KU LEUVEN

FACULTY OF
ENGINEERING SCIENCE

Packed Pre-Constructed Publicly Verifiable Secret Sharing and Applications

Dheeraj Kumar Suryakari

Thesis submitted for the degree of
Master of Science in Cybersecurity

Supervisor
Prof. Nigel P. Smart

Assessors
Prof. Frederik Vercauteren
Dr. Steven Keuchel

Assistant-supervisors
Dr. Karim Baghery
Mr. Mahdi Rahimi

Academic year 2024 – 2025

Published by Dheeraj Kumar Suryakari,
Faculty of Engineering Science, Kasteelpark Arenberg 1 bus 2200, B-3001 Leuven

Preface

I express my gratitude to all those who have supported me throughout my master's thesis journey. I thank my promoter, Prof. Nigel P. Smart, for this opportunity and remarks during my mid term presentation. I am grateful to my assessors, Prof. Frederik Vercauteren and Dr. Steven Keuchel, for reading and evaluating this thesis report.

Shout out to my daily supervisors, Dr. Karim Baghery and Mr. Mahdi Rahimi, for their guidance and immense support throughout my thesis work. I had a great time doing the research for my thesis with them, where I learned a lot more about proof systems. I sincerely thank Dr. Karim Baghery for explaining some important concepts that were crucial for this thesis and we still have good ideas to explore, hopefully in the future.

Last but not the least, I would like to thank my family and friends for their unwavering support and encouragement throughout this journey. I would'nt be here without them.

Dheeraj Kumar Suryakari

Contents

Abstract

Publicly Verifiable Secret Sharing (PVSS) is one of the popular choices in some applications which require public verifiability when the secret is shared amongst entities. The goal of this thesis is to revisit some applications which can be made more efficient in a secure way using PVSS, for which a new variant of PVSS is proposed in thesis and it is called Packed Pre-Constructed Publicly Verifiable Secret Sharing (PPPVSS or 3PVSS).

Firstly, some preliminaries are recalled, followed by a literature review. The sections on preliminaries will give a clear insight into the security properties of the popular Shamir secret sharing which this thesis is based on, and in the section on literature the more recent advancement in PVSS is discussed. Based on the new variant of PVSS proposed in [4], which the authors call it Pre-Constructed Publicly Verifiable Secret Sharing (PPVSS), this thesis proposes 3PVSS which in fact is an extension to PPVSS and gives two practical schemes along with their security proofs.

Approaching to the goal of this thesis, an application is revisited and some changes are proposed using the proposed extension of PPVSS to make it more efficient without compromising much in security.

List of Figures and Tables

List of Figures

List of Tables

List of Abbreviations and Symbols

Abbreviations

DL	Discrete Logarithm
DLEQ	Discrete Logarithm Equality
PDL	Polynomial Discrete Logarithm
$\mathcal{PPT}$	Probabilistic Polynomial Time
NIZK	Non-Interactive Zero Knowledge
PoK	Proof of Knowledge
AoK	Argument of Knowledge
PSSS	Packed Shamir Secret Sharing
PVSS	Publicly Verifiable Secret Sharing
PPVSS	Pre-Constructed Publicly Verifiable Secret Sharing
PPPVSS	Packed Pre-Constructed Publicly Verifiable Secret Sharing

Symbols

q	prime number
$\mathbb{G}$	Cyclic group of order q
$\mathbb{Z}_q$	Modular ring with q elements
$\mathbb{Z}_q[X]$	Univariate polynomial ring in the variable X with coefficients in $\mathbb{Z}_q$
$\mathbb{Z}_q[X]_t$	Set of polynomials in $\mathbb{Z}_q[X]$ of degree t
$\mathbb{Z}_q[X]_{\leq t}$	Set of polynomials in $\mathbb{Z}_q[X]$ of degree at most t
λ	Security Parameter
$negl$	Negligible function
$\mathcal{O}$	Big-O notation

Chapter 1

Introduction

In this rapidly evolving digital world, cryptography was and is playing a crucial role, which allows a user to securely communicate and process sensitive information without any fear of eavesdropping or tampering the data. Their whole trust on cryptography boils down to their secret keys, which are used to encrypt and/or decrypt their data. In reality, however, these secret keys are stored as a whole in their device, making it a single point of failure such that an adversary can just steal their secret keys. This seriously affects the robustness and availability of the services offered by such digital infrastructures. A trivial solution for availability aspect is to make multiple copies of the secret keys and store them in multiple devices. However, this is not a good idea because now an adversary has multiple points to attack leading to decrease in the robustness of the system security. Now one can possibly think to divide a given secret key into multiple pieces such that any single piece or up to a threshold number of pieces can reveal nothing about the secret key.

In 1979, Shamir introduced a threshold secret sharing scheme called Shamir Secret Sharing scheme [20], which is now a well-known and widely used secret sharing scheme to this day because of its numerous applications in cryptography. Basically, it allows a person to divide their secret into many pieces and distribute those pieces amongst other entities, such that a single entity or up to a group of entities cannot determine anything about the secret. To reconstruct the secret back, one would require more than a threshold number of entity shares. Importantly and interestingly, under certain conditions it is proven to be secure against passive adversaries who can see secret shares of some parties and also have unlimited computational power.

Shamir secret sharing scheme was first of its kind to have such Information Theoretic (IT) security, under certain assumptions, against such passive adversaries. In reality, however, the adversaries are usually stronger than just being passive, moreover, they can be active where they possess the power to manipulate the share values of the corrupted parties itself. For example, an active adversary can manipulate some secret key shares then reconstruction protocol for the secret key may yield a wrong secret key, which will badly affect the robustness of the system. As Shamir's scheme is not tailored to defend against active adversaries as one cannot

verify the correctness of the shares, it led to inventing Verifiable Secret Sharing (VSS) schemes, which not only does allow the parties to verify the correctness of the shares shared by the dealer but also allows the parties to verify the correctness of the shares when revealed by the parties during the reconstruction phase. This allows VSS schemes to pin point who cheated during the protocol unlike in Shamir's scheme which cannot feasibly achieve such functionality. Because of the feature of verifiability, VSS schemes found their way into many applications which require security against active adversaries.

There are many VSS schemes ([10], [11]) in the literature which are based on Shamir secret sharing scheme. Throughout the years, many advancements have been made in the field of VSS schemes, and as of writing this report the efficient VSS schemes are Π_F, Π_P and Π_{LA} [3], each of which have distinct security features. In VSS, only shareholders can actually verify the correctness of the shares. Certain applications demand to have verifiability feature available to anyone, such is offered by Publicly Verifiable Secret Sharing (PVSS) schemes. PVSS is an extension of VSS, where the correctness of the shares can be verified by anyone. Many cool applications exist today which use PVSS schemes, such as, e-voting [19], randomness beacons [7], etc. In [4], authors have noticed that the Schoenmakers' PVSS scheme used for the e-voting application in [19] is actually more than a PVSS scheme, and they coined the term Pre-Constructed Publicly Verifiable Secret Sharing (PPVSS) scheme. PPVSS is a special type of PVSS where the dealer additionally publishes a commitment (/encryption) to the secret itself. The authors have also shown that any PVSS scheme can be transformed into a PPVSS scheme with minimal changes, and constructed a PPVSS Λ_{RO} from the PVSS Π_S [3] and used it to build an efficient e-voting application.

With PPVSS, one can build versatile applications and also can improve the efficiency of existing applications. For instance, in ALBATROSS [8] authors built a randomness beacon application using a PVSS. We have an intuition that an efficient randomness beacon application can be built using a scheme based on PPVSS on certain conditions. In this thesis, we will introduce Packed PPVSS (PPPVSS or 3PVSS), an extension of PPVSS where shares representing multiple secrets are secret shared, along with its security proofs and give an example based on Λ_{RO}, which will be used to improve ALBATROSS in many cases.

The research work presented in this thesis is based on the following questions:

- *What more functionalities can be achieved through PPVSS schemes?*
- *How to generalize the PPVSS schemes to the packed version which allows to secret share multiple secrets efficiently?*
- *Can we improve the efficiency of some existing applications using PPVSS schemes without compromising their security aspects?*

Outline of the thesis

The next chapter [2] gives the necessary preliminaries required to understand the mathematical security guarantees of packed Shamir secret sharing, sigma protocols and some examples that were used to build some PVSS and PPVSS schemes. The chapter ends with the description of PPVSS scheme and a realtime example of the same. Moving forward, chapter 3 introduces packed PPVSS (PPPVSS) scheme and two examples along with their security proofs where the proof for latter example is a bit non-trivial. The chapter 4 presents a new randomness beacon protocol based on a PPPVSS and compares it with the state-of-the-art randomness beacon protocol. Finally, the chapter 5 concludes the thesis with a summary of the results and discusses the possible future work and applications of the PPPVSS scheme.

Chapter 2

Preliminaries and Literature Review

As one of the goals of this chapter is to give enough background to understand the mathematical security guarantees from packed Shamir secret sharing, some basic concepts of coding theory and Reed-Solomon codes will be discussed in subsequent sections. Basics of sigma protocols will be recalled as many Verifiable secret sharing (VSS) schemes security guarantees are fundamentally based on them. Lastly, Pre-Constructed Publicly Verifiable Secret Sharing (PPVSS) is recalled in this chapter as the final goal of this thesis is to make use of their functionalities to build and improve existing applications.

2.1 Notation

Let $(\mathbb{G}, \times)$ be a cyclic group of prime order q with hard Discrete Log (DL) and its generator being g. Also, we write $\mathbb{Z}_q[X]_d$ to denote the set of all d degree polynomials univariate in X with coefficients in the finite field $\mathbb{Z}_q$. For the remainder of this chapter we let $n > t$ for some positive integers n and t.

2.2 Coding Theory

This subsection is a brief recall of linear codes and their properties.

Definition 2.2.1 (Codeword)**.** *A* ***codeword*** *of length n is a vector $c \in \mathbb{Z}_q^n$.*

Definition 2.2.2 (Linear Code)**.** *[14] If $\mathcal{C}$ be a vector subspace of $\mathbb{Z}_q^n$ with dimension k, then $\mathcal{C}$ is said to be a* ***linear code****(/ linear $q-$ary code) of length n and dimension k.*

In the remainder of the subsection, we let $\mathcal{C}$ be a linear $q-$ary code of length n and dimension k.

Definition 2.2.3 (Hamming distance)**.** *The hamming distance d of two codewords of equal length is the number of positions at which the codewords differ. Also, the hamming distance of $\mathcal{C}$, $d(\mathcal{C})$ is defined to be the minimum hamming distance of any two codewords in $\mathcal{C}$.*

Definition 2.2.4 (Hamming weight)**.** *The hamming weight wt of a codeword c is the number of non-zero positions in c. Also, the hamming weight of $\mathcal{C}$, $wt(\mathcal{C})$ is defined to be the minimum hamming weight of any codeword in $\mathcal{C}$.*

Lemma 2.2.1. *[14] Given a tuple of codewords of equal length n, (u, v, w), let $d(u, v)$ and $wt(w)$ denote the hamming distance of u, v and the hamming weight of w respectively. Then $d(u, v) = wt(u - v)$ and $d(u, v) \leq d(u, w) + d(w, v)$.*

Definition 2.2.5 (error)**.** *A vector r is said to be an **error** of a codeword $c \in \mathcal{C}$ if $r = c + e$ for some $e \neq 0$ and e is called error term of r.*

It is trivial to observe that the hamming distance of error r of $c \in \mathcal{C}$ is the minimum of the hamming distances of r with each codeword in $\mathcal{C}$.

Definition 2.2.6 (Detectable Error)**.** *An error r of $c \in \mathcal{C}$ is said to be **detectable** in $\mathcal{C}$ if $r \notin \mathcal{C}$, otherwise it is said to be an **undetectable**.*

Theorem 2.2.1. *[14] An error r of $c \in \mathcal{C}$ is detectable if the hamming distance of c and r is less than the hamming distance of $\mathcal{C}$, more precisely $d(r, c) < d(\mathcal{C})$.*

Proof. Consider the negation of the statement, i.e., the hamming distance of c and r is less than the hamming distance of $\mathcal{C}$ and r is an undetectable error in $\mathcal{C}$, mathematically we have $d(r, c) < d(\mathcal{C})$ and $r \in \mathcal{C}$. The distance of any two codewords in $\mathcal{C}$ should be at least $d(\mathcal{C})$ implying $d(r, c) \geq d(\mathcal{C})$, which is a contradiction to the negation of our statement. □

The theorem 2.2.1 says that any error of a codeword in $\mathcal{C}$ is detectable as long as their hamming distance is strictly less than the hamming distance of $\mathcal{C}$ itself.

Definition 2.2.7 (Correctable Error)**.** *A detectable error r of $c \in \mathcal{C}$ is said to be **correctable** if one can obtain its error term e such that $c + e = r$.*

Theorem 2.2.2. *[14] One can find the error term e of the detectable error r of $c \in \mathcal{C}$ if $wt(e) < \frac{d(\mathcal{C})}{2}$.*

Proof. We have the following triangular inequality from lemma 2.2.1 for any $w \in \mathcal{C}$ with $w \neq c$:

$$d(\mathcal{C}) \leq d(w, c) \leq d(w, r) + d(r, c). \tag{2.1}$$

From the equation (2.1), we get

$$d(w, r) \geq d(\mathcal{C}) - d(r, c). \tag{2.2}$$

Since, $w \neq c$ we always will have

$$d(w, r) > d(r, c). \tag{2.3}$$

. From the equations (2.2) and (2.3), we have the following result:

$$d(\mathcal{C}) - d(r, c) > d(r, c) \implies d(\mathcal{C}) > 2d(r, c) \iff d(\mathcal{C}) > 2wt(e).$$

□

If one wants to correct a detectable error of a codeword in $\mathcal{C}$ then from theorem 2.2.2, its hamming distance with the codeword should be strictly less than half the hamming distance of $\mathcal{C}$ itself.

Definition 2.2.8 (Dual Code)**.** *The vector subspace $\mathcal{C}^{\perp}$ is called a Dual (Code) of $\mathcal{C}$ if it is orthogonal to $\mathcal{C}$.*

Definition 2.2.9 (Generating Matrix)**.** *The $k \times n-$matrix $\mathcal{G}$ is said to be a generating matrix of $\mathcal{C}$ if it generates $\mathcal{C}$, more precisely, the rows of G form a basis for $\mathcal{C}$. Also, $\mathcal{G}$ is said to be in its **standard form** if it is of the form*

$$\mathcal{G} = \begin{bmatrix} I_k & P \end{bmatrix},$$

where I_k is the $k \times k$ identity matrix and P is some $k \times (n-k)$ matrix.

Definition 2.2.10 (Parity Check Matrix)**.** *Consider the linear transformation ϕ as follows:*

$$\phi : \mathbb{Z}_q^n \to \mathbb{Z}_q^{n-k},$$

*where kernel of ϕ is $\mathcal{C}$. Then the matrix associated to ϕ, $\mathcal{H}$, is called the **parity check matrix** of $\mathcal{C}$.*

Lemma 2.2.2. *[14] If $\mathcal{G}$ is a generating matrix of $\mathcal{C}$ in its standard form, i.e., $\mathcal{G} = \begin{bmatrix} I_k & P \end{bmatrix}$, then $\mathcal{H}$ being a parity check matrix of $\mathcal{C}$ is given by*

$$\mathcal{H} = \begin{bmatrix} -P^T & I_{n-k} \end{bmatrix},$$

where I_{n-k} is the $(n-k) \times (n-k)$ identity matrix where P^T is the transpose of P.

One can easily check if a codeword is in $\mathcal{C}$ by multiplying it with its corresponding parity check matrix $\mathcal{H}$.

2.2.1 Reed Solomon Codes

Consider the set of all univariate polynomials in X of degree at most t over $\mathbb{Z}_q$, denoted by $\mathbb{Z}_q[X]_{\leq t}$. It is trivial to observe that $\mathbb{Z}_q[X]_{\leq t}$ is isomorphic to the $t+1$ dimensional vector space $\mathbb{Z}_q^{t+1}$, where each vector consists the coefficients of a unique polynomial in $\mathbb{Z}_q[X]_{\leq t}$. Now, consider the following set of codewords determined by the evaluation of the polynomials in $\mathbb{Z}_q[X]_{\leq t}$ at n **distinct** points $x_1, \ldots, x_n \in \mathbb{Z}_q$:

$$RS = \{(f(x_1), \ldots, f(x_n)) : f \in \mathbb{Z}_q[X]_{\leq t}, x_i \neq x_j \text{ for } i \neq j\}. \tag{2.4}$$

Lemma 2.2.3. *Assume $n > t$. The hamming distance of any two codewords in RS is at least $n-t$. Furthermore, RS is a linear code of length n and dimension $t+1$.*

Proof. For $n > t$, saying that the hamming distance of any two codewords is at least $n-t$ is same as saying that the two codewords in RS are equal in at most t positions. Assume by contradiction that there exists two **distinct** t degree polynomials f, g in $\mathbb{Z}_q[X]$ with corresponding codewords in RS are equal in $t+1$ positions, i.e., their hamming distance is $n-t-1$. As $\mathbb{Z}_q$ is an integral domain, any $t+1$ distinct points in the set $\mathbb{Z}_q \times \mathbb{Z}_q$ will determine a unique t degree polynomial in $\mathbb{Z}_q[X]_t$. As a consequence, we should have $f = g$ in $\mathbb{Z}_q[X]$ which is a contradiction.

The remainder of the proof is by a consequence of the first part. More precisely, each codeword in RS determined by a polynomial in $\mathbb{Z}_q[X]_{\leq t}$ is actually a unique representation of the polynomial itself as it consists at least $t+1$ distinct evaluations of that polynomial where $n > t$. That is, RS is isomorphic to $\mathbb{Z}_q[X]_{\leq t}$ as a vector space which has dimension $t+1$. □

Definition 2.2.11 (Reed Solomon Code)**.** *The $q-$ary linear code RS of length n and dimension $t+1$ defined in (2.4) with minimum hamming distance $n-t$ is called a $[n, t+1, n-t]-$**Reed Solomon Code** [17] in $\mathbb{Z}_q$.*

Corollary 2.2.1. *All errors in a $[n, t+1, n-t]-$Reed Solomon (RS) code are detectable if their hamming distances with RS are at most t and $2t < n$. Moreover, all errors of hamming distance with RS being at most t can be corrected if $3t < n$.*

Proof. We have that any error of RS has hamming distance at most t with RS. From theorem 2.2.1, any error of a codeword in RS is detectable if its hamming distance is strictly less than the hamming distance of RS itself, i.e., $t < n-t$, which is equivalent to $2t < n$.

To be able to correct the errors of hamming distance at most t with RS, we need $t < \frac{n-t}{2}$ from theorem 2.2.2 which is equivalent to $3t < n$. □

In this report, we will use $[n, t+1, n-t]-$Reed Solomon code of the following form:

$$RS = \{(f(1), \ldots, f(n)) : f \in \mathbb{Z}_q[X]_{\leq t}\}. \tag{2.5}$$

The importance of the Reed-Solomon codes is its direct correspondence with Shamir's secret sharing scheme. This allows to trivially analyze their security guarantees. Therefore, the packed Shamir secret sharing scheme will now be recalled in the next section 2.3.

2.3 Packed Shamir Secret Sharing

(n, t, ℓ)-Packed Shamir secret sharing ([13],[5]) scheme is a threshold secret sharing scheme which is a variant of (n, t)-Shamir's secret sharing scheme [20] where $n > 2t + \ell - 1$. In a nutshell, the $t + \ell - 1$ degree secret polynomial with coefficients in $\mathbb{Z}_q$ which evaluates to ℓ secrets is secret shared amongst n parties such that any

$t+\ell$ parties can reconstruct back the secret polynomial. Recall that Shamir's secret sharing scheme requires at least $t+1$ parties to reconstruct the secret polynomial in contrast to the $t+\ell$ parties in the Packed Shamir secret sharing scheme. The scheme is summarized in the Figure 2.1.

Packed Shamir Secret Sharing

Given ℓ secrets to share amongst n parties, where at most t of them can be *(passively)* corrupt, the (n, t, ℓ)-Packed Shamir secret sharing scheme description is as follows:

Sharing Algorithm:

- Dealer constructs the secret polynomial $f \in \mathbb{Z}_q[X]_{t+l-1}$ via the lagrange interpolation by choosing $t+\ell$ elements in $\mathbb{Z}_q$ where ℓ of them are secrets, $\{s_i\}_{i=0}^{\ell-1}$, with $f(-i) = s_i$ for all i and remaining t are chosen uniformly at random in $\mathbb{Z}_q$.
- Each party P_i receives their share $f(i)$ from the Dealer for each $i \in \{1, \ldots, n\}$

Reconstruction Algorithm:

- Any $\mathcal{Q}$ set containing at least $t+\ell$ parties can use the lagrange interpolation to compute $\{s_i\}_{i=0}^{\ell-1}$ as follows:

$$s_m = \sum_{i \in \mathcal{Q}} f(i) \left[\prod_{j \in \mathcal{Q}, j \neq i} \frac{-m-j}{i-j} \right] \quad , m \in \{0, \ldots, \ell-1\}$$

- The secrets $\{s_i\}_{i=0}^{\ell-1}$ are outputted as the result.

FIGURE 2.1: Packed Shamir Secret Sharing

One can observe that all the secret shares of a secret polynomial chosen by the dealer form a codeword in $[n, t+\ell, n-t-\ell+1]-$RS code. If the adversary is malicious and can corrupt at most t parties, then from corollary 2.2.1, the honest shareholders can detect the errors if $2t \leq n-\ell$ and moreover all such errors can be corrected if $3t \leq n-\ell$. Also, one can use the Berlekamp-Welch algorithm [22] to correct the errors.

But Shamir secret sharing scheme is designed particularly to defend against passive adversaries and not against malicious adversaries. A class of threshold secret sharing schemes which are designed to defend against malicious adversaries is Verifiable Secret Sharing (VSS). There are many VSS schemes in the literature, and as of writing this report the efficient VSS schemes based on Shamir secret sharing are Π_F,

Π_P and Π_{LA} [3] which are alternatives to the original VSS schemes from Feldman [11], Pedersen [15] and the more recent ABCP [2]. VSS schemes based on Shamir secret sharing allow shareholders to verify the correctness of the shares obtained during both the sharing and reconstruction phases. This enables these VSS schemes to defend against malicious adversaries who can corrupt t parties as long as $t \leq \frac{n-1}{2}$ (In contrast to $t < \frac{n}{3}$ in Shamir secret sharing), and as a result it can detect the exact entity who is deviating from the protocol. Publicly Verifiable Secret Sharing (PVSS) is an extension of VSS where anyone can verify the validity of the secret shares during the sharing phase. More recently, Pre-Constructed Publicly Verifiable Secret Sharing (PPVSS)[4] was proposed which is an extension of PVSS. The main tools used in VSS, PVSS and PPVSS schemes are the **Zero Knowledge Proofs** which we overview in the next section 2.4.

2.4 Zero Knowledge Proofs

The agenda of this subsection is to give a brief formal background about some important primitives used in VSS Π_F, Π_P, Π_{LA}, the PVSS Π_S [3], and the PPVSS Λ_{RO} [4], schemes. A *Zero Knowledge Argument of Knowledge* (ZK AoK) is a protocol between a prover and a verifier where the prover tries to convince the verifier that a statement is true without revealing any information about why it is true. Unlike the Zero Knowledge Proof of Knowledge (ZK PoK) where the prover cannot cheat the verifier even with unbounded computational power, a ZK AoK requires the prover to be computationally bounded to not be able to cheat any verifier. More about ZK AoK and ZK PoK can be found in [6].

Let Y, X and W be three sets with R being a ternary relation on $Y \times X \times W$. Consider the three PPT interactive algorithms (*Setup*, P, V), where *Setup* returns a common reference string (CRS) σ when input 1^λ. Given σ, the following is the CRS-dependent language of the relation R.

$$L_\sigma = \{x \in X : \exists w \in W, (\sigma, x, w) \in R\},$$

where w is called a witness for a statement x if $(\sigma, x, w) \in R$. Also, let $\mathcal{R}$ be a PPT algorithm that returns an element in the relation R when input 1^λ.

Given a relation R and its corresponding language, a **$\sum$-protocol** is a 3-round *interactive* protocol between two Probabilistic Polynomial Time (PPT) algorithms, a prover P and a verifier V. For some statement in the language of R, in the first round P sends a commitment a to V. To which V sends a challenge d to P in the second round and finally P responds back with the response z to V in the third round. V outputs **true** or **false** upon the proof verification on transcript $trans := (a, d, z)$. Informally, with a $\sum-$protocol a prover P tries to convince a verifier V that they know a witness w for a given statement x in the language without revealing any information about w. To state it formally, a $\sum-$protocol is supposed to satisfy *completeness*, *Honest Verifier Zero Knowledge* (HVZK) and *Special Soundness* which are defined as follows.

Definition 2.4.1 (Completeness). *A $\sum-$protocol is said to be **complete** for $\mathcal{R}$ if the honest verifier V always accepts the honest prover P for any statement in the language defined by R.*

Definition 2.4.2 (HVZK). *A $\sum-$protocol is said to be Honest Verifier Zero Knowledge(**HVZK**) for $\mathcal{R}$ if there exist a PPT algorithm $\mathcal{S}$ that simulates trans of the scheme corresponding to a given statement, x, in the language that has witness w. That is, given x,*

$$trans(P(x,w) \leftrightarrow V(x)) \approx trans(\mathcal{S}(x) \leftrightarrow V(x)) \qquad \text{, for any witness } w \text{ of } x.$$

Where $trans(P(\cdot) \leftrightarrow V(\cdot))$ is the transcript of the $\sum-$protocol amongst P and V and $\approx$ denotes the indistinguishability of the two transcripts.

Definition 2.4.3 (Special Soundness). *A $\sum-$protocol is said to satisfy **Special Soundness** for $\mathcal{R}$, if there exists a PPT extractor $\mathcal{E}$ for any two valid transcripts, (a,d,z) and (a,d',z'), corresponding to a given statement x in the language with only a unique witness w and $d \neq d'$ such that $\mathcal{E}(a,d,z,d',z')$ outputs the witness w.*

It is shown that a public-coin, complete, HVZK, special soundness $\sum-$protocol can be made into a Non Interactive Zero Knowledge (NIZK) Proof of Knowledge (PoK) or Argument of Knowledge (AoK) in the Random Oracle (RO) model using Fiat-Shamir transform [12]. In the following subsections, two important NIZK PoK schemes are recalled which were used in Π_S and Λ_{RO} schemes and we are going to use them to build our protocols introduced in the next chapter.

2.4.1 Chaum-Pedersen Protocol for DL Equality

Recall $\mathbb{G}$ being the cyclic group of prime order q with hard Discrete Logarithm (DL). For some $g, h \in \mathbb{G}$ consider the following relation:

$$R_{DLEQ} = \{(g,h,a,b), x : a = g^x, b = h^x\}.$$

In [9], Chaum and Pedersen proposed a NIZK PoK scheme for the DL Equality relation, R_{DLEQ}. Informally, a prover P can convince a verifier V that they know x such that it can be used with both g and h to obtain a and b respectively. This protocol is widely used in many cryptographic applications like threshold decryption, e-voting and Randomness Beacons. We summarize the protocol in Figure 2.2.

2.4.2 NIZK PoK for Polynomial DL

Recall $\mathbb{G}$ being the cyclic group of prime order q with hard Discrete Logarithm (DL) and g being its generator. Consider the following relation for some polynomial $f \in \mathbb{Z}_q[X]_t$ with degree $t < n$:

$$R_{PDL} = \{(g, x_1, \ldots, x_n, F_1, \ldots, F_n), f(X) : (F_i = g^{f(x_i)}, 1 \leq i \leq n) \wedge (x_i \neq x_j, i \neq j)\}.$$

The R_{PDL} is based on the Polynomial Discrete Logarithm (PDL) formally introduced in [3], and its definition is recalled in the following.

Chaum-Pedersen Protocol for DLEQ

Let g be the group generator sampled in the setup phase, $[(g, h, a, b), x] \in R_{DLEQ}$ and $\mathcal{H}$ be a Random Oracle (RO).

Prover

- Samples $r \in_R \mathbb{Z}_q$ uniformly at random and sets $c_1 = g^r$ and $c_2 = h^r$.
- Sets $d \leftarrow \mathcal{H}(a, b, c_1, c_2)$, where $\mathcal{H}$ is an agreed upon Random Oracle (RO).
- Sets $z \equiv r + dx \pmod q$ and returns the proof(/transcript) $\pi := (d, z)$.

Verifier

- Checks if $d \leftarrow \mathcal{H}(a, b, \frac{g^z}{a^d}, \frac{h^z}{b^d})$ and outputs **true** or **false** accordingly.

Figure 2.2: Chaum-Pedersen NIZK PoK for DLEQ

Definition 2.4.4 (Polynomial Discrete Logarithm)**.** *Let g be a generator for the prime order q cyclic group $\mathbb{G}$. Given $F_1, \ldots, F_n$ and distinct elements $x_1, \ldots, x_n$ in $\mathbb{Z}_q$, find a polynomial $f \in \mathbb{Z}_q[X]$ with degree at most t, where $F_i = g^{f(x_i)}$ for $1 \leq i \leq n$ and $t \leq n - 1$.*

In other words, an algorithm $\mathcal{A}$ is said to have an advantage ϵ in solving PDL if

$$Pr[\mathcal{A}(x_1, \ldots, x_n, g, g^{f(x_1)}, \ldots, g^{f(x_n)})] \geq \epsilon,$$

where $f \in \mathbb{Z}_q[X]$ is at most a t degree polynomial with $t \leq n - 1$ and the probability is over a chosen random generator g of $\mathbb{G}$ with $q = |\mathbb{G}|$ being prime and distinct $x_1, \ldots, x_n$ elements in $\mathbb{Z}_q$.

Based on the hardness of PDL, a NIZK PoK scheme π_{PDL} was proposed in [3] which is summarized in figure 2.3.

Theorem 2.4.1. *[3] Let $[(g, x_1, \ldots, x_n, F_1, \ldots, F_n), f(X)] \in R_{PDL}$ where $f \in \mathbb{Z}_q[X]_{\leq t}$. Assuming PDL is hard, for $0 \leq t \leq n-1$, the protocol π_{PDL} 2.3 (described in figure 2.3) is a NIZK PoK for R_{PDL} in the RO model.*

In [3], Baghery formally introduced a NIZK PoK scheme for the Polynomial DL relation, R_{PDL}, which is a generalization of Schnorr's ID protocol [18]. Informally, a prover P can convince a verifier V that they know a $t-$degree polynomial f such that it can be used with g to obtain F_i for $1 \leq i \leq n$. A fairly recent application based on this protocol is used to construct an efficient e-voting protocol [4] where

π_{PDL}

Let g be a group generator sampled in the setup phase, $[(g, x_1, \ldots, x_n, F_1, \ldots, F_n), f(X)] \in R_{PDL}$ and $\mathcal{H}$ be a random oracle (RO).

Prover

- Samples $r \in_R \mathbb{Z}_q[X]_t$ uniformly at random and sets $\Gamma_i = g^{r(x_i)}$ for $1 \leq i \leq n$.
- Sets $d \leftarrow \mathcal{H}(F_1, \ldots, F_n, \Gamma_1, \ldots, \Gamma_n)$, where $\mathcal{H}$ is an agreed upon Random Oracle (RO).
- Sets $z(X) \equiv r(X) + df(X) \pmod q$ and returns the proof(/transcript) $\pi := (d, z(X))$.

Verifier

- First checks if z is at most a t degree polynomial in $\mathbb{Z}_q[X]_t$. If so, they proceed with the next step.
- Checks if $d \leftarrow \mathcal{H}(F_1, \ldots, F_n, \frac{g^{z(x_1)}}{F_1^d}, \ldots, \frac{g^{z(n)}}{F_n^d})$.
- If first two steps are correct then they output **true**, otherwise **false**.

FIGURE 2.3: A NIZK PoK for Polynomial DL based on Schoenmakers' PVSS

the authors have introduced the Pre-Constructed Publicly Verifiable Secret Sharing scheme and used it as a building block.

2.5 Pre-Constructed Publicly Verifiable Secret Sharing

Pre-Constructed Publicly Verifiable Secret Sharing (PPVSS) was first introduced in [4], which was used as a building block to construct an efficient e-voting protocol alternative to Schoenmakers' e-voting protocol. Interestingly, the authors of [4] observed that the Schoenmakers' e-voting protocol, though not efficient in practice, published in 1999 is an unusual application as it is based on a PVSS, which led them to discover that the underlying PVSS used is actually a PPVSS. What sets PPVSS apart from standard PVSS schemes is that it can be used to build versatile applications, such as e-voting, randomness beacons etc., and can also improve efficiency of some existing protocols. The subtle difference between PPVSS and PVSS is that the secret itself is committed by the prover along with all its corresponding

secret shares. The relevance of this section is to generalize the PPVSS to the case where there is more than one secret and use them to revisit a real time application to improve its efficiency in the subsequent chapters. Because of its importance in this thesis, the definitions of a PPVSS and an example is recalled in the upcoming sections.

2.5.1 Definitions

The following definitions are directly taken from [4].

Definition 2.5.1 (PPVSS)**.** *A PPVSS scheme should have four algorithms, namely, Initial, Share, Verify and Reconstruction whose descriptions are as follows:*

- ***Initial*** $(1^\lambda) \rightarrow (\{PK_i, SK_i\}_{i=1}^n \sqcup \{h_0 \text{ or } (PK_0, SK_0)\})$*: When given* 1^λ*, each party* P_i *for* $1 \leq i \leq n$ *registers their public key* PK_i *in a public ledger and withholds the corresponding secret key* SK_i*. Also, all parties and the dealer* D *agree on a commitment key or public key whose secret key is known to a target person. Note that the message space of the public-key scheme is a subgroup of* $(\mathbb{G}, \times)$*.*

- ***Share*** $(n, t, f_0, \{PK_i\}_{i=1}^n \bigsqcup \{h_0 \text{ or } PK_0\})$
  $\rightarrow (\{y_i\}_{i=0}^n, \pi_{PPVSS})$*:*
  It secret shares f_0 *to obtain the shares* $\{f_i\}_{i=1}^n$*. For* $1 \leq i \leq n$*, uses the public key* PK_i *to encrypt* f_i *and obtain the encrypted share* y_i*. Now, it uses the commitment key* h_0 *(or public key* PK_0*) to commit(/encrypt)* f_0 *to obtain* y_0*. In the next step, it uses a NIZK proof* π_{PPVSS} *protocol to prove that* y_0 *is a valid commitment(/encryption) of the secret and* $\{y_i\}_{i=1}^n$ *has valid encryptions of the corresponding shares. Finally, it returns* $(\{y_i\}_{i=0}^n, \pi_{PPVSS})$*.*

- ***Verify*** $(n, t, \{y_i\}_{i=0}^n, \pi_{PPVSS}) \rightarrow$ ***true/false****: This algorithm(which can be performed by anyone) checks if the NIZK proof* π_{PPVSS} *is valid for* $\{y_0, \ldots, y_n\}$ *and then returns true, otherwise false.*

- ***Reconstruct****: There are two approaches based on cooperation of the dealer* D *and are as follows:*

  - ***(Optimistic)*** $Reconstruction^{opt}[\{h_0, f_0, r_0, y_0\} \text{ or } \{PK_0, SK_0, y_0\}]$
    $\rightarrow (f_0 \text{ or } false)$*:*
    Given the input a verifier checks if y_0 *is a valid commitment(/encryption) of the secret. If so, it returns* f_0*; otherwise it returns* $false$*.*
  - ***(Pessimistic)*** $Reconstruction^{pes}[\{y_i, SK_i\}_{i \in \mathcal{Q}, |\mathcal{Q}|=t+1}] \rightarrow (f_0 \text{ or } false)$*: Given any* $t+1$ *encrypted shares along with corresponding secret keys, it outputs the secret* f_0 *or* $false$*. This can be done in two phases as follows:*
    - ***Decryption of the shares****, each party* $P_i \in \mathcal{Q}$ *decrypts* y_i *to obtain* f_i *using its secret key* SK_i*. Then it generates a NIZK proof* π_i^{Dec}

which proves that f_i is the correct decryption of y_i. Now, P_i publishes (f_i, π_i^{Dec}).

- ∗ ***Share pooling**, a verifier V (not necessarily from the shareholders) checks if proof π_i^{Dec} is correct for each $P_i \in \mathcal{Q}$. If any check fails, then V returns $false$; otherwise V applies a reconstruction procedure to the set of valid shares, $\{f_i\}_{i\in\mathcal{Q},|\mathcal{Q}|=t+1}$, and returns f_0.*

A PPVSS is said to be secure:

- **Correctness**: If the dealer and parties follow the protocol, then the ***Verify*** algorithm returns ***true*** and the ***Reconstruct*** algorithm returns f_0 irrespective of which approach. For any integer $n \geq t+1$ with $t \geq 0$, a PPVSS is said to be correct for $(\{PK_i, SK_i\}_{i=1}^n \sqcup \{h_0 \text{ or } (PK_0, SK_0)\}) \leftarrow$**Initial** (1^λ) when it satisfies the following based on the output of the *Share* algorithm.

  When *Share* algorithm outputs a commitment(/encryption) of the secret, then

$$Pr\begin{bmatrix}(\{y_i\}_{i=0}^n, \pi_{PPVSS}) & \leftarrow \textbf{\textit{Share}}\big(n,t,f_0,\{PK_i\}_{i=1}^n \bigsqcup \\ & \{h_0 \text{ or } PK_0\}: \\ \textbf{\textit{true}} & \leftarrow \textbf{\textit{Verify}}(n,t,\{y_i\}_{i=0}^n,\pi_{PPVSS})\end{bmatrix} = 1,$$

$$Pr\begin{bmatrix}(\{y_i\}_{i=0}^n, \pi_{PPVSS}) & \leftarrow \textbf{\textit{Share}}\big(n,t,f_0,\{PK_i\}_{i=1}^n \bigsqcup \\ & \{h_0 \text{ or } PK_0\}), \\ f_0' & \leftarrow Reconstruction^{opt}[\{h_0,f_0,r_0,y_0\} \\ & or\ \{PK_0,SK_0,y_0\} \bigvee \\ f_0' & \leftarrow Reconstruction^{pes}[\{y_i,SK_i\}_{i\in\mathcal{Q},|\mathcal{Q}|=t+1}]: \\ & f_0' = f_0\end{bmatrix} = 1.$$

- **Verifiability**: If ***Verify*** returns ***true***, then ***(Optimistic)*** $Reconstruct^{opt}$ and/or ***(Pessimistic)*** $Reconstruct^{pes}$ output being f_0 is the actual secret of whom the shares are encrypted. Moreover, $\{y_i\}_{i=1}^n$ are valid encryptions of the shares of same secret with high probability if the following statement is true.

  y_0 is a valid commitment(/encryption) of the secret with high probability. More formally, given λ, for any integers $n \geq 2t+1$ and $t \geq 0$, a PPVSS is said to be verifiable if for any $\mathcal{PPT}$ adversary $\mathcal{A}$, we have:

$$Pr\begin{bmatrix}(\{PK_i,SK_i\}_{i=1}^n \sqcup & \{h_0 \text{ or } (PK_0,SK_0)\}) \leftarrow \textbf{ Initial } (1^\lambda), \\ (\{y_i\}_{i=0}^n, \pi_{PPVSS}) & \leftarrow \mathcal{A}\big(n,t,f_0,\{PK_i\}_{i=1}^n \bigsqcup \\ & \{h_0 \text{ or } PK_0\}\big), \\ f_0' & \leftarrow Reconstruction^{opt}[\{h_0,f_0,r_0,y_0\} \\ & or\ \{PK_0,SK_0,y_0\}] \bigvee \\ f_0' & \leftarrow Reconstruction^{pes}[\{y_i,SK_i\}_{i\in\mathcal{Q},|\mathcal{Q}|=t+1}]: \\ \textbf{\textit{true}} & \leftarrow \textbf{\textit{Verify}}(n,t,\{y_i\}_{i=0}^n,\pi_{PPVSS}) \bigwedge f_0' \neq f_0\end{bmatrix} \leq negl(\lambda),$$

  where $\mathcal{Q}$ is the set of honest parties.

- **IND1-Secrecy (Indistinguishability of Secrets)**: Before reconstruction phase, any amount of public information along with secret keys of at most t parties excluding SK_0 will give absolutely no information about the secret f_0. More formally, for integers $n > 1$ and $t + 1 \leq n$, the PPVSS is said to satisfy *IND1-Secrecy* if for any $\mathcal{PPT}$ adversary $\mathcal{A}$ corrupting at most t parties, excluding the owners of SK_0, has negligible advantage in the following game played against a challenger.

  – The challenger runs ***Initial*** (1^λ) of PPVSS to obtain $\{PK_i, SK_i\}_{i=1}^n \sqcup \{h_0 \text{ or } (PK_0, SK_0)\}$ and sends all public information along with secret information of all corrupted parties to $\mathcal{A}$.

  – The challenger chooses two secrets, $s_0 = f_0$ and $s_1 = f_0'$ at random in the space of secrets. Furthermore, it chooses $b \in \{0, 1\}$ uniformly at random and runs ***Share*** $(n, t, s_0, \{PK_i\}_{i=1}^n \bigsqcup \{h_0 \text{ or } PK_0\})$ algorithm of the PPVSS scheme and obtains $(\{y_i\}_{i=0}^n, \pi_{PPVSS})$. Finally, it sends all public information generated in *Share* phase together with s_b.

  – $\mathcal{A}$ guesses a bit $b' \in \{0, 1\}$.

  The advantage of $\mathcal{A}$ is defined to be $|Pr[b' = b] - \frac{1}{2}|$.

Λ_{RO} is the PPVSS scheme introduced in [4], recalled in figure 2.4, which is actually based on Π_S [3].

The security guarantees of Λ_{RO} are explained in [4] which follows from the security guarantees of Π_S, also the authors used Λ_{RO} to build the new e-voting protocol alternative to the Schoenmakers' e-voting protocol [19] and showed that they achieved improvement in the verification phase (7 to 30 times faster in implementation for some parameters).

2.6 Conclusion

In summary, some basic concepts of linear codes and Reed-Solomon codes were recalled. Next, Packed Shamir secret sharing scheme which is a generalization of the Shamir Secret Sharing scheme [20] was recalled and its correspondence with Reed-Solomon codes is discussed to explain how its mathematical security guarantees are achieved against passive adversaries. As one of the goals of this thesis is to explore threshold secret sharing schemes that commits to secret and also can defend against malicious adversaries, PPVSS [4] was recalled. Moving forward, in the next chapter we will introduce Packed Pre-Constructed Publicly Verifiable Secret Sharing (PPPVSS) which is a generalization of PPVSS and give two examples whose inspiration is drawn from Λ_{RO}.

Λ_{RO} [4]

Initialization: All parties $\{P_i\}_{i=1}^n$ and dealer D agree on the prime field $\mathbb{Z}_q$, a group $(\mathbb{G}, \times)$ of order q with a generator g, random oracle $\mathcal{H}$. Also, each party P_i registers their public key PK_i in the public ledger and all agree on a commitment key or a public key, $PK_0 \neq g$, whose secret key is known to a target person.

Share:

- Dealer D samples a t-degree polynomial $f \in \mathbb{Z}_q[X]$ uniformly at random and sets g^{f_0} as the secret to be shared, where $f_0 = f(0)$.
- For each $1 \leq i \leq n$, D computes $f(i) = f_i$ and uses PK_i to encrypt the secret share to obtain $PK_i^{f_i} = y_i$. D also encrypts(/commits) to the secret g^{f_0} using the encryption(/commitment) key PK_0 to obtain $y_0 = PK_0^{f_0}$.
- D uses the PDL proof scheme 2.4.2 to generate π_{PDL} as follows:
  - Samples a $t-$degree polynomial $r \in \mathbb{Z}_q[X]$ uniformly at random and computes $c_i = PK_i^{r_i}$ where $r_i = r(i)$ for $0 \leq i \leq n$.
  - Using $\mathcal{H}$, $d = \mathcal{H}(y_0, \ldots, y_n, c_0, \ldots, c_n)$ is computed.
  - Sets $z(X) = r(X) + df(X)$, hence $\pi_{PDL} = (d, z(X))$ is obtained.
- D broadcasts the encryptions of the shares along with the encryption(/commitment) of the secret with π_{PDL} which proves the validity of the encrypted shares and encrypted(/committed) secret, i.e., broadcasts $\{y_i\}_{i=0}^n$ and $(d, z(X))$.

Verification: Given public keys $\{PK_i\}_{i=1}^n$ and public(commitment) key PK_0, any entity can check π_{PDL} to verify the correctness of the encrypted shares $\{y_i\}_{i=1}^n$ and y_0 being the encryption(/commitment) of the secret. They will output **true** or **false** based on the verification of the proof. The procedure is outlined as follows:

- The entity checks if $z(X)$ is a $t-$degree polynomial or not.
- Checks if $d = \mathcal{H}(y_0, y_1, \ldots, y_n, \frac{PK_0^{z(0)}}{y_0^d}, \frac{PK_1^{z(1)}}{y_1^d}, \ldots, \frac{PK_n^{z(n)}}{y_n^d})$.
- Outputs **true** if both of the above checks are satisfied, otherwise **false**.

Reconstruction: There are two approaches to reconstruct the secret g^{f_0} based on the cooperation of the dealer D which are as follows:

- **Optimistic Reconstruction:** D publishes f_0, then any verifier (not necessarily shareholders) when given g, PK_0, y_0 can check if $y_0 = PK_0^{f_0}$ and returns g^{f_0} if the check passes, if not they return **false**.
- **Pessimistic Reconstruction:** If D refuses to reveal f_0, then any set $\mathcal{Q}$ consisting at least $t+1$ shareholders will do the following (which is same as the reconstruction step in the PVSS Π_S protocol):
  - Each party $P_i \in \mathcal{Q}$ decrypts their share y_i using their private key SK_i corresponding to PK_i to obtain g^{f_i}. Then they publish g^{f_i} along with a DLEQ proof 2.4.1, π_{DLEQ} which proves that the g^{f_i} is the correct decryption of y_i.
  - If $\mathcal{Q}$ consists at least $t+1$ honest parties, they can use the lagrange interpolation to compute the secret g^{f_0} as follows:

$$g^{f_0} = \prod_{i=1}^{t+1} g^{f_i^{\lambda_i}} = g^{\sum_{i=1}^{t+1} f_i \cdot \lambda_i},$$

    where $\lambda_i = \prod_{j \neq i} \frac{j}{j-i}$ are lagrange coefficients.

FIGURE 2.4: a Pre-Constructed Publicly Verifiable Secret Sharing (PPVSS) scheme

Chapter 3

Packed Pre-Constructed Publicly Verifiable Secret Sharing

In most of the real time distributed applications, each participant runs as a dealer once to share their secrets with other participants. To retrieve back the secrets, remaining participants need to open their shares and perform a bunch of operations, which requires a lot of communication amongst the participants. During the whole time of secret reconstruction one could have asked to the dealer itself to reveal the secret which will save a lot of time and communication. This is the main motivation behind the work of *Pre-Constructed Publicly Verifiable Secret Sharing* (PPVSS) [4], where the authors have given the complete description of PPVSS and an example Λ_{RO} to improve the original e-voting protocol proposed by Schoenmakers [19].

In this chapter we will introduce Packed Pre-Constructed Publicly Verifiable Secret Sharing (PPPVSS or 3PVSS) which merely is an extension to the notion of PPVSS. As Packed Shamir secret sharing 2.3 allows a dealer to share multiple secrets encoded in a single polynomial, we want to extend PPVSS to Packed Pre-Constructed Publicly Verifiable Secret Sharing where multiple secrets are encoded in a single polynomial. We remark that the definition of PPVSS naturally extends to 3PVSS where a commitment to the secret is replaced by a vector of commitments of the secrets or a single commitment that represents all secrets. In the latter case where a single commitment is used, we will detail the assumptions due to which our protocols can remain secure. In the subsequent sections we will provide our two constructions for 3PVSS based on packed secret sharing scheme 2.3 along with the security guarantees these schemes can achieve.

A general construction of Shamir based PPVSS is given in [4], and we extend this idea to give two Packed Shamir based 3PVSS outlined in figures 3.1 and 3.2. Subsequently, we will give two practical 3PVSS schemes.

PPPVSS based on Packed Shamir secret sharing 2.3

Initialization: All parties $\{P_i\}_{i=1}^n$ and dealer D agree on the prime field $\mathbb{Z}_q$. Also, each party P_i registers their public key PK_i in the public ledger and all agree on a set of commitment keys or public keys, $\{h_j \text{ or } PK_j\}_{j=-(\ell-1)}^0$, whose secret keys are known to target entities with the cipher text space being a group $(\mathcal{G}, \times)$. More importantly, all entities agree on two NIZK proof protocols, one for the dealer during sharing phase π_{PPPVSS}^{share} and the other for Pessimistic reconstruction phase π_{PPPVSS}^{pes}.

Share:

- Dealer D samples a $t+\ell$-degree polynomial $f \in \mathbb{Z}_q[X]$ uniformly at random and sets $\{f(j) = f_j\}_{j=-(\ell-1)}^0$ as the set of all secrets.
- For each $1 \leq i \leq n$, D computes $f(i) = f_i$ and and encrypts it with PK_i to obtain $y_i = Enc(PK_i, f_i)$. D also encrypts(/commits) to the secrets $\{f_j\}_{j=-(\ell-1)}^0$ using the encryption(/commitment) keys $\{h_j \text{ or } PK_j\}_{j=-(\ell-1)}^0$ to obtain $\{y_j\}_{j=-(\ell-1)}^0$.
- D uses the agreed upon proof system π_{PPPVSS}^{share} to prove that they have encrypted the shares and commitments(/encryptions) of ℓ secrets corresponding to the correct polynomial that evaluates to ℓ secrets.
- D broadcasts $\{y_{-(\ell-1)}, \ldots, y_0, \ldots, y_n, \pi_{PPPVSS}^{share}\}$.

Verification: Given public keys $\{PK_i\}_{i=1}^n$ and commitment(/public) keys $\{h_j, PK_j\}_{j=-(\ell-1)}^0$, any entity can check π_{PPPVSS}^{share} to verify the correctness of the encrypted shares $\{y_i\}_{i=1}^n$ and $\{y_j\}_{j=-(\ell-1)}^0$ being the commitments(/encryptions) of the ℓ secrets, and outputs **true** or **false** accordingly.

Reconstruction: Similar to [4], there are two approaches to reconstruct the secrets $\{f_j\}_{j=-(\ell-1)}^0$ based on the cooperation of the dealer D which are as follows:

- **Optimistic Reconstruction:** D publishes $\{f_j\}_{j=-(\ell-1)}^0$, then any verifier (not necessarily shareholders) when given $\{y_j, h_j \text{ or } PK_j\}_{j=-(\ell-1)}^0$ can check if each y_j is a valid commitment(/encryption) of the secret f_j and returns **false** if the check fails; otherwise returns $\{f_j\}_{j=-(\ell-1)}^0$.
- **Pessimistic Reconstruction:** If D refuses to reveal $\{f_j\}_{j=-(\ell-1)}^0$, then any set $\mathcal{Q}$ consisting at least $t+\ell$ shareholders will do the following:
  - Each party $P_i \in \mathcal{Q}$ decrypts their share y_i using their private key SK_i corresponding to PK_i to obtain f_i and then they publish f_i along with proof π_{PPPVSS}^{pes} which proves that the f_i is the correct decryption of y_i.
  - If $\mathcal{Q}$ consists at least $t+\ell$ honest parties, they can use the lagrange interpolation to compute the secrets $\{f_j\}_{j=-(\ell-1)}^0$ as follows:

    $$f_j = \textstyle\sum_{i\in\mathcal{Q}} f_i(\prod_{k\in\mathcal{Q},k\neq i} \frac{-k-j}{i-j}), j \in \{-(\ell-1), \ldots, 0\}$$

    where $\lambda_i = \prod_{k\neq i} \frac{k}{k-i}$ are lagrange coefficients.

Figure 3.1: PPPVSS based on Packed Shamir secret sharing, version 1

PPPVSS based on Packed Shamir secret sharing 2.3

Initialization: All parties $\{P_i\}_{i=1}^n$ and dealer D agree on the prime field $\mathbb{Z}_q$. Also, each party P_i registers their public key PK_i in the public ledger and all agree on a set of commitment keys or public keys, $\{h_j \text{ or } PK_j\}_{j=-(\ell-1)}^0$, whose secret keys are known to target entities with the cipher text space being a group $(\mathcal{G}, \times)$. More importantly, all entities agree on two NIZK proof protocols, one for the dealer during sharing phase π_{PPPVSS}^{share} and the other for Pessimistic reconstruction phase π_{PPPVSS}^{pes}.

Share:

- Dealer D samples a $t+\ell$-degree polynomial $f \in \mathbb{Z}_q[X]$ uniformly at random and sets $\{f(j) = f_j\}_{j=-(\ell-1)}^0$ as the set of all secrets.
- For each $1 \leq i \leq n$, D computes $f(i) = f_i$ and encrypts it with PK_i to obtain $y_i = Enc(PK_i, f_i)$. D also encrypts(/commits) to the secrets $\{f_j\}_{j=-(\ell-1)}^0$ using the encryption(/commitment) keys $\{h_j \text{ or } PK_j\}$ and multiplies them together to obtain y_0.
- D uses the agreed upon proof system π_{PPPVSS}^{share} to prove that they have encrypted the shares and committed to the ℓ secrets corresponding to the correct polynomial that evaluates to ℓ secrets.
- D broadcasts $\{y_0, \ldots, y_n, \pi_{PPPVSS}^{share}\}$.

Verification: Given public keys $\{PK_i\}_{i=1}^n$ and commitment(/public) keys $\{h_j, PK_j\}_{j=-(\ell-1)}^0$, any entity can check π_{PPPVSS}^{share} to verify the correctness of the encrypted shares $\{y_i\}_{i=1}^n$ and y_0 being the commitment of the ℓ secrets, and outputs **true** or **false** accordingly.

Reconstruction: Similar to [4], there are two approaches to reconstruct the secrets $\{f_j\}_{j=-(\ell-1)}^0$ based on the cooperation of the dealer D which are as follows:

- **Optimistic Reconstruction:** D publishes $\{f_j\}_{j=-(\ell-1)}^0$, then any verifier (not necessarily shareholders) when given $\{h_j \text{ or } PK_j\}_{j=-(\ell-1)}^0, y_0$ can check if y_0 is the valid product of the commitment(/encryptions) of the secrets $\{f_j\}_{j=-(\ell-1)}^0$ and returns **false** if the check fails; otherwise returns $\{f_j\}_{j=-(\ell-1)}^0$.
- **Pessimistic Reconstruction:** If D refuses to reveal $\{f_j\}_{j=-(\ell-1)}^0$, then any set $\mathcal{Q}$ consisting at least $t+\ell$ shareholders will do the following:
  - Each party $P_i \in \mathcal{Q}$ decrypts their share y_i using their private key SK_i corresponding to PK_i to obtain f_i and then they publish f_i along with proof π_{PPPVSS}^{pes} which proves that the f_i is the correct decryption of y_i.
  - If $\mathcal{Q}$ consists at least $t+\ell$ honest parties, they can use the lagrange interpolation to compute the secrets $\{f_j\}_{j=-(\ell-1)}^0$ as follows:

$$f_j = \sum_{i \in \mathcal{Q}} f_i \left[\prod_{k \in \mathcal{Q}, k \neq i} \frac{-k-j}{i-j} \right] \qquad , j \in \{-(\ell-1), \ldots, 0\}$$

    where $\lambda_i = \prod_{k \neq i} \frac{k}{k-i}$ are lagrange coefficients.

FIGURE 3.2: PPPVSS based on Packed Shamir secret sharing, version 2

3.1 First Practical 3PVSS scheme

We present our first practical 3PVSS scheme in figure 3.3. Λ'_{RO} is a direct extension of Λ_{RO} [4] (which originally is based on Π_S [3]) where we publish individual commitments to the secrets. Similar to the binary e-voting protocol in [4], we believe that this scheme is useful in some e-voting protocols where there are more than two candidates to be voted for.

Theorem 3.1.1. *Assuming Polynomial Discrete Logarithm (PDL) and Decisional Diffie-Hellman (DDH) are computationally hard, Λ'_{RO} is secure against any static computationally bounded adversary. Moreover, if n is the number of parties then the security of Λ'_{RO} holds whenever $n \geq 2t + \ell$ where t is the maximum number of adversarially corrupted parties and $\ell \geq 1$ is the number of secrets to be shared.*

Proof. This directly follows from the proof of Λ_{RO} in [4] as we mainly make use of π_{PDL} and π_{DLEQ}. For the sake of completeness, we give the proof as follows:

- **Correctness**: Assume the dealer D and parties $\{P_i\}_{i=1}^n$ follow the protocol. Given a setup from initial phase of the protocol for an input 1^λ we have $(y_{-(\ell-1)}, \ldots, y_n, (d, z(X)))$ from the sharing phase of the Λ'_{RO} where $y_i = (h_j \text{ or } PK_j)^{f_j}$ for $-(\ell-1) \leq j \leq 0$ and $y_i = PK_i^{f_i}$ for $1 \leq i \leq n$, d is an output from the random oracle $\mathcal{H}$ for the input $(y_{-(\ell-1)}, \ldots, y_n, c_{-(\ell-1)}, \ldots, c_n)$ and $z(X) = r(x) + df(X)$ is a $t + \ell-$degree polynomial such that $f(X)$ is the secret polynomial, $r(X)$ is the polynomial (also secret to D) chosen uniformly at random, $c_i = (h_j \text{ or } PK_j)^{r(j)}$ for $-(\ell-1) \leq j \leq 0$ and $c_i = PK_i^{r(i)}$ for $1 \leq i \leq n$.

  Now with $(y_{-(\ell-1)} \ldots, y_n, (d, z(X)))$, a verifier checks that the following part of the verification phase evaluates to d when additionally given with ℓ public keys of target entities for the secrets and n public keys for the shares:

$$\begin{aligned}
&\mathcal{H}(y_{-(\ell-1)} \ldots, y_n, \frac{PK_{-(\ell-1)}^{z(-(\ell-1))}}{y_{-(\ell-1)}^d}, \ldots, \frac{PK_n^{z(n)}}{y_n^d}) \\
=&\mathcal{H}(y_{-(\ell-1)}, \ldots, y_n, \frac{PK_{-(\ell-1)}^{r(-(\ell-1))+df(-(\ell-1))}}{PK_{-(\ell-1)}^{df(-(\ell-1))}}, \ldots, \frac{PK_n^{r(n)+df(n)}}{PK_n^{df(n)}}) \\
=&\mathcal{H}(y_{-(\ell-1)}, \ldots, y_n, PK_{-(\ell-1)}^{r(-(\ell-1))}, \ldots, PK_n^{r(n)}).
\end{aligned}$$

  Verifier computes d even when up to ℓ commitment keys for secrets are used instead of the public keys of the target entities for the secrets.

  Moreover, the reconstruction phase always yields $\{g^{f_j}\}_{j=-(\ell-1)}^0$ in both the approaches. Explicitly, it is clear in the optimistic phase that yields $\{g^{f_j}\}_{j=-(\ell-1)}^0$ after one checks $\{y_j = (h_i \text{ or } PK_j)^{f_j}\}$ for $-(\ell-1) \leq j \leq n$, when given $(g, \{y_j, f_j, h_j \text{ or } PK_j\}_{j=-(\ell-1)}^0)$. The Pessimistic case also yields $\{g^{f_j}\}_{j=-(\ell-1)}^0$ which inherently is the reconstruction step from the PVSS Π_S [3]. In essence,

we just proved that if the dealer and parties follow the protocol, then verification step returns true and the Reconstruction phase returns the actual secrets.

- **Verifiability**: We need to show that if verify algorithm returns true then for $1 \leq i \leq n$, y_i is a valid encryption of i'th share of the secrets $\{g^{f_j}\}_{j=-(\ell-1)}^{0}$ which is intended to party P_i for $1 \leq i \leq n$ and for $-(\ell-1) \leq j \leq 0$, y_j is the valid commitment(/encryption) of the secret g^{f_j} and reconstruct algorithm returns $\{g^{f_j}\}_{j=-(\ell-1)}^{0}$ irrespective of the approach with high probability. The trick to do that is to leverage the special soundness property of underlying proof system.

  Assuming verification phase outputs true, consider two acceptable transcripts $(c_{-(\ell-1)}, \ldots, c_n, d, z(X))$ of the interactive version of the NIZK argument for PDL problem output by the dealer during the sharing phase as follows (WLOG assume the secrets are encrypted using the ℓ encryption keys of some target entities).

$$PK_i^{z(i)} = c_i y_i^d, PK_i^{z'(i)} = c_i y_i^{d'}, \text{ for } -(\ell-1) \leq i \leq n;$$

  this implies,

$$PK_i^{z(i)-z'(i)} = y_i^{d-d'} \iff PK_i^{\frac{z(i)-z'(i)}{d-d'}} = y_i \text{ for } -(\ell-1) \leq i \leq n.$$

  If all $n \geq t+\ell$ checks for verification pass, then it means that the set $\{y_i\}_{-(\ell-1)}^{n}$ consists valid encryptions(/commitments) of ℓ secrets and encryptions of n shares, as a consequence if any set $\mathcal{Q}$ of $t+\ell$ honest parties decrypt the encryptions of their shares to reconstruct secrets via pessimistic approach, then the following is obtained,

$$g^{\frac{z(i)-z'(i)}{d-d'}} = F_i \text{ for } i \in \mathcal{Q}, |\mathcal{Q}| = t+\ell.$$

  After the DLEQ proofs π_{DLEQ} are checked, as with high probability $z(X), z'(X)$ are exactly $t+\ell-1$ degree polynomial, an extractor having access to the two acceptable transcripts can construct the secrets from $f_i = \frac{z(i)-z'(i)}{d-d'}$ for $i \in \mathcal{Q}$ which are $t+\ell$ evaluations of the $t+\ell-1$-degree polynomial $f(X) = \frac{z(X)-z'(X)}{d-d'}$. This implies that pessimistic approach yields the actual secrets $\{g^{f_j}\}_{j=-(\ell-1)}^{0}$ with high probability.

  Similarly, the optimistic approach also yields the actual secrets $\{g^{f_j}\}_{j=-(\ell-1)}^{0}$ with high probability as a consequence of special soundness of the NIZK argument for PDL problem deployed in our 3PVSS.

- **IND1-Secrecy**: We will prove that our scheme will not leak any information about the secrets to any static computationally bounded adversary $\mathcal{A}$ who

corrupts at most t parties and has access to their secret keys, all public information except the secret keys of target entities, i.e., $\{SK_j\}_{j=-(\ell-1)}^{0}$, assuming the DDH assumption is true.

In a nutshell, let $s_0 = (g^{f_{-(\ell-1)}}, \ldots, g^{f_0})$ and $s_1 = (g^{f'_{-(\ell-1)}}, \ldots, g^{f'_0})$ be the tuples of secrets in $\mathbb{Z}_q$ chosen at random by the IND1-Secrecy challenger. The challenger runs the setup phase of our 3PVSS scheme Λ'_{RO} to obtain the public keys $\{PK_j\}_{j=-(\ell-1)}^{0}$ for encrypting the secrets and the public keys $\{PK_i\}_{i=1}^{n}$ to encrypt the shares meant for shareholders P_i for $1 \leq i \leq n$. The challenger runs $share(n, t+\ell-1, s_0, \{PK_i\}_{i=-(\ell-1)}^{n})$ and obtains $(y_{-(\ell-1)}, \ldots, y_n, \pi_{3PVSS}^{share})$ as the encrypted secrets and shares and the proof of correctness of the sharing phase. The challenger chooses $b \in \{0, 1\}$ uniformly at random and sends $s_b, (y_{-(\ell-1)}, \ldots, y_n, \pi_{3PVSS}^{share})$ along with the public information generated in the sharing phase to $\mathcal{A}$. *Without loss of generality, let $f_i = f'_i$ for $-(\ell-1) \leq i \leq -1$, i.e., $\mathcal{A}$ is trying to distinguish between two tuples of secrets where they may differ only in last position.*

By contradiction, assume $\mathcal{A}$ can break the IND1-Secrecy property of our scheme. Then we will show that there exists an adversary $\mathcal{B}$ who can break the DDH assumption using $\mathcal{A}$ as a subroutine. Also, without loss of generality assume $\mathcal{A}$ corrupts first t shareholders, i.e., corrupts $P_1, \ldots, P_t$ so it knows their secret keys.

Let $\mathcal{B}$ be an adversary who is given a DDH instance $(h, h^\alpha, h^\beta, h^\gamma)$, where $\alpha, \beta, \gamma \in \mathbb{Z}_q$ with α and β being non-zero. Now $\mathcal{B}$ using $\mathcal{A}$ simulates the IND1 game $\mathcal{A}$ as follows:

- $\mathcal{B}$ sets $g = h^\alpha$.
- For $t+1 \leq i \leq n$ and $-(\ell-1) \leq i \leq 0$, $\mathcal{B}$ selects $u_i \in \mathbb{Z}_q$ (implicitly defines $s_i = \frac{u_i}{\alpha}$) uniformly at random and sends $PK_i = h^{u_i}$ to $\mathcal{A}$.
- For $1 \leq i \leq t$, $\mathcal{A}$ selects $SK_i \in \mathbb{Z}_q$ uniformly at random and sends $PK_i = g^{SK_i}$ to $\mathcal{B}$ (challenger in the perspective of $\mathcal{A}$).
- For $1 \leq i \leq t$ and $-(\ell-1) \leq i \leq -1$, $\mathcal{B}$ selects $f_i \in \mathbb{Z}_q$ uniformly at random, sets $v_i = h^{f_i}$ and $y_i = PK_i^{f_i}$. Finally, for $i = 0$, $\mathcal{B}$ sets $v_0 = h^\beta$ and $y_0 = v_0^{u_0}$, i.e., internally she assumes $f_0 = \beta$ but note that she does not know f_0 explicitly.
- For $t+1 \leq i \leq n$, $\mathcal{B}$ generates $v_i = h^{p(i)}$ via lagrange interpolation in the exponent, where $p \in \mathbb{Z}_q[X]_{t+\ell-1}$ which is determined by the $t+\ell$ evaluations, namely, $p(0) = f_0 = \beta$, $p(i) = f_i$ for $-(\ell-1) \leq i \leq -1$ and $1 \leq i \leq t$. After obtaining v_i, $\mathcal{B}$ sets $y_i = v_i^{u_i}$ for $t+1 \leq i \leq n$.
- Now that $\mathcal{B}$ has $\{PK_i, y_i\}_{i=-(\ell-1)}^{n}$, as a simulator she samples a random $t+\ell-1$ degree polynomial $z'(X)$ and sets $c'_i = \frac{PK_i^{z'(i)}}{y_i^d}$ when the challenge value d in the interactive version of the proof scheme is given.
- $\mathcal{B}$ sets h^γ as the last element in s_b.

– Finally, $\mathcal{B}$ sends $\{y_{-(\ell-1)}, \ldots, y_n, c'_{-(\ell-1)}, \ldots, c'_n, d, z'(X)\}$ along with s_b and public information to $\mathcal{A}$, simulating the role of a IND1-Secrecy challenger.

– $\mathcal{A}$ outputs the bit b' as its guess for b. If $b' = 0$ then $\mathcal{B}$ guesses $\gamma = \alpha.\beta$, otherwise γ is guessed to be a random element in $\mathbb{Z}_q$.

□

Note that our aforementioned 3PVSS scheme Λ'_{RO} is exactly Λ_{RO} if the number of secrets is only one. Though our scheme can be applicable in some e-voting protocols, it may not be suitable for other applications, such as randomness beacons whose goal is to distributively and securely generate fresh unbiased randomness. In the next section, we present an alternative 3PVSS scheme which is more suitable for randomness beacons.

Λ'_{RO}

Initialization: All parties $\{P_i\}_{i=1}^n$ and dealer D agree on the prime field $\mathbb{Z}_q$, a group $(\mathbb{G}, \times)$ of order q with a generator g, random oracle $\mathcal{H}$. Also, each party P_i registers their public key PK_i in the public ledger and all agree on a set of commitment keys or public keys, $\{h_j \text{ or } PK_j\}_{j=-(\ell-1)}^0$, whose secret keys are known to target entities.

Share:

- Dealer D samples a $t+\ell$-degree polynomial $f \in \mathbb{Z}_q[X]$ uniformly at random and sets $\{g^{f_j} : f_j = f(j)\}_{j=-(\ell-1)}^0$ as the set of all secrets.
- For each $1 \le i \le n$, D computes the encryptions of g^{f_i} with PK_i using $f(i) = f_i$ to obtain $y_i = PK_i^{f_i}$. D also commits(/encrypts) to the secrets $\{g^{f_j}\}_{j=-(\ell-1)}^0$ using the encryption(/commitment) keys $\{h_j \text{ or } PK_j\}_{j=-(\ell-1)}^0$ to obtain $\{y_j = (h_j \text{ or } PK_j)^{f_j}\}_{j=-(\ell-1)}^0$.
- D uses a modified PDL proof scheme 2.4.2 to generate π_{share} as follows:
  - Samples a $t+\ell-$degree polynomial $r \in \mathbb{Z}_q[X]$ uniformly at random and computes $\{c_j = (h_j \text{ or } PK_j)^{r(j)}\}_{j=-(\ell-1)}^0$ and $c_i = PK_i^{r(i)}$ for $1 \le i \le n$.
  - Using $\mathcal{H}$, $d = \mathcal{H}(y_{-(\ell-1)} \dots, y_n, c_{-(\ell-1)}, \dots, c_n)$ is computed.
  - Sets $z(X) = r(X) + df(X)$, hence $\pi_{share} = (d, z(X))$ is obtained.
- D broadcasts the encryptions of the shares along with the commitment (or product of the encryptions) of the secrets with π_{PDL} which proves the validity of the encrypted shares and committed (/encrypted) secrets, i.e., broadcasts $\{y_i\}_{i=-(\ell-1)}^n$ and $(d, z(X))$.

Verification: Given public keys $\{PK_i\}_{i=1}^n$ and commitment(/public) keys $\{h_j \text{ or } PK_j\}_{j=-(\ell-1)}^0$, any entity can check π_{PDL} to verify the correctness of the encrypted shares $\{y_i\}_{i=1}^n$ and $\{y_j\}_{j=-(\ell-1)}^0$ being the commitments(/encryptions) of the secrets. They will output **true** or **false** based on the verification of the proof. The procedure is outlined as follows:

- The entity checks if $z(X)$ is a $t+\ell-$degree polynomial or not.
- Checks if $d = \mathcal{H}(y_{-(\ell-1)}, \dots, y_n, \frac{(h_{-(\ell-1)} \text{ or } PK_{-(\ell-1)})^{z(-(\ell-1))}}{y_{-(\ell-1)}^d}, \dots, \frac{PK_n^{z(n)}}{y_n^d})$.
- Outputs **true** if both of the above checks are satisfied, otherwise **false**.

Reconstruction: Similar to [4], there are two approaches to reconstruct the secrets $\{g^{f_j}\}_{j=-(\ell-1)}^0$ based on the cooperation of the dealer D which are as follows:

- **Optimistic Reconstruction:** D publishes $\{f_j\}_{j=-(\ell-1)}^0$, then any verifier (not necessarily a shareholder) when given $g, \{h_j \text{ or } PK_j\}_{j=-(\ell-1)}^0, y_{-(\ell-1)}, \dots, y_0$ can check if $\{y_j = h_j^{f_j} \text{ or } PK_j^{f_j}\}_{j=-(\ell-1)}^0$ and returns $\{g^{f_j}\}_{j=-(\ell-1)}^0$ if all checks pass, if not they return **false**.
- **Pessimistic Reconstruction:** If D refuses to reveal $\{f_j\}_{j=-(\ell-1)}^0$, then any set $\mathcal{Q}$ consisting at least $t+\ell$ shareholders will do the following:
  - Each party $P_i \in \mathcal{Q}$ decrypts their share y_i using their private key SK_i corresponding to PK_i to obtain g^{f_i} and then they publish g^{f_i} along with a DLEQ proof 2.4.1, π_{DLEQ} which proves that g^{f_i} is the correct decryption of y_i.
  - If $\mathcal{Q}$ consists at least $t+\ell$ honest parties, they can use the lagrange interpolation to compute the secrets $\{g^{f_j}\}_{j=-(\ell-1)}^0$ as follows:

$$g^{f_j} = \prod_{i \in \mathcal{Q}} (g^{f_i})^{\prod_{k \in \mathcal{Q}, k \neq i} \frac{-k-j}{i-j}} = g^{\sum_{i \in \mathcal{Q}} f_i \prod_{k \in \mathcal{Q}, k \neq i} \frac{-k-j}{i-j}}.$$

Figure 3.3: Λ'_{RO}, a packed version of Λ_{RO} [4]

3.2 More efficient 3PVSS scheme

One might have already noticed that the number of commitments increases linearly with the increase in the number of secrets in our first 3PVSS scheme presented in the previous section 3.1. As a consequence, the computation cost in the verification phase also increases. In this section, we present a more efficient 3PVSS scheme which is efficient both in terms of communication and overall computation cost. We denote this scheme as Λ_{RO}^{packed} which is presented in figure 3.5. The main idea is to have a single representation of the secrets and still remain secure against static computationally bounded adversaries. In order to show that our new optimized construction is secure we will require a new NIZK AoK protocol which is discussed in the upcoming subsection 3.2.1.

3.2.1 A NIZK AoK protocol

Consider the set $\{g_i\}_{i=-(\ell-1)}^{n}$ containing $n+\ell$ distinct generators (obtained as CRS) of the prime order q cyclic group $\mathcal{G}$ different from g with g also being a generator of $\mathbb{G}$, where $n+\ell < \varphi(q)$ and $\varphi(q)$ is the total number of distinct generators of the cyclic group $\mathbb{G}$. Consider the following relation for some polynomial $f \in \mathbb{Z}_q[X]_{t+\ell-1}$ with $t+\ell \leq n$:

$$R_{mod-PDL} = \{(g_{-(\ell-1)}, \ldots, g_n, g, x_{-(\ell-1)}, \ldots, x_n, F_0, \ldots, F_n), f(X) : F_0 = \prod_{i=-(\ell-1)}^{0} g_i^{f(x_i)}, F_i = g_i^{f(x_i)}, 1 \leq i \leq n\}, \quad (3.1)$$

where all x_i's are distinct in $\mathbb{Z}_q$. The $R_{mod-PDL}$ is based on *modified*-PDL (inspired from the PDL in [3]) which is defined in the following.

Definition 3.2.1 (*modified*- Polynomial Discrete Logarithm problem)**.** *Let* $\{g_i : g_i \neq g\}_{i=-(\ell-1)}^{n}$ *be a set of random distinct generators for the prime order* q *cyclic group* $\mathbb{G}$ *generated by* g*. Given* $F_0, \ldots, F_n$ *and distinct elements* $x_{-(\ell-1)}, \ldots, x_n$ *in* $\mathbb{Z}_q$*, find a polynomial* $f \in \mathbb{Z}_q[X]$ *with degree at most* $t+\ell-1$*, where* $F_0 = \prod_{i=-(\ell-1)}^{0} g_i^{f(x_i)}$ *and* $F_i = g_i^{f(x_i)}$ *for* $1 \leq i \leq n$ *with* $t+\ell \leq n$*.*

In other words, an algorithm $\mathcal{A}$ *is said to have an advantage* ϵ *in solving modified-PDL if*

$$Pr[\mathcal{A}(x_{-(\ell-1)}, \ldots, x_n, g, g_{-(\ell-1)}, \ldots, g_n, \prod_{i=-(\ell-1)}^{0} g_i^{f(x_i)}, g_1^{f(x_1)}, \ldots, g_n^{f(x_n)})] \geq \epsilon,$$

where $f \in \mathbb{Z}_q[X]$ *is at most a* $t+\ell-1$ *degree polynomial with* $t \leq n$ *and the probability is over distinct generators* $g, g_{-(\ell-1)}, \ldots, g_n$ *of* $\mathbb{G}$ *chosen at random and distinct* $x_{-(\ell-1)}, \ldots, x_n$ *elements in* $\mathbb{Z}_q$*.*

It is intuitive to observe that the *modified*-PDL problem can be reduced to the Discrete Logarithm (DL) problem. We will prove this via contradiction. More

$\pi^{AoK}_{mod-PDL}$

Let $(g, x_{-(\ell-1)}, \ldots, x_n, F_0, \ldots, F_n) \in L^{mod-PDL}_{g_{-(\ell-1)},\ldots,g_n}$ with its corresponding witness being $f \in \mathbb{Z}_q[X]_{\leq t+\ell-1}$ where $L^{mod-PDL}_{g_{-(\ell-1)},\ldots,g_n}$ is the CRS dependent language defined by the relation $R_{mod-PDL}$ and $\mathcal{H}$ be a random oracle (RO).

Prover

- Samples $r \in_R \mathbb{Z}_q[X]_{t+\ell-1}$ uniformly at random and sets $\Gamma_0 = \prod_{j=-(\ell-1)}^{0} g_j^{r(x_j)}$ and $\Gamma_i = g_i^{r(x_i)}$ for $1 \leq i \leq n$.
- Sets $d \leftarrow \mathcal{H}(F_0, \ldots, F_n, \Gamma_0, \ldots, \Gamma_n)$, where $\mathcal{H}$ is an agreed upon Random Oracle (RO).
- Sets $z(X) \equiv r(X) + df(X) \pmod q$ and returns the proof(/transcript) $\pi := (d, z(X))$.

Verifier

- First checks if z is a $t + \ell - 1$ degree polynomial in $\mathbb{Z}_q[X]$. If so, they proceed with the next step.
- Checks if $d \leftarrow \mathcal{H}(F_0, \ldots, F_n, \frac{\prod_{i=-(\ell-1)}^{0} g_i^{z(x_i)}}{F_0^d}, \frac{g_1^{z(x_1)}}{F_1^d}, \ldots, \frac{g_n^{z(x_n)}}{F_n^d})$.
- If first two steps are correct then they output **true**, otherwise **false**.

Figure 3.4: A NIZK AoK for the *modified* Polynomial DL

explicitly, let there exist an adversary $\mathcal{A}$ who can solve the *modified*-PDL problem. We will now construct an adversary $\mathcal{B}$ who can solve DL problem using $\mathcal{A}$ as a subroutine. Let $(g, h := g^f)$ be the challenge for the DL problem, now $\mathcal{B}$ will set $F_1 = h$, $x_i = i$ for $-(\ell-1) \leq i \leq n$ and sets $F_0, F_2, \ldots, F_n$ to be n random elements in the group $\mathbb{G}$. $\mathcal{B}$ will send $(g_{-(\ell-1)}, \ldots, g_n, g, -(\ell-1), \ldots, n, F_0, \ldots, F_n)$ to $\mathcal{A}$. If $\mathcal{A}$ outputs a polynomial $f \in \mathbb{Z}_q[X]_{t+\ell-1}$ such that $F_0 = \prod_{j=-(\ell-1)}^{0} g_j^{f(j)}$ and $F_i = g_i^{f(i)}$ for $1 \leq i \leq n$, then $\mathcal{B}$ can send $f(1)$ to the DL challenger.

Now consider the CRS dependent language of $R_{mod-PDL}$ as follows:

$$L^{mod-PDL}_{g_{-(\ell-1)},\ldots,g_n} = \{(g, x_{-(\ell-1)}, \ldots, x_n, F_0, \ldots, F_n) : \exists f \in \mathbb{Z}_q[X]_{t+\ell-1}, \\ [(g_{-(\ell-1)}, \ldots, g_n, g, x_{-(\ell-1)}, \ldots, x_n, F_0, \ldots, F_n), f(X)] \in R_{mod-PDL}\}.$$

We will now present a new NIZK AoK $\pi^{AoK}_{mod-PDL}$ for the aforementioned relation $R_{mod-PDL}$ in the figure 3.4.

The following theorem states that in fact $\pi^{AoK}_{mod-PDL}$ is a NIZK AoK for the relation $R_{mod-PDL}$ under certain conditions.

Theorem 3.2.1 (A NIZK Argument of Knowledge for $R_{mod-PDL}$)**.** *Consider* $(g, x_{-(\ell-1)}, \ldots, x_n, F_0, \ldots, F_n) \in L^{mod-PDL}_{g_{-(\ell-1)},\ldots,g_n}$, *where* $L^{mod-PDL}_{g_{-(\ell-1)},\ldots,g_n}$ *is the CRS dependent language defined by the relation* $R_{mod-PDL}$, *with its corresponding witness being* $f \in \mathbb{Z}_q[X]_{\leq t+\ell-1}$. *Assuming modified-PDL is computationally hard, for* $t + \ell \leq n$, *the protocol* $\pi^{AoK}_{mod-PDL}$ *(described in figure 3.4) is a NIZK AoK for* $R_{mod-PDL}$ *in the RO model.*

Proof. The corresponding proof is similar to the proof of theorem 2.4.1 given in [3]. Formally, we will prove the security of the interactive setting (i.e., without *RO* being used) and then using Fiat-Shamir transform it can be extended for the non-interactive setting in the *RO* model.

- ***Correctness***: If both prover and verifier are honest, then for $1 \leq i \leq n$ we have

$$\begin{aligned} g_i^{z(i)} &= g_i^{r(i)+df(i)} \\ &= g_i^{r(i)}(g^{f(i)})^d \\ &= \Gamma_i F_i^d, \end{aligned} \tag{3.2}$$

  and

$$\begin{aligned} \prod_{j=-(\ell-1)}^{0} g_j^{z(j)} &= \prod_{j=-(\ell-1)}^{0} g_j^{r(j)+df(j)} \\ &= (\prod_{j=-(\ell-1)}^{0} g_j^{r(j)})(\prod_{j=-(\ell-1)}^{0} g_j^{f(j)})^d \\ &= \Gamma_0 F_0^d. \end{aligned} \tag{3.3}$$

  The aforementioned equations imply that verification returns *true* and honest verifier accepts the honest prover.

- ***Special Soundness***: Let $(\Gamma_0, \ldots, \Gamma_n, d, z(X))$, $(\Gamma_0, \ldots, \Gamma_n, d', z'(X))$ be two acceptable transcripts where response polynomials will differ as a consequence of different challenge values. For $1 \leq i \leq n$ we have the following from equation 3.2:

$$g_i^{z(x_i)} = \Gamma_i F_i^d, g_i^{z'(x_i)} = \Gamma_i F_i^{d'};$$

  which implies

$$g_i^{z(x_i)-z'(x_i)} = F_i^{d-d'} \iff g_i^{\frac{z(x_i)-z'(x_i)}{d-d'}} = F_i, \tag{3.4}$$

[*if and only if because* $d - d'$ *is always invertible in modulo prime* q *whenever* $d \neq d' \pmod q$]. Similarly, we have the following from equation 3.3:

$$\prod_{j=-(\ell-1)}^{0} g_j^{z(x_j)} = \Gamma_0 F_0^d, \quad \prod_{j=-(\ell-1)}^{0} g_j^{z'(x_j)} = \Gamma_0 F_0^{d'};$$

implying

$$\prod_{j=-(\ell-1)}^{0} g_j^{z(x_j)-z'(x_j)} = F_0^{d-d'} \iff \prod_{j=-(\ell-1)}^{0} g_j^{\frac{z(x_j)-z'(x_j)}{d-d'}} = F_0. \tag{3.5}$$

As we have $n \geq t+\ell$, information from equation 3.4 is enough for an extractor to extract a unique witness polynomial f for the given statement in $R_{mod-PDL}$ as it implies that $f_i = \frac{z(x_i)-z'(x_i)}{d-d'}$ for $1 \leq i \leq n$. More explicitly, as $z(X)$ is a $t+\ell-1$ degree polynomial with high probability in $\mathbb{Z}_q[X]$, an extractor $\mathcal{E}$ can construct the unique $t+\ell-1$-degree polynomial $f \in \mathbb{Z}_q[X]$, being the desired witness (resp. solution) for a given statement in $R_{mod-PDL}$ relation (resp. *modified*-PDL problem), from any $t+\ell$ evaluation points in $\{f_i\}_{i=1}^n$ whenever $n \geq t+\ell$.

- ***Honest Verifier Zero Knowledge (HVZK)***: Given the statement $(g, x_{-(\ell-1)}, \ldots, x_n, F_0, \ldots, F_n) \in L^{mod-PDL}_{g_{-(\ell-1)},\ldots,g_n}$ and a challenge value d, a simulator $\mathcal{S}$ can choose a polynomial $z' \in \mathbb{Z}_q[X]_{t+\ell-1}$ uniformly at random and sets $\Gamma'_0 = \frac{\prod_{i=-(\ell-1)}^{0} g_i^{z'(x_i)}}{F_0^d}$ and $\Gamma'_i = \frac{g_i^{z'(x_i)}}{F_i^d}$ for $1 \leq i \leq n$. Now, $\mathcal{S}$ returns $(\Gamma'_0, \ldots, \Gamma'_n, z'(X))$ as the simulated proof. As $z(X)$ is a random degree $t+\ell-1$-polynomial in $\mathbb{Z}_q[X]$ and $z'(X)$ is chosen uniformly at random, the simulated proof of $\mathcal{S}$ is indistinguishable from the real one.

As the interactive scheme is public coin, satisfies *completeness*, (computational) *Special Soundness* and (computational) *HVZK*, then in the random oracle (*RO*) model, using Fiat-Shamir transform [12], it can be turned into a NIZK Argument of Knowledge for $R_{mod-PDL}$ (defined in equation 3.1). □

The importance of our NIZK AoK protocol $\pi^{AoK}_{mod-PDL}$ is that it can be used to prove the security of our new efficient 3PVSS scheme Λ^{packed}_{RO}. As a consequence, we give the security proof of Λ^{packed}_{RO} in the following.

Theorem 3.2.2. *Assuming Polynomial Discrete Logarithm (PDL) and Decisional Diffie-Hellman (DDH) are computationally hard,* Λ^{packed}_{RO} *is secure against any static computationally bounded adversary. Moreover, if* n *is the number of parties then the security of* Λ^{packed}_{RO} *holds whenever* $n \geq 2t+\ell$ *where* t *is the maximum number of adversarially corrupted parties and* $\ell \geq 1$ *is the number of secrets to be shared.*

Proof. We want to prove the *Correctness, Verifiability* and *IND1-Secrecy* properties and it follows directly as in the PPVSS Λ_{RO}.

- **Correctness**: Assume the dealer D and parties $\{P_i\}_{i=1}^n$ follow the protocol. We show that verification returns *true* and reconstruction of secrets in both approaches will return the intended secrets. Given a setup from initial phase of the protocol for an input 1^λ we have $(y_0, y_1, \ldots, y_n, (d, z(X)))$ from the sharing phase of the Λ_{RO}^{packed} where $y_0 = \prod_{j=-(\ell-1)}^{0}(h_j \text{ or } PK_j)^{f_j}$, $y_i = PK_i^{f_i}$ for $1 \leq i \leq n$, d is an output from the random oracle $\mathcal{H}$ for the input $(y_0, \ldots, y_n, c_0, \ldots, c_n)$ and $z(X) = r(x) + df(X)$ is a $t+\ell-$degree polynomial such that $f(X)$ is a secret polynomial, $r(X)$ is the polynomial (also secret to D) chosen uniformly at random, $c_0 = \prod_{j=-(\ell-1)}^{0}(h_j \text{ or } PK_j)^{r(j)}$ and $c_i = PK_i^{r(i)}$ for $1 \leq i \leq n$.

  Now with $(y_0, y_1, \ldots, y_n, (d, z(X)))$, a verifier does the following computation for verification using ℓ commitment (/encryption) keys:

$$\begin{aligned}
&\mathcal{H}(y_0, y_1, \ldots, y_n, \frac{\prod_{j=-(\ell-1)}^{0} h_j^{z(j)}}{y_0^d}, \frac{PK_1^{z(1)}}{y_1^d}, \ldots, \frac{PK_n^{z(n)}}{y_n^d}) \\
=&\mathcal{H}(y_0, y_1, \ldots, y_n, \prod_{j=-(\ell-1)}^{0} \frac{h_j^{r(j)+df(j)}}{h_j^{df(j)}}, \frac{PK_1^{r(1)+df(1)}}{PK_1^{df(1)}}, \ldots, \frac{PK_n^{r(n)+df(n)}}{h_n^{df(n)}}) \\
=&\mathcal{H}(y_0, y_1, \ldots, y_n, \prod_{j=-(\ell-1)}^{0} h_j^{r(j)}, PK_1^{r(1)}, \ldots, PK_n^{r(n)}).
\end{aligned}$$

  Hence, the computation will result d value and consequently verifier will return *true* as this is exactly verification step in $\pi_{mod-PDL}^{AoK}$.

  Moreover, the reconstruction phase always yields $\{g^{f_j}\}_{j=-(\ell-1)}^{0}$ in both the approaches. Explicitly, it is clear in the optimistic phase that yields $\{g^{f_j}\}_{j=-(\ell-1)}^{0}$ after one checks $y_0 = \prod_{j=-(\ell-1)}^{0}(h_j \text{ or } PK_j)^{f_j}$ when given $(g, \{h_j \text{ or } PK_j\}_{j=-(\ell-1)}^{0}, y_0, \{f_j\}_{j=-(\ell-1)}^{0})$. The Pessimistic case also yields $\{g^{f_j}\}_{j=-(\ell-1)}^{0}$ which inherently is the reconstruction step from the PVSS Π_S [3]. In essence, we just proved that if the dealer and parties follow the protocol, then verification step returns true and the Reconstruction phase returns the actual secrets.

- **Verifiability**: We will show that if verification algorithm returns true then with high probability y_i is a valid encryption of i'th share of the secrets $\{g^{f_j}\}_{j=-(\ell-1)}^{0}$ which is intended to party P_i for $1 \leq i \leq n$, y_0 is the commitment of ℓ secrets and reconstruct algorithm returns $\{g^{f_j}\}_{j=-(\ell-1)}^{0}$ irrespective of the two approaches.

  Without loss of generality, assume ℓ public keys, $\{PK_j\}_{j=-(\ell-1)}^{0}$ for encrypting secrets is given. Then, it is easy to observe that $(g, -(\ell-1), \ldots, n, y_0, \ldots, y_n) \in L_{PK_{-(\ell-1)}, \ldots, PK_n}^{mod-PDL}$, where $L_{PK_{-(\ell-1)}, \ldots, PK_n}^{mod-PDL}$ is the CRS dependent language defined by the relation $R_{mod-PDL}$ (defined in equation 3.1) and the witness is

$f \in \mathbb{Z}_q[X]_{\leq t+\ell-1}$. As a consequence, if the verification phase outputs true, then y_0 is a valid commitment of the secrets, $\{y_1, \ldots, y_n\}$ is a set of valid encryptions of the corresponding shares and *optimistic reconstruction* approach will yield the secrets $\{g^{f_j}\}_{j=-(\ell-1)}^{0}$. Also, due to the special soundness property of $\pi_{mod-PDL}^{AoK}$ and as $n \geq t + \ell$ any extractor can extract the unique witness polynomial when given two acceptable transcripts in the interactive setting of $\pi_{mod-PDL}^{AoK}$, which means that the pessimistic reconstruction approach also yields the same secrets as in optimistic approach, i.e., $\{g^{f_j}\}_{j=-(\ell-1)}^{0}$.

- **IND1-Secrecy**: We will prove that our scheme will not leak any information about the secrets to any static computationally bounded adversary $\mathcal{A}$ who corrupts at most t parties and has access to their secret keys, all public information except the secret keys of target entities, i.e., $\{SK_j\}_{j=-(\ell-1)}^{0}$, assuming the DDH assumption is true.

  In a nutshell, let $s_0 = (g^{f_{-(\ell-1)}}, \ldots, g^{f_0})$ and $s_1 = (g^{f'_{-(\ell-1)}}, \ldots, g^{f'_0})$ be the tuples of secrets in $\mathbb{Z}_q$ chosen at random by the IND1-Secrecy challenger. The challenger runs the setup phase of our 3PVSS scheme Λ_{RO}^{packed} to obtain the public keys $\{PK_j\}_{j=-(\ell-1)}^{0}$ for encrypting the secrets and the public keys $\{PK_i\}_{i=1}^{n}$ to encrypt the shares meant for shareholders P_i for $1 \leq i \leq n$. The challenger runs $share(n, t + \ell - 1, s_0, \{PK_i\}_{i=-(\ell-1)}^{n})$ and obtains $(y_0, \ldots, y_n, \pi_{3PVSS}^{share})$ as the encrypted secrets and shares and the proof of correctness of the sharing phase. The challenger chooses $b \in \{0, 1\}$ uniformly at random and sends $s_b, (y_0, \ldots, y_n, \pi_{3PVSS}^{share})$ along with the public information generated in the sharing phase to $\mathcal{A}$. *Without loss of generality, let $f_i = f'_i$ for $-(\ell-1) \leq i \leq -1$, i.e., $\mathcal{A}$ is trying to distinguish between two tuples of secrets where they may differ only in last position.*

  By contradiction, assume $\mathcal{A}$ can break the IND1-Secrecy property of our scheme. Then we will show that there exists an adversary $\mathcal{B}$ who can break the DDH assumption using $\mathcal{A}$ as a subroutine. Also, without loss of generality assume $\mathcal{A}$ corrupts first t shareholders, i.e., corrupts $P_1, \ldots, P_t$ so it knows their secret keys.

  Let $\mathcal{B}$ be an adversary who is given a DDH instance $(h, h^\alpha, h^\beta, h^\gamma)$, where $\alpha, \beta, \gamma \in \mathbb{Z}_q$ with α and β being non-zero. Now $\mathcal{B}$ using $\mathcal{A}$ simulates the IND1 game $\mathcal{A}$ as follows:

  - $\mathcal{B}$ sets $g = h^\alpha$.
  - For $t + 1 \leq i \leq n$ and $-(\ell - 1) \leq i \leq 0$, $\mathcal{B}$ selects $u_i \in \mathbb{Z}_q$ (implicitly defines $s_i = \frac{u_i}{\alpha}$) uniformly at random and sends $PK_i = h^{u_i}$ to $\mathcal{A}$.
  - For $1 \leq i \leq t$, $\mathcal{A}$ selects $SK_i \in \mathbb{Z}_q$ uniformly at random and sends $PK_i = g^{SK_i}$ to $\mathcal{B}$ (challenger in the perspective of $\mathcal{A}$).
  - For $1 \leq i \leq t$, $\mathcal{B}$ selects $f_i \in \mathbb{Z}_q$ uniformly at random, sets $v_i = h^{f_i}$ and $y_i = PK_i^{f_i}$.

– For $-(\ell-1) \leq j \leq -1$, $\mathcal{B}$ selects $f_j \in \mathbb{Z}_q$ uniformly at random, sets $v_j = h^{f_j}$. Also, $\mathcal{B}$ sets $v_0 = h^{\beta}$. Now that she has $\{v_j, u_j\}_{j=-(\ell-1)}^{0}$, she sets $y_0 = \prod_{j=-(\ell-1)}^{0} v_j^{u_j}$. Observe that $\mathcal{B}$ assumes $f_0 = \beta$ but does not know it explicitly.

– For $t+1 \leq i \leq n$, $\mathcal{B}$ generates $v_i = h^{p(i)}$ via lagrange interpolation in the exponent, where $p \in \mathbb{Z}_q[X]_{t+\ell-1}$ which is determined by the $t+\ell$ evaluations, namely, $p(0) = f_0 = \beta$, $p(i) = f_i$ for $-(\ell-1) \leq i \leq -1$ and $1 \leq i \leq t$. After obtaining v_i, $\mathcal{B}$ sets $y_i = v_i^{u_i}$ for $t+1 \leq i \leq n$.

– Now that $\mathcal{B}$ has $\{PK_i, y_i\}_{i=0}^{n}$, as a simulator she samples a random $t+\ell-1$ degree polynomial $z'(X)$ and sets $c'_i = \frac{PK_i^{z'(i)}}{y_i^d}$ for $1 \leq i \leq n$ and $c'_0 = \frac{\prod_{j=-(\ell-1)}^{0} PK_j^{z'(j)}}{y_0^d}$ when the challenge value d in the interactive version of the proof scheme is given.

– $\mathcal{B}$ sets h^{γ} as the last element in s_b.

– Finally, $\mathcal{B}$ sends $\{y_0, \ldots, y_n, c'_0, \ldots, c'_n, d, z'(X)\}$ along with s_b and public information to $\mathcal{A}$, simulating the role of a IND1-Secrecy challenger.

– $\mathcal{A}$ outputs the bit b' as its guess for b. If $b' = 0$ then $\mathcal{B}$ guesses $\gamma = \alpha.\beta$, otherwise γ is guessed to be a random element in $\mathbb{Z}_q$.

□

Remark 3.2.1. *A crucial observation is that the number of commitments to be downloaded by any verifier is same regardless of the number of secrets our 3PVSS scheme Λ_{RO}^{packed} is secret sharing. This is because only one single commitment represents all the secrets, moreover, it improves the verification cost as well in contrast to our first 3PVSS scheme Λ'_{RO} 3.3. We give explicit cost analysis of both of our schemes in the next section 3.3.*

Λ_{RO}^{packed}

Initialization: All parties $\{P_i\}_{i=1}^n$ and dealer D agree on the prime field $\mathbb{Z}_q$, a group $(\mathbb{G}, \times)$ of order q with a generator g, random oracle $\mathcal{H}$. Also, each party P_i registers their public key PK_i in the public ledger and all agree on a set of commitment keys or public keys, $\{h_j \text{ or } PK_j\}_{j=-(\ell-1)}^0$, whose secret keys are known to target entities.

Share:

- Dealer D samples a $t+\ell$-degree polynomial $f \in \mathbb{Z}_q[X]$ uniformly at random and sets $\{g^{f_j} : f_j = f(j)\}_{j=-(\ell-1)}^0$ as the set of all secrets.
- For each $1 \leq i \leq n$, D encrypts g^{f_i} $(f(i) = f_i)$ using PK_i to obtain $y_i = PK_i^{f_i}$. D also encrypts(/commits) to the secrets $\{g^{f_j}\}_{j=-(\ell-1)}^0$ using the encryption(/commitment) keys $\{h_j \text{ or } PK_j\}_{j=-(\ell-1)}^0$ and multiplies them together to obtain $y_0 = \prod_{j=-(\ell-1)}^0 (h_j \text{ or } PK_j)^{f_j}$.
- D uses a variant of the modified-PDL proof scheme 3.2.1 to generate π_{share} as follows:
  - Samples a $t+\ell-$degree polynomial $r \in \mathbb{Z}_q[X]$ uniformly at random and computes $c_0 = \prod_{j=-(\ell-1)}^0 (h_j \text{ or } PK_j)^{r(j)}$ and $c_i = PK_i^{r(i)}$ for $1 \leq i \leq n$.
  - Using $\mathcal{H}$, $d = \mathcal{H}(y_0, \ldots, y_n, c_0, \ldots, c_n)$ is computed.
  - Sets $z(X) = r(X) + df(X)$, hence $\pi_{share} = (d, z(X))$ is obtained.
- D broadcasts the encryptions of the shares along with the commitment of the secrets with π_{share} which proves the validity of the encrypted shares and committed secret, i.e., broadcasts $\{y_i\}_{i=0}^n$ and $(d, z(X))$.

Verification: Given public keys $\{PK_i\}_{i=1}^n$ and commitment(/public) keys $\{h_j \text{ or } PK_j\}_{j=-(\ell-1)}^0$, any entity can check π_{share} to verify the correctness of the encrypted shares $\{y_i\}_{i=1}^n$ and y_0 being the commitment of the secrets. They will output **true** or **false** based on the verification of the proof. The procedure is outlined as follows:

- The entity checks if $z(X)$ is a $t+\ell-$degree polynomial or not.
- Checks if $d = \mathcal{H}(y_0, y_1, \ldots, y_n, \frac{\prod_{j=-(\ell-1)}^0 (h_j \text{ or } PK_j)^{z(j)}}{y_0^d}, \frac{PK_1^{z(1)}}{y_1^d}, \ldots, \frac{PK_n^{z(n)}}{y_n^d})$.
- Outputs **true** if both of the above checks are satisfied, otherwise **false**.

Reconstruction: Similar to [4], there are two approaches to reconstruct the secrets $\{f_j\}_{j=-(\ell-1)}^0$ based on the cooperation of the dealer D which are as follows:

- **Optimistic Reconstruction:** D publishes $\{f_j\}_{j=-(\ell-1)}^0$, then any verifier (not necessarily shareholders) when given $g, \{h_j \text{ or } PK_j\}_{j=-(\ell-1)}^0, y_0$ can check if $y_0 = \prod_{j=-(\ell-1)}^0 (h_j \text{ or } PK_j)^{f_j}$ and then returns $\{g^{f_j}\}_{j=-(\ell-1)}^0$ when the check passes, if not they return **false**.
- **Pessimistic Reconstruction:** If D refuses to reveal $\{f_j\}_{j=-(\ell-1)}^0$, then any set $\mathcal{Q}$ consisting at least $t+\ell$ shareholders will do the following:
  - Each party $P_i \in \mathcal{Q}$ decrypts their share y_i using their private key SK_i corresponding to PK_i to obtain g^{f_i} and then they publish g^{f_i} along with a DLEQ proof 2.4.1, π_{DLEQ} which proves that g^{f_i} is the correct decryption of y_i.
  - If $\mathcal{Q}$ consists at least $t+\ell$ honest parties, they can use the lagrange interpolation to compute the secrets $\{g^{f_j}\}_{j=-(\ell-1)}^0$ as follows:

$$g^{f_j} = \prod_{i \in \mathcal{Q}} (g^{f_i})^{\prod_{k \in \mathcal{Q}, k \neq i} \frac{-k-j}{i-j}} = g^{\sum_{i \in \mathcal{Q}} f_i \prod_{k \in \mathcal{Q}, k \neq i} \frac{-k-j}{i-j}}.$$

Figure 3.5: Λ_{RO}^{packed}, an alternative packed version of Λ_{RO} [4]

3.3 Cost analysis of 3PVSS schemes

In this section, we analyze the computational and communication cost of both our 3PVSS schemes Λ'_{RO} and Λ^{packed}_{RO}. One can explicitly find the table 3.1 which summarizes these costs.

TABLE 3.1: Cost comparisons between Λ'_{RO} and Λ^{packed}_{RO}. BC: Dealer's Broadcase size, Dow: Download size by the verifier, Opt. Recon: Optimistic Reconstruction, Pes. Recon: Pessimistic Reconstruction. $\mathbb{E}_x$: Group exponentiation, $\mathbb{P}_e$: Polynomial evaluation, $M_{\mathbb{G}}$: multiplication in the cyclic group $\mathbb{G}$ of prime order q, $\mathbb{Z}_q$: modular prime group under addition.

Scheme	**Share**	**Broadcaset, Dow.**	**Verification**	**Opt. Recon**	**Pes. Recon**
Λ'_{RO}	$2(n+\ell)[\mathbb{E}_x + \mathbb{P}_e]$	**BC:** $(n+\ell)\lvert\mathbb{G}\rvert+(t+\ell+1)\lvert\mathbb{Z}_p\rvert$ **Dow:** $2(n+\ell)\lvert\mathbb{G}\rvert+(t+\ell+1)\lvert\mathbb{Z}_p\rvert$	$(n+\ell)[2\mathbb{E}_x+\mathbb{P}_e+M_{\mathbb{G}}]$	$\ell\mathbb{E}_x$	$5(t+\ell)\mathbb{E}_x+t\ell M_{\mathbb{G}}$
Λ^{packed}_{RO}	$2(n+\ell)[\mathbb{E}_x + \mathbb{P}_e]+2\ell M_{\mathbb{G}}$	**BC:** $(n+1)\lvert\mathbb{G}\rvert+(t+\ell+1)\lvert\mathbb{Z}_p\rvert$ **Dow:** $(2n+\ell+1)\lvert\mathbb{G}\rvert+(t+\ell+1)\lvert\mathbb{Z}_p\rvert$	$(2n+\ell+1)\mathbb{E}_x+(n+\ell)\mathbb{P}_e+(n+\ell+1)M_{\mathbb{G}}$	$\ell(\mathbb{E}_x+M_{\mathbb{G}})$	$5(t+\ell)\mathbb{E}_x+t\ell M_{\mathbb{G}}$

3.3.1 On the communication cost

Apart from downloading the public values from the initialization phase, the communication cost of Λ^{packed}_{RO} is independent of the number of secrets in contrast to Λ'_{RO}, where the communication cost grows linearly with the number of secrets. This is the main advantage of using Λ^{packed}_{RO} over Λ'_{RO}. We plotted the speculated communication costs of both the schemes in figure 3.6.

In the graph, we fixed the number of parties, $n = 10000$, for which we varied ℓ with $1 \leq t \leq \frac{n-\ell}{2}$.

3.3.2 On the computational cost

As Λ'_{RO} is a direct extension of Λ_{RO} [4], the computational cost grows linearly with the number of secrets. And as Λ^{packed}_{RO} is a slightly different approach, we save number of exponentiations in verification by a constant factor of ℓ but it costs additional group multiplications in share, verification and optimistic reconstruction phases. As group exponentiations with random exponents are usually more expensive than group multiplications (to put that in the perspective, computing g^x for a random $x \in \mathbb{Z}_q$ can take $\mathcal{O}(\log q)$ group multiplications). Hence, the realistic increase in cost of Λ^{packed}_{RO} due to additional group multiplications can be negligible.

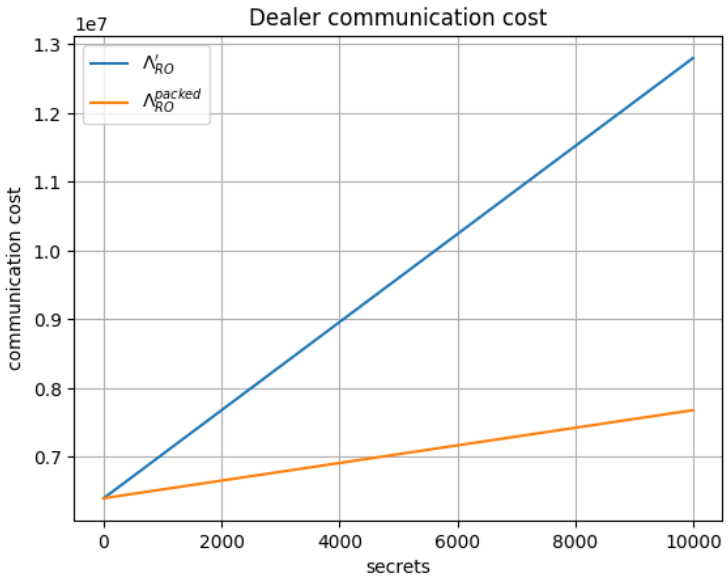

Figure 3.6: This plot compares Λ'_{RO} and Λ^{packed}_{RO} in terms of communication costs. The x-axis represents the possible number of secrets ℓ when the total number of parties is fixed, while the y-axis shows the communication cost for varying ℓ and threshold t. The blue line corresponds to Λ'_{RO}, and the orange line corresponds to Λ^{packed}_{RO}.

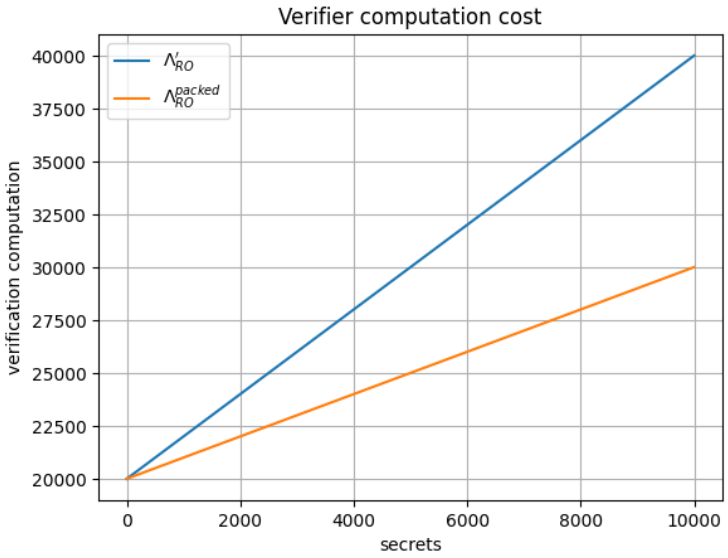

Figure 3.7: This plot compares Λ'_{RO} and Λ^{packed}_{RO} in terms of computation cost a verifier has to bear in the verification phase. The x-axis represents the possible number of secrets ℓ when the total number of parties is fixed, while the y-axis shows the verification cost for varying ℓ and threshold t. The blue line corresponds to Λ'_{RO}, and the orange line corresponds to Λ^{packed}_{RO}.

Similar to the graph in previous subsection, we plotted the speculated computational costs of both the schemes in figure 3.7 where we fixed the number of parties, $n = 10000$, for which we varied ℓ with $1 \leq t \leq \frac{n-\ell}{2}$. For this plot, we considered the group to be an elliptic curve over a 256-bit prime field which is cyclic of order a 256-bit prime q, i.e., each element (x, y) on the elliptic curve is typically of 512-bit size.

3.4 Conclusion

In this chapter, we generalized PPVSS to Packed PPVSS and presented two practical schemes based on Packed Shamir secret sharing. The reason to introduce the 3PVSS Λ'_{RO} is because of its potential applications in some e-voting protocols. In this thesis, we wanted to mainly focused on randomness beacon protocols based on PVSS, due to this reason we introduced our second 3PVSS Λ^{packed}_{RO} which is based on the NIZK AoK $\pi^{AoK}_{mod-PDL}$ for the modified-PDL problem.

In the next chapter, we will revisit the randomness beacon ALBATROSS [8] based on PVSS, and replace the Packed Shamir secret sharing based PVSS with our 3PVSS Λ^{packed}_{RO}.

Chapter 4

Revisiting a Randomness Beacon Protocol

Randomness Beacon [16] is required in applications like e-voting [1] and anonymous messaging ([23],[21]) to provide fresh unbiased random values to all the parties. In 2020, Cascudo and David published ALBATROSS [8], the state-of-the art randomness beacon protocol based on a PVSS as a building block where each party in the randomness beacon protocol acts as a dealer once, so that all parties can influence the output randomness. Interestingly, we observed that each party is expected to reveal their secrets (they secret shared as a dealer) as part of the randomness beacon protocol, but to prove that the secrets are valid and not just some random evaluations of the secret polynomial, they have to reveal the whole secret polynomial itself. As a consequence, if some entity wants to verify the secrets' validity then they have to simulate the whole sharing phase of the underlying PVSS protocol, which is very expensive because all the rest of the parties are expected to do the simulation of that party as a dealer. For reference, if there are n parties, then $n-1$ parties should simulate the sharing phase of the protocol, which in total is $\mathcal{O}(n^2)$ simulations.

In this chapter, we present our randomness beacon protocol in figures 4.1 and 4.2 which in many cases is efficient than ALBATROSS. To put simply, we replaced the building block being PVSS with our 3PVSS Λ_{RO}^{packed} 3.5. In the subsequent sections, we will discuss a bottleneck in ALBATROSS protocol, security analysis of our protocol briefly, followed by the computational and communication costs of our protocol and compare it with the ALBATROSS. We will show that our protocol performs more efficiently when compared to ALBATROSS in many cases and also address the cases where we are not computationally efficient. More interestingly, we will show that in terms of communication, we outperform ALBATROSS.

4.1 On the bottleneck of ALBATROSS

The main bottleneck of ALBATROSS is in its *Reveal* phase (description in figure 8 in section 4.3 of [8]). More precisely, the goal of the *Reveal* phase is to reveal the secrets of whom the dealer has done the secret sharing. But in order to verify that

Randomness Beacon using 3PVSS, Λ_{RO}^{packed}

Our protocol with 3PVSS is run between a set $\mathcal{P}$ of n parties $P_1, \ldots, P_n$ who have access to a public ledger where they can post information for later verification. It is assumed that the Setup phase of Λ_{RO}^{packed} is already done and the public keys pk_i of each party P_i along with $\{\mathbb{P}_i\}_{i=1}^{\ell}$ being Commitment keys (or public keys of target people) to encrypt the ℓ secrets are already registered in the ledger. In addition, the parties have agreed on a Vandermonde $(n-2t)\times(n-t)$-matrix $M = M(\omega, n-2t, n-t)$ with $\omega \in \mathbb{Z}_q^*$.

1. **Commit:** For $1 \leq j \leq n$:
   - Shareholder P_j executes the Distribution phase of the PP-PVSS as Dealer for $\ell = n-2t$ secrets, publishing commitments (/encryptions) of secrets, $y^j_{-(\ell-1)}, \ldots, y^j_{-1}, y^j_0$, and encryptions of shares $\{y^j_i\}_{i=1}^n$ along with π^j_{proof}, which is a NIZK AoK for proving the correctness of committed(/encrypted) secrets and encrypted secret shares on the public ledger, also learning the secrets $h^{s^j_0}, ..., h^{s^j_{-(\ell-1)}}$ and their corresponding exponents $s^j_0, \ldots, s^j_{-(\ell-1)}$.
2. **Reveal:**
   - Each shareholder checks the validity of the proof π^j_{proof}, i.e., the **verification phase of 3PVSS protocol** λ_{RO}^{packed}.
   - After a set $\mathcal{C}$ containing at least $n-t$ shareholders publish their shares in the public ledger, $P_j \in \mathcal{C}$ reveals ℓ secrets.
   - Every shareholder verifies the validity of secrets by reproducing the commitments using the commitment keys (/public keys of target people).
   - At this point, if every party in $\mathcal{C}$ has opened their secrets correctly, go to step 4' in Figure 4.2. Otherwise, proceed to step 3 in Figure 4.2.

Figure 4.1: Commit and Reveal phase of the Randomness Beacon using 3PVSS

those values are actual secrets, in ALBATROSS, a verifier has to simulate the whole sharing phase of the dealer with the values that were revealed. This is because, the underlying proof system of a general PVSS says that *if the PVSS verification algorithm outputs true, then the encryptions of the secret shares correspond to some unique secret with high probability.*

Randomness Beacon using 3PVSS, Λ_{RO}^{packed} (cont.)

3. **Recovery:** Let $\mathcal{C}_a$ be the set containing at most t malicious shareholders(as Dealers) who did not open the exponents corresponding to their ℓ secrets, $\{h^{s_i^k}\}_{i=0}^{-(\ell-1)}$ for each $P_k \in \mathcal{C}_a$, in *Reveal* phase.

    - Every shareholder P_j should decrypt the secret share of each malicious shareholder(Dealer) in $\mathcal{C}_a$, and give a DLEQ proof 2.4.1 which asserts that the decryption is performed correctly,i.e., each shareholder should perform the *pessimistic* reconstruction phase of the 3PVSS Λ_{RO}^{packed} for every shareholder(Dealer) who has not revealed the exponents corresponding to their secrets.

4 **Output:** Let T be the $(n-t) \times \ell$ matrix with rows indexed by the shareholders in $\mathcal{C}$ and where the row corresponding to $P_a \in \mathcal{C}$ is $(h^{s_0^a}, ..., h^{s_{-(\ell-1)}^a})$.

- Each computes the $\ell \times \ell$-matrix $R = M \circ T$ by applying FFTE to each column $T^{(j)}$ of T, resulting in column $R^{(j)}$ of R (since $R^{(j)} = M \circ T^{(j)}$ and M is Vandermonde) for $j \in [0, \ell - 1]$.
- Shareholders output the ℓ^2 elements of R as final randomness.

4' **Alternative Output:** if every party in $\mathcal{C}$ has opened her secrets correctly in step *Reveal*, then:

- Shareholders compute $R = M \circ T$ in the following way: Let S be the $(n-t) \times \ell$ matrix with rows indexed by the shareholders in $\mathcal{C}$ and where the row corresponding to $P_a \in \mathcal{C}$ is $(s_0^a, ..., s_{-(\ell-1)}^a)$. Then each party computes $U = M \circ S \in \mathbb{Z}_q^{\ell \times \ell}$ (using the standard FFT in $\mathbb{Z}_q$ to compute each column) and $R = h^U$.
- Shareholders output the ℓ^2 elements of R as final randomness.

FIGURE 4.2: Recovery and Output phase of the Randomness Beacon using 3PVSS

Whereas, in the case of (P)PPVSS, *if the verification algorithm outputs true then it means that the encryptions of the secret shares correspond to the unique secret(s) which was already committed by the dealer (which is the reason why commitment value of secret(s) is a part of the proof statement) with high probability.* So, the

intuition is that if one uses a PPVSS, that packs multiple secrets in a single polynomial, over the PVSS used in ALBATROSS (a Random Beacon protocol) then the efficiency of the new Randomness beacon protocol can be improved.

4.2 Security analysis

This section will briefly discuss why our protocol can remain secure. Note that, the major changes in our protocol in contrast to ALBATROSS are the *Commit* and *Reveal* phases. As long as the randomizers used to commit (/secret keys to decrypt) secrets are not leaked then *Commit* phase is as secure such that it does not reveal anything about the secrets. This is because of the mathematical security guarantees achieved through the sharing phase of our 3PVSS, Λ_{RO}^{packed}.

In ALBATROSS, the whole point of the *Reveal* phase is to open the secrets itself, but to verify that the secrets are valid for a given secret shares, one has to simulate the whole sharing phase of the respective dealer. As our 3PVSS, Λ_{RO}, already proves that the commitment is a valid commitment of dealers' secrets and their respective secret sharing is done correctly, it does not require to perform the whole simulation of the sharing phase of any dealer. Because the dealer has already committed to the secrets, he cannot cheat by revealing wrong secrets. Therefore, the *Reveal* phase of our protocol also is secure due to the security guarantees of Λ_{RO}^{packed}.

4.3 Computational Complexity

One would notice that the major changes are in the commit and reveal phase of the protocol. Hence, in this section we only will talk about the performance analysis in those phases. See table 4.1 for an overview. Also, we ignore the number of multiplications in the group $\mathbb{G}$, i.e., $M_{\mathbb{G}}$ in the table as it is negligible compared to the number of group exponentiations $\mathbb{E}_x$ with random exponents in practice.

- In ALBATROSS, a dealer(as a part of **commit**) should compute $n(\mathbb{E}_x + \mathbb{P}_e)$ commitments and to give a proof he should do an additional $n(\mathbb{E}_x + \mathbb{P}_e)$. Also, the dealer should do $\ell(\mathbb{E}_x + \mathbb{P}_e)$ for computing secrets and keeping it to himself. In total dealer needs to do $(2n + \ell)[\mathbb{E}_x + \mathbb{P}_e]$.
  - In **Reveal**, a verifier should compute $2n\mathbb{E}_x$ which internally requires additional $n\mathbb{P}_e$, i.e., in total it requires $(n-1)n(2\mathbb{E}_x + \mathbb{P}_e)$ computations for each verifier.
    - In **Robust case** where t dealers do not open their polynomials, a verifier should verify $n - t$ polynomials of honest dealers, i.e., for each honest dealer, a verifier has to do $n\mathbb{P}_e$ to evaluate secret share exponents and does $n\mathbb{E}_x$ to get secret shares and cross checks them in the public ledger. Also, finally the verifier computes $\ell\mathbb{P}_e$ to get secret exponents and get ℓ secrets by doing $\ell\mathbb{E}_x$. As there are $n - t$ honest dealers, the verifier has to compute $(n - t)(n + \ell)(\mathbb{E}_x + \mathbb{P}_e)$.

TABLE 4.1: Computational cost of dealer and shareholders, $\mathbb{E}_x$ =group exponentiation and $\mathbb{P}_e$ =polynomial evaluation in group $\mathbb{G}$ with order q, where q is a large prime

Protocol	**Output size**	**Commit** *(by Dealer)*	**Reveal** *(by shareholder)*	**Recovery** *(by shareholder)*
ALBATROSS, *Honest case*	ℓ^2	$(2n+\ell)[\mathbb{E}_x+\mathbb{P}_e]$	**Share Verification -** $(n-1)n[2\mathbb{E}_x+\mathbb{P}_e]$ **Secret Verification -** $(n-1)(n+\ell)[\mathbb{E}_x+\mathbb{P}_e]$	-
with PPPVSS, *Honest case*	ℓ^2	$2(n+\ell)[\mathbb{E}_x+\mathbb{P}_e]$	**Share Verification -** $(n-1)(n+\ell)[2\mathbb{E}_x+\mathbb{P}_e]$ **Secret Verification -** $(n-1)\ell\mathbb{E}_x$	-
ALBATROSS, *Robust case*	ℓ^2	$(2n+\ell)[\mathbb{E}_x+\mathbb{P}_e]$	**Share Verification -** $(n-1)n[2\mathbb{E}_x+\mathbb{P}_e]$ **Secret Verification -** $(n-t-1)(n+\ell)[\mathbb{E}_x+\mathbb{P}_e]$	$[3+4(n-t)]t\mathbb{E}_x$
with PPPVSS, *Robust case*	ℓ^2	$2(n+\ell)[\mathbb{E}_x+\mathbb{P}_e]$	**Share Verification -** $(n-1)(n+\ell)[2\mathbb{E}_x+\mathbb{P}_e]$ **Secret Verification -** $(n-t-1)\ell\mathbb{E}_x$	$[3+4(n-t)]t\mathbb{E}_x$

    - In **Honest case**, everyone would have been honest and so each verifier has to do $(n-1)(n+\ell)(\mathbb{E}_x+\mathbb{P}_e)$.
  - **Recovery** phase only exists if some party does not open the polynomial leading to PVSS reconstruction phase, in the worst case there should be reconstruction for the secrets of t malicious parties. Given a malicious shareholder who has not opened the secret polynomial, each shareholder/reconstructor has to decrypt their share, which requires $1\mathbb{E}_x$ and should give a DLEQ proof that they have decrypted correctly, which additionally requires $2\mathbb{E}_x$; Also the re-constructor should verify DLEQ proofs of correct share decryption from $n-t$ honest shareholders requiring them to do $4(n-t)\mathbb{E}_x$. In total, each re-constructor requires $[3+4(n-t)]t\mathbb{E}_x$.

- Using 3PVSS in randomness beacon protocol, a dealer(as a part of **commit**) requires to do $(n+\ell)[\mathbb{E}_x+\mathbb{P}_e]$ and $(\ell-1)M_{\mathbb{G}}$ to compute $\{y_i\}_{i=0}^{n}$. For generating the proof that y_i's are valid encryptions of the secret shares and also y_0 is a commitment of the ℓ secrets, the dealer should do $(n+\ell)[\mathbb{E}_x+\mathbb{P}_e]$ which internally requires additional $(\ell-1)M_{\mathbb{G}}$. In total, a dealer has to do $2\left[(n+\ell)[\mathbb{E}_x+\mathbb{P}_e]+(\ell-1)M_{\mathbb{G}}\right]$.

  - In **Reveal**, a verifier should do $(n+\ell)(2\mathbb{E}_x+\mathbb{P}_e)$ and $(\ell-1)M_{\mathbb{G}}$ for each

proof. In total, a verifier has to do $(n-1)(n+\ell)[2\mathbb{E}_x + \mathbb{P}_e] + (n-1)(\ell-1)M_{\mathbb{G}}$.

* In **Robust case** with t malicious parties not opening the secret polynomials, a verifier should do $\ell\mathbb{E}_x + (\ell-1)M_{\mathbb{G}}$ to verify each proof, so in total each verifier should do $(n-t-1)[l\mathbb{E}_x + (\ell-1)M_{\mathbb{G}}]$.
* In **Honest case** where everyone is honest, a verifier will do $(n-1)\ell(\mathbb{E}_x + M_{\mathbb{G}})$.

– The computational complexity of each re-constructor in **Recovery** phase is exactly same as in the case of ALBATROSS.

4.3.1 Computational Cost analysis

The dealer has to do a bit more work in the case of our protocol in contrast to ALBATROSS, more explicitly, they have to compute ℓ more group exponentiations and polynomial evaluations. But as a consequence, we decrease computational cost in the *Reveal* phase whenever $\ell < \frac{n(n-t-1)}{2(n-1)}$, roughly speaking, if the number of secrets are less than half of the honest parties then we always perform better in terms of computation when compared to the ALBATROSS.

More precisely, in the reveal phase, we observed that our protocol gains at least 30% more efficiency for performing polynomial evaluations, and at least 23% more efficiency for performing group exponentiations when compared to ALBATROSS in the robust case where the number of secrets ℓ is equal to one. See plots 4.3 and 4.4 for the performance analysis of polynomial evaluations and group exponentiations in the reveal phase of our protocol and ALBATROSS.

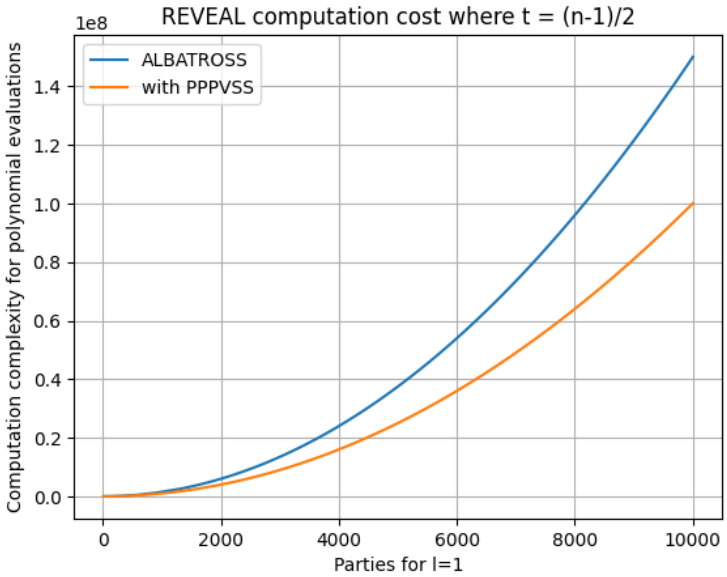

Figure 4.3: Computation costs due to polynomial evaluations bore by each verifier in reveal phase for the most robust case where $\ell = 1$

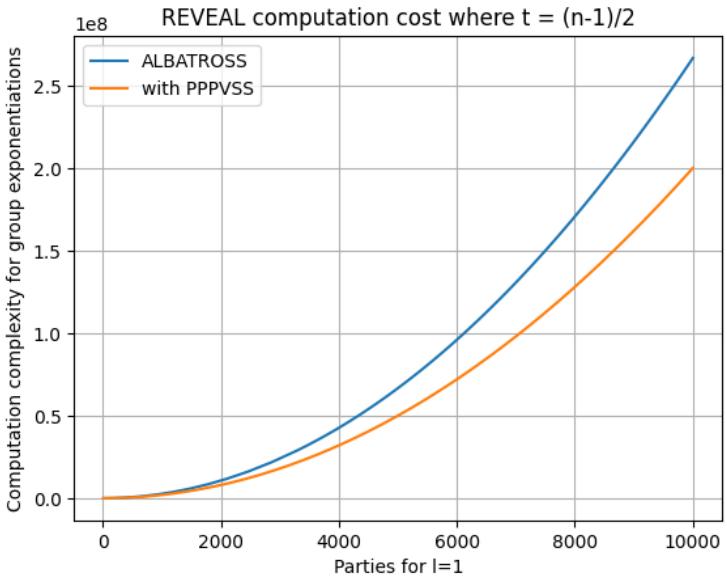

FIGURE 4.4: Computation costs due to group exponentiations bore by each verifier in reveal phase for the most robust case where $\ell = 1$

4.4 Communication Complexity

TABLE 4.2: Communication cost of dealer and (each) shareholder, $\mathbb{G}$ =group of order q and $\mathbb{Z}_q$ = modular group of order q, where q is a large prime

Protocol	**Commit** *(by Dealer)*	**Reveal** *(by Dealer)*	**Recovery** *(by shareholder)*
ALBATROSS	$n\mathbb{G} + (t+\ell)\mathbb{Z}_q$	$(t+\ell)\mathbb{Z}_q$	$1\mathbb{G} + 2\mathbb{Z}_q$
with PPPVSS	$(n+1)\mathbb{G} + (t+\ell)\mathbb{Z}_q$	$\ell\mathbb{Z}_q$	$1G + 2\mathbb{Z}_q$

See table 4.2 for an overview.

- In ALBATROSS, a dealer (as a part of **commit**) should send n group elements as commitments, $t + \ell$ elements in $\mathbb{Z}/q\mathbb{Z}$ that defines the polynomial used in the ZKP and 1 extra element in $\mathbb{Z}/q\mathbb{Z}$ from RO.
  - In **Reveal**, an honest dealer would broadcast $t + \ell$ coefficients in $\mathbb{Z}/q\mathbb{Z}$ concerning the secret polynomial.
  - If some party has not revealed their polynomial, then in **Recovery** phase a re-constructor using PVSS reconstruction protocol should broadcast 1 element in group which is being the decrypted secret, for the proof of correct decryption, they have to broadcast 3 more group elements along with a polynomial which requires $t + \ell$ coefficients in $\mathbb{Z}/q\mathbb{Z}$ and 1 group element from RO.

- Using 3PVSS in randomness beacon protocol, a dealer (as a part of **commit**) should send $n+1$ group elements as commitments, $t+\ell$ elements in $\mathbb{Z}/q\mathbb{Z}$ that defines the polynomial used in the ZKP and 1 extra element in $\mathbb{Z}/q\mathbb{Z}$ from RO.
  - In **Reveal**, an honest dealer would broadcast ℓ elements in $\mathbb{Z}_q$ concerning the exponents to construct the secret.
  - If some part has not revealed their secrets, then the communication cost of each re-constructor is exactly same as in the case of ALBATROSS.

4.4.1 Communication Cost analysis

The best to offer from our randomness beacon protocol is the communication cost. Though the dealer has to communicate only one extra group element compared to ALBATROSS in the commit phase, as a consequence for a fixed number of secrets the dealers' communication cost is constant as opposed to linear in number of corrupted parties in ALBATROSS.

In the plots 4.6 and 4.5, we show the communication cost of a dealer to bare in ALBATROSS and our protocol in both the commit and reveal phases.

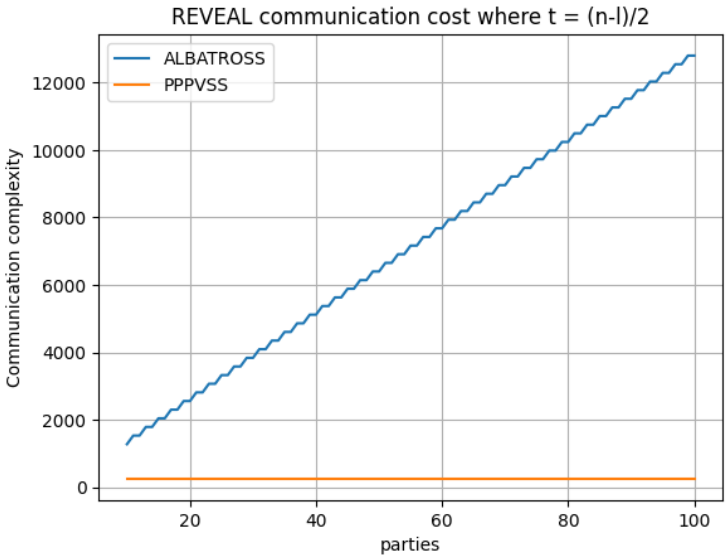

Figure 4.5: Communication cost of a dealer in reveal phase of ALBATROSS and our protocol

4.5 Conclusion

The noteworthy efficiency gains our protocol achieves over ALBATROSS is in the communication complexity, which does not depend on the number of secrets at all

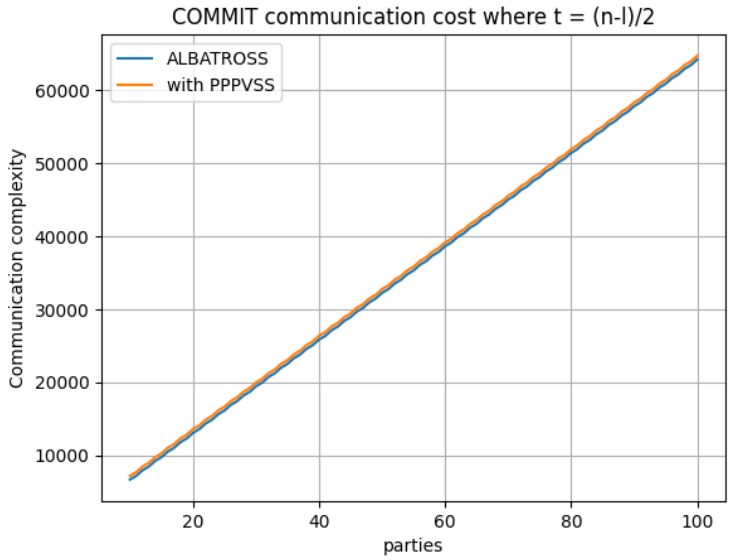

FIGURE 4.6: Communication cost of a dealer in commit phase phase of ALBATROSS and our protocol

in contrast to ALBATROSS which depends linearly. Also, our protocol is better efficient in terms of time complexity in most of the realistic cases where the number of secrets is roughly bounded by half the number of honest parties. In the next and final chapter, this thesis work is summarized at a higher level and also discusses the potential future scope of it.

Chapter 5

Conclusion and Future scope

This thesis report gives a detailed background on the mathematical security aspects of Shamir secret sharing, and necessary required literature is reviewed by recalling a very recent variant of PVSS, more precisely, PPVSS is recalled along with an example candidate Λ_{RO} proposed in [4]. And further, the PPVSS is generalized to 3PVSS (/PPPVSS) in this thesis where one has to share multiple secrets packed in a single polynomial, and subsequently, two candidate 3PVSS schemes are proposed in this report. Also, the security guarantees of both candidates are shown.

The goal of this thesis was to revisit the state-of-the-art randomness beacon protocol based on PVSS, ALBATROSS [8]. A significant bottleneck of ALBATROSS is in its *Reveal* phase, as name suggests it reveals the dealers' secrets and verifies the validity of the revealed secrets. With 3PVSS, it was shown in this report that ALBATROSS could be further improved in efficiency in many cases. The report also highlighted the cases where our protocol can underperform ALBATROSS. The security properties of our protocol was briefly discussed which strongly depends on the 3PVSS, Λ_{RO}, mathematical security guarantees.

When trying to prove the mathematical security guarantees of Λ_{RO}^{packed}, the underlying mathematical proof system was acknowledged. So, before delving into the security proof of Λ_{RO}^{packed}, its underlying proof system with its properties was written down formally, which were essential and desirable to explain its achievable security. Though Λ_{RO}^{packed} is more efficient than the first proposed 3PVSS, Λ'_{RO}, it is essential to note that Λ'_{RO} was tailored for specific applications where using Λ_{RO}^{packed} may not be suitable. Also, the discussed security guarantees of Λ'_{RO} in this report is trivially realized from the security guarantees of Λ_{RO}. For completeness, both proposed candidates for 3PVSS computation and communication costs were discussed.

To conclude, this thesis gave enough background and literature review required to progress till the end. An extension of newly proposed variant of PVSS is proposed and two candidate examples with their security proofs were given. An application based on PVSS was revisited, where its bottleneck is taken into account, and improved its performance in many cases. Especially, the communication cost of the new protocol was always less in contrast to ALBATROSS.

Future scope

The new proof system $\pi^{AoK}_{mod-PDL}$ on which Λ^{packed}_{RO} is based on, can be further improved. So in future, one can think if a new proof system that can be built to improve not only Λ^{packed}_{RO} but also the state-of-the-art PVSS Π_S [3], which can lead to revisiting all the applications built so far with the aforementioned schemes.

Bibliography

[1] B. Adida. Helios: web-based open-audit voting. In *Proceedings of the 17th Conference on Security Symposium*, SS'08, page 335–348, USA, 2008. USENIX Association.

[2] S. Atapoor, K. Baghery, D. Cozzo, and R. Pedersen. VSS from distributed ZK proofs and applications. Cryptology ePrint Archive, Paper 2023/992, 2023.

[3] K. Baghery. π: A unified framework for computational verifiable secret sharing. Cryptology ePrint Archive, Paper 2023/1669, 2023.

[4] K. Baghery, N. Knapen, G. Nicolas, and M. Rahimi. Pre-constructed publicly verifiable secret sharing and applications. Cryptology ePrint Archive, Paper 2025/576, 2025.

[5] G. R. Blakley and C. Meadows. Security of ramp schemes. In *Advances in Cryptology, Proceedings of CRYPTO '84, Santa Barbara, California, USA, August 19-22, 1984, Proceedings*, volume 196 of *Lecture Notes in Computer Science*, pages 242–268. Springer, 1984.

[6] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell. Bulletproofs: Short proofs for confidential transactions and more. Cryptology ePrint Archive, Paper 2017/1066, 2017.

[7] I. Cascudo and B. David. SCRAPE: Scalable randomness attested by public entities. Cryptology ePrint Archive, Paper 2017/216, 2017.

[8] I. Cascudo and B. David. ALBATROSS: publicly AttestabLe BATched randomness based on secret sharing. Cryptology ePrint Archive, Paper 2020/644, 2020.

[9] D. Chaum and T. P. Pedersen. Wallet databases with observers. In E. F. Brickell, editor, *Advances in Cryptology — CRYPTO' 92*, pages 89–105, Berlin, Heidelberg, 1993. Springer Berlin Heidelberg.

[10] B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch. Verifiable secret sharing and achieving simultaneity in the presence of faults (extended abstract). In *SFCS '85*, pages 383–395. IEEE, 1985. DBLP's bibliographic metadata

records provided through http://dblp.org/search/publ/api are distributed under a Creative Commons CC0 1.0 Universal Public Domain Dedication. Although the bibliographic metadata records are provided consistent with CC0 1.0 Dedication, the content described by the metadata records is not. Content may be subject to copyright, rights of privacy, rights of publicity and other restrictions.; 26th Annual Symposium on Foundations of Computer Science ; Conference date: 21-10-1985 Through 23-10-1985.

[11] P. Feldman. A practical scheme for non-interactive verifiable secret sharing. In *28th Annual Symposium on Foundations of Computer Science, Los Angeles, California, USA, 27-29 October 1987*, pages 427–437. IEEE Computer Society, 1987.

[12] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In A. M. Odlyzko, editor, *Advances in Cryptology — CRYPTO' 86*, pages 186–194, Berlin, Heidelberg, 1987. Springer Berlin Heidelberg.

[13] M. Franklin and M. Yung. Communication complexity of secure computation (extended abstract). In *Proceedings of the Twenty-Fourth Annual ACM Symposium on Theory of Computing*, STOC '92, page 699–710, New York, NY, USA, 1992. Association for Computing Machinery.

[14] J. Gallian. *Contemporary Abstract Algebra*. CRC Press, 2024.

[15] T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings*, volume 576 of *Lecture Notes in Computer Science*, pages 129–140. Springer, 1991.

[16] M. O. Rabin. Transaction protection by beacons. *Journal of Computer and System Sciences*, 27(2):256–267, 1983.

[17] I. S. Reed and G. Solomon. Polynomial codes over certain finite fields. *Journal of the Society for Industrial and Applied Mathematics*, 8(2):300–304, 1960.

[18] C.-P. Schnorr. Efficient identification and signatures for smart cards. In *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, volume 435 of *Lecture Notes in Computer Science*, pages 239–252. Springer, 1989.

[19] L. Schoenmakers. A simple publicly verifiable secret sharing scheme and its application to electronic voting. In M. Wiener, editor, *Advances in Cryptology - CRYPTO'99 (Proceedings 19th Annual International Cryptology Conference, Santa Barbara CA, USA, August 15-19, 1999)*, Lecture Notes in Computer Science, pages 148–164, Germany, 1999. Springer.

[20] A. Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, Nov. 1979.

[21] J. van den Hooff, D. Lazar, M. Zaharia, and N. Zeldovich. Vuvuzela: scalable private messaging resistant to traffic analysis. In *Proceedings of the 25th Symposium on Operating Systems Principles*, SOSP '15, page 137–152, New York, NY, USA, 2015. Association for Computing Machinery.

[22] L. R. Welch and E. Berlekamp. Error correction for algebraic block codes, December 1986. U.S. Patent 4,633,470.

[23] D. I. Wolinsky, H. Corrigan-Gibbs, B. Ford, and A. Johnson. Dissent in numbers: Making strong anonymity scale. In *10th USENIX Symposium on Operating Systems Design and Implementation (OSDI 12)*, pages 179–182, Hollywood, CA, Oct. 2012. USENIX Association.