathcal{U}'_j$ outputs $B'(x,j)$ for a degree-$\langle t, 2t\rangle$ bivariate polynomial $B'(x,y)$ with $B'(0,0) = B(0,0)$

4 : **Public Output:** $\left\{C_{B'(x,j)}\right\}_{j\in[2t+1]}$

5 : node $\mathcal{U}'_i$:

6 : Generate $\{s_{ij}\}_{j\in[2t+1]}$ that form a 0-sharing i.e., $\sum_{j=1}^{2t+1} \lambda_j^{2t} s_{ij} = 0$.

7 : Publish $\{g^{s_{ij}}\}_{j\in[2t+1]}$, $\{\mathsf{Enc}_{\mathsf{pk}_j}[s_{ij}]\}_{j\in[2t+1]}$ and zk proofs of correctness of the encryptions on-chain.

8 : node $\mathcal{U}'_j$:

9 : Decrypt $\{\mathsf{Enc}_{\mathsf{pk}_j}[s_{ij}]\}$ from node i and verify s_{ij} using $g^{s_{ij}}$ on-chain.

10 : node $\mathcal{U}'_j$:

11 : If any adversarial node i is detected in step 9, add it to $\mathcal{U}'_{corrupted}$, and publish s_{ji}.

12 : Set $s_j = \sum_{i\in\mathcal{U}'\setminus\mathcal{U}'_{corrupted}} s_{ij}$.

13 : Execute step 7-9, 11-12 of Opt-Proactivize in Figure 6, with messages posted on the chain in step 12.

14 : node $\mathcal{C}'_i$

15 : Execute step 16 of Opt-Proactivize in Figure 6. If it outputs `fail`, add j to $\mathcal{U}'_{corrupted}$. Nodes in $\mathcal{U}'$ discard shares by executing step 12 again.

16 : node $\mathcal{C}_i$:

17 : For all malicious nodes j detected in step 9 and 15, publish point and witness $\{B(i,j), w_{i,j}\}$ on-chain.

Figure 6.4: Exp-Proactivize protocol

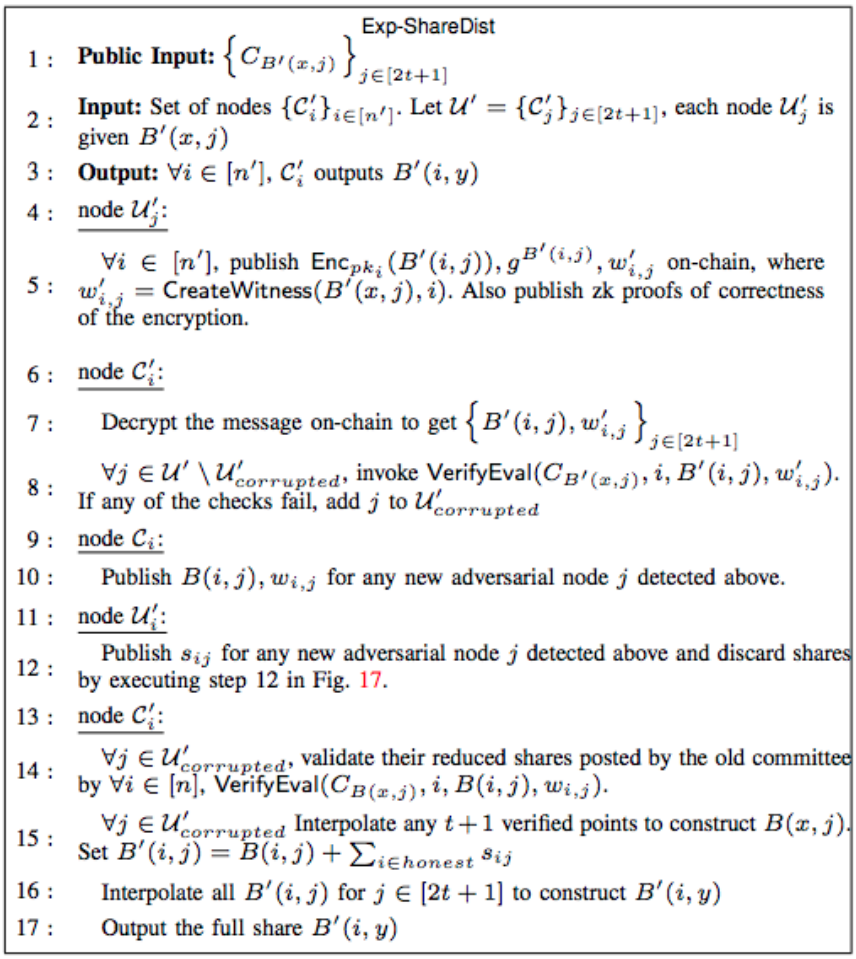

Figure 6.5: Exp-ShareDist protocol

State Verification (StateVerif)

Both Opt-CHURP and Exp-CHURP-A use the KZG commitment scheme, which requires a trusted setup phase and its security relies on the t-SDH assumption. In this section, we devise a hedge against these - a verification phase, StateVerif, that relies only on discrete log assumptions.

Recall that by the end of Opt-CHURP or Exp-CHURP-A, each member C'_i in the new committee holds a full share. StateVerif further checks that the invariants given in Chapter 5 still hold. Specifically, we check Inv-Secret: the secret is unchanged, Inv-State: these full shares form a $\langle t, 2t\rangle$ bivariate polynomial.

StateVerif can fail for two reasons: either the commitments are computed incorrectly by adversarial nodes, or the assumptions in the KZG scheme fail. If the commitments are computed incorrectly, CHURP switches to the pessimistic Exp-CHURP-A. Otherwise, it implies a failure of the assumptions in the KZG scheme. In this case, we switch to a different

pessimistic path Exp-CHURP-B.

Exp-CHURP-B

This pessimistic path is taken only after a detection of a breach in the underlying assumptions of the KZG scheme.

In this path, we use relatively expensive polynomial commitments (Pedersen commitments) instead of KZG and support a lower threshold on the number of adversarial nodes $t < n/3$. In the share reduction phase, since $n > 3t$, we rely on the error correcting mechanisms of Reed-Solomon codes to construct reduced shares, instead of the verification of the KZG scheme. In the proactivization phase and full share distribution phase, we replace the KZG commitments and verification with the Pedersen commitments (step 13 in Fig. 6.4 and steps 5, 8, 12 in Fig. 6.5). Exp-CHURP-B incurs $O(n^2)$ on-chain cost and $O(n^3)$ off-chain cost, assuming $t < n/3$.

Chapter 7

Implementation and Evaluation

7.1 Implementation

We implemented Opt-CHURP in about 2,100 lines of Go code. Our implementation uses the GNU Multiprecision Library [19] and the Pairing-Based Cryptography Library [37] for cryptographic primitives, and gRPC [20] for network infrastructure. We plan to open-source our code for CHURP in the near future.

For polynomial arithmetic, we used the polynomial ring $\mathbb{F}_p[x]$ for a 256-bit prime p. For the KZG commitment scheme, we used a type A pairing on an elliptic curve $y^2 = x^3 + x$ over F_q for a 512-bit q. The order of the EC group is also p. We use SHA256 for hashing.

CHURP can be deployed on both permissioned and permissionless blockchains. To abstract away the specific choice, we simulate one using a trusted node. Note that when deployed in the wild, writing to the blockchain would incur an additional constant latency.

7.2 Evaluation

In our evaluation, experiments are run in a distributed network of up to 1000 EC2 `c5.large` instances, each with 2 vCPU and 4GB of memory. Each instance acts as a node in the committee and the handoff protocol is executed assuming a static committee. All experiments are averaged over 1000 epochs, ie, 1000 invocations of Opt-CHURP. We measure three metrics for each epoch of the protocol: the latency (total execution time), the on-chain communication complexity (total bytes written to the bulletinboard/blockchain/trusted node), and the off-chain communication complexity (total bytes transmitted between all nodes). The evaluation results are presented below.

Latency

In the first set of experiments, all EC2 instances belong to the same region, also referred to as the LAN setting. This setting is useful to understand the time spent in computation.

Fig. 7.1 shows the latency of Opt-CHURP. The experimental results show a quadratic increase in consistent with the $O(n^2)$ asymptotic computational complexity of Opt-CHURP and suggests a low constant, eg, for a committee of size 1001 the total protocol execution time is only about 3 minutes. (Fig. 7.2). As noted before, this does not include the additional latency for on-chain writes. Note that Opt-CHURP involves only 1 on-chain write per node which happens at the end of Opt-Proactivize, and in Ethereum currently each write takes about 15 seconds. Fig. 7.2 also shows that among the three phases, Opt-ShareDist dominates the execution time due to the relatively expensive $O(n)$ calls to KZG's CreateWitness per node. (CreateWitness involves $O(n)$ group element exponentiation, thus a total of $O(n^2)$ computation).

In the second set of experiments, we select EC2 instances across multiple regions in the US, Canada, South America, Asia, and Europe, also referred to as the WAN setting. In this setting the network latency is relatively unstable, although even in the worst case it is still sub-second. Hence, during a handoff of Opt-CHURP in the WAN setting, we expect a constant increase in the latency over the LAN setting. Moreover, we expect this constant to be relatively small compared to the time spent in computation. We validate our hypothesis - for a committee of size 100, the WAN latency is 4.54 seconds while the LAN latency is 2.92 seconds (Fig. 7.1) i.e. the additional time spent in network latency is around 1.6 seconds and constant across the different committee sizes as expected. Note that we were unable to execute experiments in the WAN setting for committee sizes beyond 100 due to scaling limitations in the Amazon infrastructure.

On-chain communication complexity

Opt-CHURP incurs a linear on-chain communication complexity - n hashes, i.e. 32n bytes, are written to the blockchain in each handoff.

Off-chain communication complexity

Fig. 7.3 compares the off-chain complexity for different committee sizes for Opt-CHURP and Schultz-MPSS [34], we discuss the comparison in the following section. The experimental performance numbers are consistent with the expected $O(n^2)$ asymptotic complexity.

The off-chain data transmitted per node includes: $2n$ (polynomial point, witness) pairs in the share reduction and the share distribution phase, and n elements of F_p in the proactivization phase; each node also sends 1 commitment to share, 3 commitments to polynomials, and 1 witness. With the aforementioned parameters, a commitment to a t-degree polynomial is of size 65B bytes (with compression) and points on polynomial are of size 32B. For example, for $t = 50$ and $n = 101$, the off-chain complexity of Opt-CHURP is about $226n^2 + 325n = 2.3$MB. In Fig. 7.3, the expected curve is slightly below the observed data points because of trivial header messages unaccounted for in the above calculations.

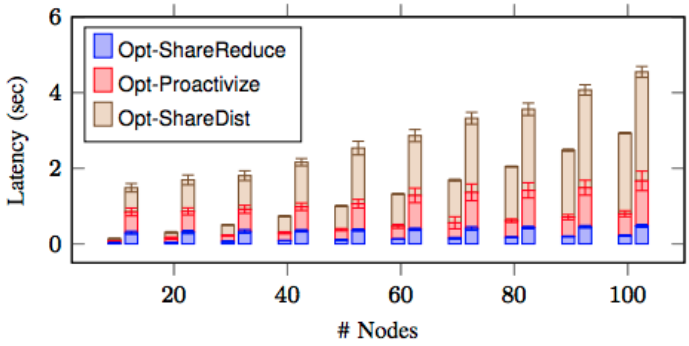

Figure 7.1: Opt-CHURP latency for the LAN and WAN setting with committee sizes 11-101, in increments of 10. The left bar is LAN latency and the right bar is WAN latency.

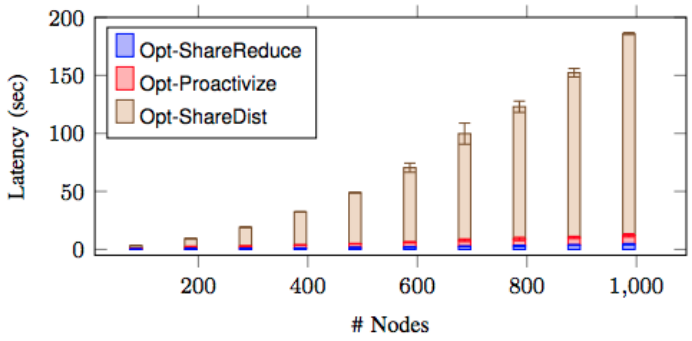

Figure 7.2: Opt-CHURP latency for the LAN setting with committee size 101-1001, in increments of 100

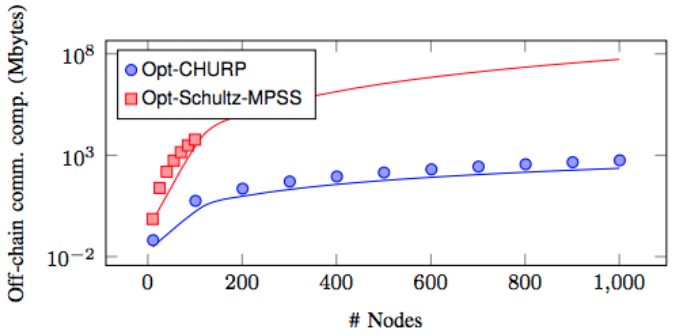

Figure 7.3: Off-chain communication complexity for Opt-CHURP and Schultz-MPSS, with log-scale y-axis. Points show experimental results; expected polynomial curves (respectively quadratic and quartic) are also shown.

7.3 Comparison with Schultz's MPSS

The Mobile Proactive Secret Sharing (MPSS) protocol of Schultz et al [34], referred to as Schultz-MPSS hereafter, achieves the similar goal as CHURP in asynchronous settings, assuming $t < n/3$. Compared to [34], Opt-CHURP achieves an $O(n^2)$ improvement for off-chain communication complexity. To evaluate the concrete performance improvement, we also implemented the optimistic path of Schultz-MPSS and evaluated the communication complexity empirically.

Asymptotic improvement: Schultz-MPSS extends the usage of expensive blinding polynomials introduced by Herzberg et al [22] to enable a dynamic committee membership. We recall briefly that the asymptotic complexity of Schultz-MPSS and refer readers to [34] for details. Each node in the old committee generates a proposal of size $O(n^2)$ and sends it to other nodes, resulting in an $O(n^4)$ off-chain communication complexity. Each node then validates the proposals and reaches consensus on the set of proposals to use by sending $O(n)$ accusations to the primary, incurring a $O(n^2)$ on-chain communication complexity. In the optimistic case where no accusation is sent - labelled as Opt-Schultz-MPSS - the consensus publishes $O(n)$ hashes of proposals on-chain and thus only incurs $O(n)$ on-chain communication complexity.

Table 5.1 compares the asymptotic communication complexity of Schultz-MPSS and CHURP. Opt-Schultz-MPSS has the same on-chain complexity as CHURP, but is $O(n^2)$ more expensive for off-chain communication.

Performance evaluation: We implemented the optimistic path of Schultz-MPSS (Section 5 of [34]) in about 3100 lines of Go code. To adapt Schultz-MPSS to the blockchain setting, we replace the BFT component of Schultz-MPSS with a trusted node. Fig. 7.3 compares the off-chain communication complexity of Opt-Schultz-MPSS and Opt-CHURP.

For practical considerations, our experiments show that Opt-CHURP can incur orders of magnitude less (off-chain) communication complexity than Opt-Schultz-MPSS. For example, for a committee of size 100, the off-chain communication complexity of Opt-Schultz-MPSS is $53.667n^4 \approx 5.3$GB, whereas that for Opt-CHURP is only 2.3MB, a 2300x improvement. Since Schultz-MPSS incurs excessive (GB) off-chain cost, we do not run it for committee sizes beyond 100.

Bibliography

[1] Brian Armstrong. *Coinbase is not a wallet*. https://blog.coinbase.com/coinbase-is-not-a-wallet-b5b9293ca0e7. Feb. 25, 2018. (Visited on 02/25/2016).

[2] Avi Asayag et al. *Helix: a scalable and fair consensus algorithm*. Tech. rep. Technical report, Orbs Research, 2018.

[3] Michael Backes, Aniket Kate, and Arpita Patra. "Computational verifiable secret sharing revisited". In: *International Conference on the Theory and Application of Cryptology and Information Security*. 2011.

[4] Kevin D Bowers, Ari Juels, and Alina Oprea. "HAIL: A high-availability and integrity layer for cloud storage". In: *the 16th ACM conference on Computer and communications security*. 2009.

[5] Vitalik Buterin. "Ethereum: A secure decentralised generalised transaction ledger". In: *Cryptography Mailing list at https://metzdowd.com* (Aug. 2013).

[6] Christian Cachin et al. "Asynchronous verifiable secret sharing and proactive cryptosystems". In: *9th ACM conference on Computer and communications security*. 2002.

[7] Miguel Castro and Barbara Liskov. "Practical Byzantine fault tolerance and proactive recovery". In: *ACM Transactions on Computer Systems* (2002).

[8] Raymond Cheng et al. "Ekiden: A Platform for Confidentiality-Preserving, Trustworthy, and Performant Smart Contract Execution". In: *arXiv preprint arXiv:1804.05141* (2018).

[9] Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. "A secure and optimally efficient multi-authority election scheme". In: *European transactions on Telecommunications* (1997).

[10] Yvo Desmedt and Sushil Jajodia. *Redistributing secret shares to new access structures and its applications*. Tech. rep. Technical Report ISSE TR-97-01, George Mason University, 1997.

[11] Shlomi Dolev et al. "Swarming secrets". In: *Allerton Conference on Communication, Control, and Computing*. 2009.

[12] Michael Egorov, MacLane Wilkison, and David Nuñez. "Nucypher KMS: decentralized key management system". In: *arXiv preprint arXiv:1707.06140* (2017).

[13] Paul Feldman. "A practical scheme for non-interactive verifiable secret sharing". In: *Foundations of Computer Science, 1987., 28th Annual Symposium on*. 1987.

[14] Pesech Feldman and Silvio Micali. "An optimal probabilistic protocol for synchronous Byzantine agreement". In: *SIAM Journal on Computing* (1997).

[15] Yair Frankel et al. "Optimal-resilience proactive public-key cryptosystems". In: *38th Annual Symposium on Foundations of Computer Science*. 1997.

[16] Rosario Gennaro et al. "Robust threshold DSS signatures". In: *International Conference on the Theory and Applications of Cryptographic Techniques*. 1996.

[17] Rosario Gennaro et al. "Secure distributed key generation for discrete-log based cryptosystems". In: *Advances in Cryptology - EUROCRYPT*. 1999.

[18] Yossi Gilad et al. "Algorand: Scaling Byzantine Agreements for Cryptocurrencies". In: *the 26th Symposium on Operating Systems Principles*. 2017.

[19] *Go language interface to GMP - GNU Multiprecision Library (golang)*. https://github.com/ncw/gmp. (Accessed on 11/14/2018).

[20] *gRPC: A high performance, open-source universal RPC framework*. https://grpc.io/. (Accessed on 11/22/2018).

[21] Amir Herzberg et al. "Proactive public key and signature systems". In: *the 4th ACM conference on Computer and communications security*. 1997.

[22] Amir Herzberg et al. "Proactive secret sharing or: How to cope with perpetual leakage". In: *Annual International Cryptology Conference*. 1995.

[23] Aniket Kate, Gregory M Zaverucha, and Ian Goldberg. "Constant-size commitments to polynomials and their applications". In: *Advances in Cryptology - ASIACRYPT*. 2010.

[24] Eleftherios Kokoris-Kogias et al. *CALYPSO: Auditable Sharing of Private Data over Blockchains*. Cryptology ePrint Archive. 2018.

[25] Haiyun Luo et al. "Self-Securing Ad Hoc Wireless Networks". In: *International Symposium on Computers and Communications*. 2002.

[26] Satoshi Nakamoto. "Bitcoin: A Peer-to-Peer Electronic Cash System". In: *Cryptography Mailing list at https://metzdowd.com* (Mar. 2009).

[27] Ventzislav Nikov and Svetla Nikova. "On proactive secret sharing schemes". In: *International Workshop on Selected Areas in Cryptography*. 2004.

[28] Mehrdad Nojoumian and Douglas R Stinson. "On dealer-free dynamic threshold schemes." In: *Adv. in Math. of Comm.* (2013).

[29] Mehrdad Nojoumian, Douglas R Stinson, and Morgan Grainger. "Unconditionally secure social secret sharing scheme". In: *IET information security* (2010).

[30] Rafail Ostrovsky and Moti Yung. "How to withstand mobile virus attacks". In: *ACM symposium on Principles of distributed computing*. 1991.

[31] Torben Pryds Pedersen. "Non-interactive and information-theoretic secure verifiable secret sharing". In: *Annual International Cryptology Conference*. 1991.

[32] Jeff John Roberts and Nicolas Rapp. *Exclusive: Nearly 4 Million Bitcoins Lost Forever, New Study Says*. en. Nov. 2017. URL: http://fortune.com/2017/11/25/lost-bitcoins/ (visited on 05/08/2018).

[33] Nitesh Saxena, Gene Tsudik, and Jeong Hyun Yi. "Efficient node admission for short-lived mobile ad hoc networks". In: *13th IEEE International Conference on Network Protocols*. 2005.

[34] David A Schultz, Barbara Liskov, and Moses Liskov. "Mobile proactive secret sharing". In: *ACM symposium on Principles of distributed computing*. 2008.

[35] Adi Shamir. "How to share a secret". In: *Communications of the ACM* (1979).

[36] Douglas R Stinson and Ruizhong Wei. "Unconditionally secure proactive secret sharing scheme with combinatorial structures". In: *International Workshop on Selected Areas in Cryptography*. 1999.

[37] *The PBC Go Wrapper*. https://github.com/Nik-U/pbc. (Accessed on 11/14/2018).

[38] Theodore M Wong, Chenxi Wang, and Jeannette M Wing. "Verifiable secret redistribution for archive systems". In: *the first International Security in Storage Workshop*. 2002.

[39] Lidong Zhou, Fred B Schneider, and Robbert Van Renesse. "APSS: Proactive secret sharing in asynchronous systems". In: *ACM transactions on information and system security (TISSEC)* (2005).

[40] Guy Zyskind, Oz Nathan, et al. "Decentralizing privacy: Using blockchain to protect personal data". In: *Security and Privacy Workshops*. 2015.