UNIVERSITÄT
DES
SAARLANDES

Gaussian Linear Secret Sharing

submitted by:
Niklas Britz

March 8, 2024

Bachelor Thesis

Reviewers: Dr. Nico Döttling, Dr. Julian Loss
Department of Computer Science
Faculty of Mathematics and Computer Science
Saarland University

Statement

I hereby confirm that I have written this thesis on my own and that I have not used any other media or materials than the ones referred to in this thesis.

Erklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

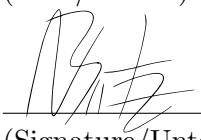
Declaration of Consent

I agree to make both versions of my thesis (with a passing grade) accessible to the public by having them added to the library of the Computer Science Department.

Einverständniserklärung

Ich bin damit einverstanden, dass meine (bestandene) Arbeit in beiden Versionen in der Bibliothek der Informatik aufgenommen und damit veröffentlicht werden.

Saarbrücken, March 8, 2024
(Date/Datum)



(Signature/Unterschrift)

Abstract

Secret sharing is a cryptographic technique in which a dealer splits a secret into multiple *shares* and distributes those shares among participating parties. In *threshold secret sharing*, shareholders with a specified threshold of those shares can collaboratively reconstruct the secret. Parties with access to a smaller, unqualified set of shares learn little or nothing about the original secret if they collaborate.

A desirable property of secret sharing schemes is *linearity*, i.e. the secret is a linear combination of the shares. Linear secret sharing schemes can be employed in distributed protocols that require some secret. In many distributed protocols, individual parties can compute a partial result that can be combined into the final result using the linearity property.

As of now, many established (linear) secret sharing protocols operate over finite algebraic structures like groups of prime order. This approach is sufficient for many practical problems since the size of the structure can be chosen as required. Nevertheless, there are protocols, like distributed exponentiation for RSA using secret sharing, where schemes over some finite structure are infeasible. The parties would have to know the group size for reconstruction. However, knowing the group size would break the security of RSA. Secret sharing over infinite algebraic structures, in which no finite group size exists, is desirable, not only in a scenario like this. It is particularly interesting if the group size is unknown before a protocol's execution or has to remain secret.

In this work, we present two threshold secret sharing schemes. The first is a continuous linear secret sharing scheme over $\mathbb{R}$. We prove the correctness and security of this scheme.

The second is an approximate linear scheme over $\mathbb{Z}$, for which we prove correctness. Here, approximate linear means that the scheme is linear up to some minor error.

We present a novel approach to linear secret sharing over $\mathbb{Z}$ and $\mathbb{R}$ by hiding secrets under Gaussian distributed shares.

Contents

1 Introduction

Consider the following scenario: An entity A possesses some secret m stored locally. In the event that a third party M compromises A, m might be revealed to M. A can be considered the *single point of failure*.

This problem gives rise to the notion of *secret sharing*. In secret sharing, the secret m is distributed among a set of parties participating in the protocol. A robust secret sharing scheme fulfills two properties: A qualified subset of the parties can collaborate to reconstruct the secret. In contrast, an unqualified subset of parties learns nothing about the secret m. The determination of qualified and unqualified parties is specific to the instance of the protocol being employed.

Coming back to our example: If M compromises A, who has secret-shared m, M will learn nothing about the secret.

1.1 Secret Sharing: A Formal Definition

Concretely, a *secret sharing scheme* $\mathcal{S}$ is a set of algorithms $(\mathcal{D}, \mathcal{R})$. The *dealer* uses the distribution algorithm $\mathcal{D}$ to split the secret m into partial information and distribute those parts to the n participating parties of the protocol. Those pieces of information $s_1, \ldots, s_n$ are called *shares* of the secret. We call a secret sharing scheme a (t, n)-*threshold scheme*, $1 \leq t \leq n$, if:

- $\mathcal{R}$ can reconstruct m efficiently if it gets t or more shares as input.
- $t - 1$ shares reveal nothing about the secret.

1.2 What is Secret Sharing used for?

There are several types of secret sharing schemes [CSNN24] [Bei11]: Besides the previously defined secret sharing schemes with threshold access structure, there are schemes with general access structure. For the latter, the dealer can choose what sets of parties can reconstruct the secret - not just some t out of n parties. The advantages and disadvantages of both will be further discussed in section 5. In addition, there is the category of verifiable secret sharing schemes, which are resistant to cheating and dishonest secret sharing parties. Secret sharing schemes can also be very domain-specific. There is, for example, DNA-based secret sharing in which DNA samples are shared with participating parties [Adh06].

One straightforward application of secret sharing is removing the risk that an attacker obtains sensitive information at a single point of failure. Sensitive information include secret keys, passwords, medical information, or bank statements.

Besides this application, classical usages of secret sharing in cryptography include Byzantine agreement, secure multiparty computations, threshold cryptography, access control, attribute-based encryption, and generalized oblivious transfer [Bei11]. Moreover, secret sharing finds application in blockchain, IoT, cloud storage, e-voting, etc. [CSNN24].

1.3 Different Kinds of Security Models

Coming back from the use cases of secret sharing, we want to discuss secret sharing itself in greater detail.

Naturally, more than just providing a secret sharing algorithm is required when presenting a new scheme. We have to prove that it is secure. However, security can be an ambiguous term, and

different kinds of security models can be used to analyze secret sharing schemes. In general, an adversary has many options for influencing the secret-sharing process, like manipulating the shares that the dealer distributes or letting corrupted parties deviate from the protocol [PM14].

In the following thesis and in most of the literature the security of a secret sharing scheme is analyzed towards an *honest-but-curious* attacker setting. [PM14] defines an honest-but-curious adversary as follows:

> "The [...] adversary is a legitimate participant in a communication protocol who will not deviate from the defined protocol but will attempt to learn all possible information from legitimately received messages."

Besides defining security by what an attacker can do, there are different definitions of what can be considered secure. While *perfect secrecy* implies that an attacker learns nothing about the secret apart from what was known before protocol execution, schemes with *statistical security* reveal a statistically negligible amount of information about the secret. Later, we will see that linear integer secret sharing can only achieve statistical security and not perfect secrecy. However, statistical security is already a strong notion since many well-known cryptographic protocols only achieve *computational security*, which states that a protocol is secure against a computationally bounded attacker. Statistically secure protocols are even secure against computationally unbounded attackers.

1.4 Secret Sharing on Finite Structures

Now, we will look at basic secret sharing schemes and motivate integer secret sharing afterward.

The most prominent secret sharing scheme was introduced in Adi Shamir's seminal paper "How to Share a Secret" [Sha79]. In this paper Shamir proposes the following method, called *Shamir secret sharing*:

- For a (t, n) secret sharing scheme, a secret m, and a prime $p > m, n$, sample random coefficients $a_i \in \mathbb{Z}_p$, set $a_0 = m$ and create the polynomial $q(x) = a_{t-1}x^{t-1} + ... + a_1x^1 + a_0$.
- Compute shares $s_1, \ldots, s_n$ as $s_i = q(i) \mod p$.
- With t shares given, one can reconstruct the polynomial using Lagrange interpolation and evaluate $q(0) = m$.
- Furthermore, if an attacker obtains $t - 1$ shares, m is still uniformly distributed on $[0, p - 1]$.

We can see that the scheme uses modular arithmetic on the (finite) group $\mathbb{Z}_p$ to perform computations.

An even simpler secret sharing scheme is *additive secret sharing* [add]. Additive secret sharing is a (n, n)-secret sharing scheme. That means that all n shares must be combined to reconstruct the secret. It works the following way:

- Let m be the secret. We choose some number p as our modulus.
- We uniformly draw numbers $s_1, \ldots, s_{n-1}$ from $\mathbb{Z}_p$ and set $s_n = m - \sum_{i=1}^{n-1} s_i \mod p$.
- Note that n parties can reconstruct the secret by adding all shares together: $m = \sum_{i=1}^{n} s_i \mod p$.

- Furthermore if an attacker obtains $n-1$ shares, any number $\in \mathbb{Z}_p$ is equally likely to be the secret. There is exactly one number namely the missing share s_k, s.t. $(\sum_{i\in\{1,\dots,n\}\setminus k} s_i) + s_k \mod p = m$. Because s_k is uniformly distributed in $\mathbb{Z}_p$ and $\mathbb{Z}_p$ is a cyclic group the secret m is also uniformly distributed.

Again, we use modular arithmetic on finite groups. In both schemes presented above, the posterior distribution of the secret is the same as before protocol execution, i.e., an attacker learns nothing about the secret. We have perfect secrecy.

For many applications, those "finite" secret sharing schemes are sufficient since the group size can be chosen arbitrarily large or the problem is of a finite/modular nature.

1.5 Integer Secret Sharing

Although secret sharing schemes that operate on finite groups may guarantee perfect security, they have other drawbacks. In this thesis, we discuss one scheme that realizes integer secret sharing, i.e., we perform integer, not modular, arithmetic.

Besides the purely academic interest in finding a secret sharing scheme that is not restricted to finite groups, there are several reasons why integer secret sharing is desirable.

One reason is that general distributed exponentiation can be realized with integer secret sharing schemes, while there might be construction issues or security risks with finite number secret sharing [Tho09].

Distributed exponentiation means computing a^m, where m is a secret exponent that is secret-shared among different parties, and a is some number in a (finite) group. In a (t, n) setting t parties can use their shares to compute a^m together. In Shamir's scheme, because it is linear, we can reconstruct $m = \sum_{i=1}^{t} \alpha_i s_i \mod q$, where α_i are the Lagrange coefficients that can be publicly computed.

Distributed exponentiation is a crucial operation for public-key cryptography and is used in the RSA algorithm [RSA78] or the Diffie-Hellman key exchange [DH76]. In the case of the RSA algorithm, we want to compute $a^m \mod \phi(N)$, where $\phi(N)$ is a number that is not prime and not public (otherwise, breaking RSA is easy). However, now we cannot use a finite secret sharing scheme like Shamir's scheme that uses $q = \phi(N)$ as a modulus since q has to be known to all parties and prime [Tho09]. There exists workarounds that do not require integer secret sharing but which [Tho09] weakens in his argumentation. We will discuss the distributed exponentiation problem in detail in section 6.

By this example, we see that integer secret sharing is desirable for (distributed) protocols, in which the group size is unknown before protocol execution or has to remain secret.

Note that we cannot just simply adjust finite schemes to be integer secret sharing schemes. For instance, while Shamir secret sharing works well on finite sets, the scheme is broken on infinite sets because there is no guarantee that inverses are defined on $\mathbb{Z}$. However, Lagrange interpolation requires the existence of inverses in the group regarding multiplication. Thus, we might not be able to reconstruct the secret with interpolation. Furthermore, using integers might reveal information about the secret, such as it's parity [int].

1.5.1 CRT-based schemes

To our knowledge, the idea of sharing integers was first presented by Maurice Mignotte in 1982 [Mig82]. In his approach, he uses the Chinese Remainder Theorem (CRT) to show that a (t, n)

secret sharing can be realized on the integers. It works the following:
For some integer secret $\alpha < S < \beta$ bounded by integers α, β, find co-prime integers $d_1 < \cdots < d_n$ s.t.

$$\prod_{i=1}^{t} d_i > \beta \text{ and } \prod_{i=n-t+2}^{n} d_i < \alpha$$

Then, create n shares $x_i = S \mod d_i$. Since $d_1, \ldots, d_t$ are co-prime the CRT states that:

$$\begin{aligned} x &\equiv x_1 \mod d_1 \\ x &\equiv x_2 \mod d_2 \\ &\vdots \\ x &\equiv x_t \mod d_t \end{aligned}$$

has exactly one solution: $x = s$. This solution can be found using the extended euclidean algorithm [Mig82], i.e., coefficients $z_1, \ldots, z_t$ can be found s.t.

$$S = z_1 x_1 + z_2 x_2 + \cdots + z_t x_t \mod d_1 d_2 \ldots d_t$$

For $t-1$ parties, the secret is uniformly distributed between $\frac{\beta-\alpha}{d_1 \ldots d_{t-1}}$ values [Mig82].

As we can see, there are substantial restrictions on the scheme's construction, especially about how to come up with appropriate d_j's. Most importantly, CRT-based schemes are non-linear in $\mathbb{Z}$ since reconstruction happens modulus $d_1 \ldots d_t$. This means that for every subset of t parties, the group in which the scheme is linear in might differ because the factors $d_1 \ldots d_t$ could be different. Similar problems arise in the CRT-based scheme presented by Asmuth and Bloom in the following year [AB83] [NMH+18].

Now, note that CRT-based schemes achieve perfect secrecy but are not linear (in $\mathbb{Z}$). In fact, [Tho09] states and [CK93] proofs that for linear integer secret sharing schemes perfect secrecy is unachievable.

1.5.2 Previous Work

One of the first researchers who came up with a linear integer secret sharing scheme that can be used for tasks like distributed exponentiation were Ivan Damgård and Rune Thorbek [DT06] [Tho09]. They use general access structure secret sharing as described above. Therefore, they first define that Γ is a set containing any subset of parties allowed to reconstruct the secret m. Γ is a monotonous access structure, i.e., $\emptyset \notin \Gamma$ and Γ is closed under taking supersets [Tho09]. On the other hand, Δ is the set containing all sets of parties forbidden to reconstruct the secret together. Δ is the complement of Γ regarding the powerset of parties. The linear integer secret sharing scheme (LISS) that they propose allows any set of parties $\in \Gamma$ to reconstruct the secret while any set of parties $\in \Delta$ are not able to reconstruct the secret. In contrast to a threshold secret sharing scheme, we can design the access structure as we wish, i.e., we can specify which parties should be able to reconstruct the secret, not just the number of parties.

A dealer shares a secret m within a publicly known interval $[-2^\lambda, 2^\lambda]$. Then the dealer samples a "distribution vector" $\mathbf{r}$ in $[-2^{\lambda_0+k}, 2^{\lambda_0+k}]^e$ where $\lambda_0 = \lambda + \lceil \log_2(\kappa_{\max}(e-1)) \rceil + 1$. k is the security parameter (the bigger k the more secure the scheme). The dealer then shares the share vector:

$$\mathbf{s} = (s_1, \ldots, s_d)^T = M\mathbf{r}$$

If there are n parties, one party gets a fair amount of shares l s.t. $ln = d$. Here, M is a Matrix M that a dealer must construct s.t.

- $M \in \mathbb{Z}^{d \times e}, \epsilon = (1, 0, \ldots, 0)^T \in \mathbb{Z}^e$
- If $A \in \Gamma$ then $\exists \lambda \in \mathbb{Z}^d$ s.t. $M_A^T \lambda = \epsilon$
- If $A \notin \Gamma$ then $\exists \kappa \in \mathbb{Z}^e$ s.t. $M_M \kappa = 0$ and $\langle \kappa, \epsilon \rangle = 1$

M_A denotes the rows of the Matrix M that correspond to the shares that the parties in a set A have access to, and $k_{\max} = \max\{|a||a \text{ is an entry in some sweeping vector } \kappa\}$. Furthermore, note that the restriction $\langle r, \epsilon \rangle = m$ must hold. All sets of parties $A \in \Gamma$ can reconstruct the secret by computing $\mathbf{s}_A^T \lambda_A = m$.

To show that parties $B \in \Delta$ should not be able to compute m, they show that the distribution of the shares for two secrets in $[-2^\lambda, 2^\lambda]$ and random coins $\mathrm{rc}, \mathrm{rc}'$ is almost identical since the statistical distance between the distributions of the shares $\{s_i(\mathbf{m}, \mathrm{rc}, k) | i \in B\}$ and $\{s_i(\mathbf{m}', \mathrm{rc}', k) | i \in B\}$ is negligible in the security parameter k.

This way, Damgård and Thorbek can solve the issues with distributed exponentiation. The parameters of their scheme remain relatively big. Notice that for certain constructions of M, $k_{\max} \in O(2^n)$. Other constructions, with smaller $k_{\max}$, have disadvantages in terms of construction time complexity or local computation time [Tho09].

While LISS is a general access structure scheme, we present threshold secret sharing schemes in this work. One might see this as a restriction, but threshold secret sharing is sufficient in many use cases (comp. threshold cryptography).

1.5.3 Gaussian Distributions

In the schemes we are going to present, we will hide the secrets using Gaussian distributed shares.

One key feature of Gaussians is that they are invariant under rotation. That means if we rotate samples of a Gaussian distribution, they still follow a Gaussian distribution. Our protocol will stretch and rotate (matrix M) a vector $\mathbf{r}$ that follows a Gaussian distribution. This means $M\mathbf{r}$ is still Gaussian distributed. We will see that given shares $M\mathbf{r}$ and a non-invertible M, there is still some (Gaussian distributed) variance of $\mathbf{r}$ left. We will make use of this leftover randomness to hide the secret m.

Since the publication of this paper and Thorbek's Ph.D. thesis about Linear Integer Secret Sharing (LISS), advancements in lattice-based cryptography have led to intensive research on discrete Gaussians, analog to continuous Gaussians in a discrete setting. A lot of interesting properties about them have been shown in the past two decades that will come in handy for integer secret sharing, as presented in this thesis.

2 Preliminaries

Now that we have introduced secret sharing and motivated our work, we provide the reader with preliminary information necessary to understand the secret sharing scheme over $\mathbb{Z}$ and $\mathbb{R}$ with their corresponding correctness and security proofs.

2.1 Negligibility

In cryptography, we are often not interested in proving that our protocol is perfectly secret, i.e., that an attacker learns nothing about the input we want to hide. It is usually sufficient to prove that the attacker learns only a negligible amount of information. This means that the attacker only learns information that does not practically help to decrypt a message or reveal a secret.

Definition 2.1 (Negligible Functions, [Bel97])**.** *A function $f(\lambda) : \mathbb{N} \to \mathbb{R}$ is **negligible** (in n) iff*

$$\forall c \in \mathbb{N} : \exists \lambda_0 \in \mathbb{N} : \forall \lambda > \lambda_0 : f(\lambda) \leq \lambda^{-c}$$

In other words: A negligible function must decrease asymptotically faster than the inverse of any polynomial.

Lemma 2.1 (Addition of Negligible Functions, [HL])**.** *If $f(\lambda)$ and $g(\lambda)$ are negligible functions then $f(\lambda) + g(\lambda)$ is a negligible function.*

Lemma 2.2 (Polynomials and Negligibility)**.** *If $f(\lambda)$ is a negligible function and $p(\lambda)$ is a polynomial then $p(\lambda) \cdot f(\lambda)$ is negligible.*

Proof. Let $p(\lambda)$ be a degree c polynomial of form $p(\lambda) = \sum_{i=0}^{c} a_i \lambda^i$. For all $\lambda \geq \max\{1, a_0 + \cdots + a_c\}$.

$$p(\lambda) = \sum_{i=0}^{c} a_i \lambda^i \leq \sum_{i=0}^{c} a_i \lambda^c = \lambda^c \sum_{i=0}^{c} a_i \leq \lambda^c \lambda = \lambda^{c+1}$$

Now, let d be arbitrary. Let $b = c + d + 1$. We know because $f(\lambda)$ is negligible that for this b, there must be some λ_0 such that for all $\lambda \geq \lambda_0$: $f(\lambda) \leq \lambda^{-b}$. Let $\lambda_0' = \max\{\lambda_0, 1, a_0 + \cdots + a_c\}$. Then, we have that for all $\lambda \geq \lambda_0'$:

$$\begin{aligned} p(\lambda) f(\lambda) &\leq p(\lambda) \lambda^{-(c+d+1)} \\ &= p(\lambda) \lambda^{-(c+1)} \lambda^{-d} \\ &\leq p(\lambda) p(\lambda)^{-1} \lambda^{-d} \\ &= \lambda^{-d} \end{aligned}$$

We conclude that $f(\lambda) \cdot p(\lambda)$ is negligible. □

2.2 Security Definitions for Secret Sharing

There are different definitions of security in cryptography and secret sharing. In the following part, we define the degree of security we want our secret sharing scheme to achieve. In order to get a good grasp on what the desired security looks like, we define a cryptographic game. λ is our security parameter in the following and defines how many input bits we have.

Definition 2.2 (Attacker)**.** *A secret sharing* ***attacker*** $\mathcal{A}_{t',n}$ *is an algorithm that has access to some* t' *out of* n *shares (typically* $t' < t$*, for a* (t, n) *secret sharing scheme) that stem from either one of two self-chosen secrets and that returns a guess on which of the secrets corresponds to the shares.*

Definition 2.3 (Secret Sharing Game)**.** *For a secret sharing scheme distribution algorithm* $\mathcal{D}$ *and an attacker* $\mathcal{A}_{t',n}$*, we define the secret sharing* ***game****:*

$$
\begin{array}{l}
\underline{\textit{SS-Game}_{\mathcal{D}}(\mathcal{A}_{t',n}, \lambda)} \\
b \leftarrow\!\!\$\ \{0,1\} \\
M_0, M_1, i_1, \ldots, i_k \leftarrow \mathcal{A}_{t',n}([], \lambda) \\
(s_1, \ldots, s_n) \leftarrow \mathcal{D}(M_b) \\
b' \leftarrow \mathcal{A}_{t',n}([s_{i_1}, \ldots, s_{i_k}], \lambda) \\
\textbf{return } b' == b
\end{array}
$$

where M_0, M_1 *are messages that the adversary can choose and* $\mathcal{D}(M)$ *returns the secret shares for some shared secret* M*.* $\leftarrow\!\!\$$ *means sample uniformly at random from a set.*

Intuitively, the game captures the following: The attacker chooses two messages M_0, M_1 and chooses which parties to corrupt before protocol execution (non-adaptive). Then, the adversary obtains the shares that the distribution algorithm $\mathcal{D}$ samples for a randomly selected message (from M_0, M_1, the attacker does not know which one). Last, the attacker must guess to which message the shares the attacker sees belong.

Definition 2.4 (Advantage)**.** *The secret sharing* ***advantage*** $\mathsf{Adv}_{\mathcal{D}}^{\textit{ss-game}}(\mathcal{A}_{t',n}, \lambda)$ *for some attacker* $\mathcal{A}_{t'}$ *and a secret sharing scheme* $\mathcal{D}$ *is defined as:*

$$
\mathsf{Adv}_{\mathcal{D}}^{\textit{ss-game}}(\mathcal{A}_{t',n}, \lambda) = \left|2 \cdot \Pr[\textit{SS-Game}_{\mathcal{D}}(\mathcal{A}_{t',n}, \lambda) \textit{ returns true}] - 1\right|
$$

If an attacker guesses b' uniformly at random, it wins the game with probability $\frac{1}{2}$, i.e., the advantage is 0. The $\mathsf{Adv}_{A}^{\text{game}}$ can be considered the advantage attacker A has to win the game compared to a randomly guessing attacker.

Definition 2.5 (Threshold Secret Sharing Scheme)**.** *We call a secret sharing scheme* $\mathcal{S} = (\mathcal{D}, \mathcal{R})$ *a* $(\mathbf{t}, \mathbf{n})^*$***-secret sharing scheme*** *if* n *shares are handed out by a dealer using the distribution algorithm* $\mathcal{D}$ *and*

- *parties with access to* t *shares can reconstruct the secret with probability* 1 *by using the reconstruction algorithm* $\mathcal{R}$*.*
- $\mathsf{Adv}_{\mathcal{D}}^{\textit{ss-game}}(\mathcal{A}_{t',n}, \lambda)$ *is negligible for any polynomial time attacker* $A_{t',n}$*,* $t' < t$*.*

If the advantage of the secret sharing game is negligible, the attacker cannot distinguish the shares of two self-chosen secrets. Note that we loosened our definition of secret sharing from perfect secrecy to statistical secrecy because perfect secrecy is unachievable for linear integer secret sharing schemes [Tho09]. After defining important properties that have to hold for our secret sharing scheme, we now define a notion of distance that will be important to analyze the quantity of security.

Definition 2.6 (Statistical Distance Cont., [DMR18]). *The **statistical distance** or total variation distance between two continuous distributions $\mathcal{A}, \mathcal{B}$ with their respective pdfs $p(x), q(x)$ over some set $S \subseteq \mathbb{R}^n$ is defined as:*

$$sd(\mathcal{A}, \mathcal{B}) = \frac{1}{2} \int_{x \in S} |p(x) - q(x)| dx$$

Lemma 2.3 (Stat. Distance & Computational Indistinguishability). *Let attacker $\mathcal{A}_{t',n}$ obtain secret sharing shares $\mathbf{s} = s_{j_1}, \ldots, s_{j_{t'}}$ that correspond to one of two adversarial chosen secrets M_0, M_1 with probability $\frac{1}{2}$ respectively. Let $sd(D_{\mathbf{s}_0}, D_{\mathbf{s}_1})$ be the statistical distance between $D_{\mathbf{s}_0}, D_{\mathbf{s}_1}$, where $D_{\mathbf{s}_i}$ is the distribution of the shares given M_i, from the view of the attacker.*
Now, let SS-Game' be a slightly modified version of SS-Game. Namely, it is guaranteed that the statistical distance $sd(D_{\mathbf{s}_0}, D_{\mathbf{s}_1})$ is independent of the attackers choice of M_0, M_1 and the parties $i_1, \ldots, i_{t'}$ to corrupt. In other words, for every output of the distribution algorithm, the statistical distance is upper bounded by some constant c. Then the attacker $\mathcal{A}_{t',n}$ against the security of the SS-Game' has a distinguishing advantage $\leq c$.

Proof. Intuitively we can see that the optimal attacker $\mathcal{A}^*_{t',n}$ does the following:

$\mathcal{A}^*_{t',n}(\mathbf{s} = (s_{j_1}, \ldots, s_{j_{t'}}), \lambda)$
if $\Pr(\mathbf{s}|M_0) \geq \Pr(s|M_1)$
  return 0
return 1

$\Pr(\mathbf{s}|M_i)$ means probability that we observe shares $\mathbf{s}$ given that the encrypted secret is M_i.
Now it becomes evident that $\mathsf{Adv}^{\text{ss-game}}_{\mathcal{S}}(\mathcal{A}^*_{t',n}, \lambda) = \text{sd}(D_{s_0}, D_{s_1})$ [Lau]. Because $\mathcal{A}^*_{t',n}$ is optimal, $\forall A_{t',n} : \mathsf{Adv}^{\text{ss-game}}_{\mathcal{S}}(\mathcal{A}_{t',n}, \lambda) \leq \text{sd}(D_{s_0}, D_{s_1})$. □

Remark 2.1. *Note that this Lemma only applies if the statistical distance $sd(D_{\mathbf{s}_0}, D_{\mathbf{s}_1})$ is independent of the (random) processes in the distribution algorithm D. We will see why this is important in the security proof for our secret sharing schemes.*

2.3 Lattices and Discrete Gaussians

Switching from the continuous case to the integer case, we introduce the notion of lattices and discrete Gaussians. Lattices can be considered some evenly spaced grid of points in $\mathbb{R}^n$. The simplest lattice is $\mathbb{Z}^n$. Lattices are an evenly spaced infinite discretization of $\mathbb{R}^n$.

Definition 2.7 (Lattice, [BLP+13]). *An n-dimensional (full-rank) **lattice** $\Lambda \subseteq \mathbb{R}^n$ is the set of all integer linear combinations of some set of n linearly independent basis vectors $B = \{\mathbf{b_1}, \ldots, \mathbf{b_n}\} \subseteq \mathbb{R}^n$:*

$$\Lambda = L(B) = \left\{ \sum_{i=1}^{n} z_i \mathbf{b_i} : z \in \mathbb{Z}^n \right\}$$

*The **dual lattice** of $\Lambda \subset \mathbb{R}^n$ is defined as $\Lambda^* = \{\mathbf{x} \in \mathbb{R}^n : \langle \Lambda, \mathbf{x} \rangle \subseteq \mathbb{Z}\}$*

Now, it is time to define discrete Gaussians. A discrete Gaussian is defined over a lattice and can be thought of as a continuous Gaussian sampled at the points of the lattice. The definition follows.

Definition 2.8 (Discrete Gaussian, [DKL+23] [AGHS13]). *A **discrete Gaussian** (with mean* $\mathbf{0}$*) over lattice* Λ *with parameter* $s > 0$ *is defined as follows:*

$$\forall x \in \Lambda, \mathcal{D}_{\Lambda,\sqrt{\Sigma}}(x) = \frac{\rho_{\sqrt{\Sigma}}(x)}{\rho_{\sqrt{\Sigma}}(\Lambda)}$$

where

$$\rho_{\sqrt{\Sigma}}(x) = \exp(-\pi x^T \Sigma^{-1} x)$$
$$\rho_{\sqrt{\Sigma}}(\Lambda) = \sum_{x' \in \Lambda} \rho_{\sqrt{\Sigma}}(x')$$

Definition 2.9 (Smoothing Parameter, [DKL+23]). *Given a lattice* Λ *and a positive real* $\epsilon > 0$*, the smoothing parameter* $\eta_\epsilon(\Lambda)$ *is the smallest real number* s *s.t.* $\rho_{1/s}(\Lambda^*\{0\})$.

Informally, the number $\eta_\epsilon(\Lambda)$ denotes the smallest amount of Gaussian blur that hides the discrete structure of the lattice when sampling from this distribution up to some error ϵ [CDLP14].

2.4 Gaussian Conditionals

In this section, we provide the reader with preliminary information about conditional distributions of Gaussians. In particular, how some Gaussian $\mathbf{r}$ is distributed given $M\mathbf{r}$ for some matrix M.

Lemma 2.4 (Gaussian Conditionals 1, similar to [DKL+23]). *Let* m, n *be integers. Let* $\Sigma \in \mathbb{R}^{n \times n}$ *be a positive definite matrix. Fix a full-rank matrix* $M \in \mathbb{R}^{m \times n}$ *and let* $\Sigma_y = M\Sigma M^T$ *and* $W = \Sigma M^T (\Sigma_y)^{-1}$ *and* $\Sigma_e = \Sigma - \Sigma M^T (\Sigma_y)^{-1} M\Sigma$.
If $\mathbf{r} \sim D_{\sqrt{\Sigma}}, \mathbf{y} \sim D_{\sqrt{\Sigma_y}}, \mathbf{e} \sim D_{\sqrt{\Sigma_e}}$ *then*

$$(\mathbf{r}, M\mathbf{r}) \approx_D (W\mathbf{y} + \mathbf{e}, \mathbf{y})$$

Where $\approx_D$ *means the two terms are identically distributed. One can also interpret this as: The distribution of* r *conditioned on* $M\mathbf{r} = \mathbf{y}$ *is* $W\mathbf{y} + \mathbf{e}$.

Proof. Assume M has full rank. Trivially, Σ_y, Σ_e are symmetric. Let:

$$\mathbf{k} = \begin{bmatrix} \mathbf{r} \\ M\mathbf{r} \end{bmatrix}$$
$$\mathbf{k}' = \begin{bmatrix} W\mathbf{y} + \mathbf{e} \\ \mathbf{y} \end{bmatrix}$$

Note that k and k' have expectation $\mathbf{0}$. We show that the covariance matrices are equal.

$$\begin{aligned}
\Sigma_1 &= \mathbb{E}\left[\mathbf{k}\mathbf{k}^T\right] \\
&= \begin{bmatrix} \mathbb{E}\left[\mathbf{r}\mathbf{r}^T\right] & \mathbb{E}\left[\mathbf{r}(M\mathbf{r})^T\right] \\ \mathbb{E}\left[(M\mathbf{r})\mathbf{r}^T\right] & \mathbb{E}\left[(M\mathbf{r})(M\mathbf{r})^T\right] \end{bmatrix} \\
&= \begin{bmatrix} \Sigma & \Sigma M^T \\ M\Sigma & M\Sigma M^T \end{bmatrix}
\end{aligned}$$

and

$$\begin{aligned}\Sigma_2 &= \mathbb{E}\left[\mathbf{k}'\mathbf{k}'^T\right] \\ &= \begin{bmatrix} W\mathbb{E}\left[\mathbf{y}\mathbf{y}^T\right]W^T + W\mathbb{E}\left[\mathbf{y}\mathbf{e}^T\right] + \mathbb{E}\left[\mathbf{e}\mathbf{y}^T\right]W^T + \mathbb{E}\left[\mathbf{e}\mathbf{e}^T\right] & W\mathbb{E}\left[\mathbf{y}\mathbf{y}^T\right] + \mathbb{E}\left[\mathbf{e}\mathbf{y}^T\right] \\ \mathbb{E}\left[\mathbf{y}\mathbf{y}^T\right]W^T + \mathbb{E}\left[\mathbf{y}\mathbf{e}^T\right] & \mathbb{E}\left[\mathbf{y}\mathbf{y}^T\right] \end{bmatrix} \\ &= \begin{bmatrix} W\Sigma_y W^T + \Sigma_e & W\Sigma_y \\ \Sigma_y W^T & \Sigma_y \end{bmatrix}\end{aligned}$$

Note that:

$$\begin{aligned}\Sigma_e &= \Sigma - \Sigma M^T(\Sigma_y)^{-1}M\Sigma \\ &= \Sigma - WM\Sigma \\ &\overset{(*)}{=} \Sigma - W\Sigma_y W^T\end{aligned}$$

Proof of $(*)$, use symmetry of Σ_y:

$$\begin{aligned}\Sigma_y W^T &= \Sigma_y(\Sigma M^T(\Sigma_y)^{-1}))^T \\ &= \Sigma_y((\Sigma_y)^T)^{-1}M\Sigma^T \\ &= \Sigma_y(\Sigma_y)^{-1}M\Sigma\end{aligned}$$

Thus $\Sigma_1 = \Sigma_2$ and $(\mathbf{r}, M\mathbf{r}) \approx_D (W\mathbf{y} + \mathbf{e}, \mathbf{y})$ □

Lemma 2.5 (Gaussian Conditionals 2)**.** *Let m, n be integers. Let $\Sigma \in \mathbb{R}^{n\times n}$ be a diagonal matrix with entries σ^2. Fix a matrix $M \in \mathbb{R}^{m\times n}$ and let $\Sigma_y = M\Sigma M^T$, $W = \Sigma M^T(\Sigma_y)^{-1}$, and $\Sigma_e = \Sigma - \Sigma M^T(\Sigma_y)^{-1}M\Sigma$. If $\mathbf{e} \sim D_{\sqrt{\Sigma_e}}$, and V a column permuted Matrix of V' where $M\sqrt{\Sigma} = UD(V')^T$ (SVD), then*

$$\Sigma_e = V\begin{bmatrix}\sigma^2 \cdot I_{n-m} & 0 \\ 0 & 0\end{bmatrix}V^T$$

Proof. [DKL+23] showed that for integers $m \leq n$, a full rank Matrix $A \in \mathbb{R}^{m\times n}$, and $A = USV^T$ be the singular value decomposition of A, we have that:

$$A^T(AA^T)^{-1}A = V\begin{bmatrix}I_m & 0 \\ 0 & 0\end{bmatrix}V^T$$

Now, if we substitute $A = M\sqrt{\Sigma}$ we get:

$$
\begin{aligned}
\Sigma_e &= \Sigma - \Sigma M^T (M\Sigma M^T)^{-1} M\Sigma \\
&= \sqrt{\Sigma}\sqrt{\Sigma} - \sqrt{\Sigma} V \begin{bmatrix} I_m & 0 \\ 0 & 0 \end{bmatrix} V^T \sqrt{\Sigma} \\
&= \sqrt{\Sigma} \left[I_n - V \begin{bmatrix} I_m & 0 \\ 0 & 0 \end{bmatrix} V^T \right] \sqrt{\Sigma} \\
&= \sqrt{\Sigma} \left[VV^T - V \begin{bmatrix} I_m & 0 \\ 0 & 0 \end{bmatrix} V^T \right] \sqrt{\Sigma} \\
&= \sqrt{\Sigma} V \left[I_n - \begin{bmatrix} I_m & 0 \\ 0 & 0 \end{bmatrix} \right] V^T \sqrt{\Sigma} \\
&= \sqrt{\Sigma} V \begin{bmatrix} 0 & 0 \\ 0 & I_{n-m} \end{bmatrix} V^T \sqrt{\Sigma}
\end{aligned}
$$

Let $\sqrt{\Sigma} = \sigma I$:

$$
\begin{aligned}
\Sigma_e &= \sigma I V \begin{bmatrix} 0 & 0 \\ 0 & I_{n-m} \end{bmatrix} V^T \sigma I \\
&= V\sigma^2 \begin{bmatrix} 0 & 0 \\ 0 & I_{n-m} \end{bmatrix} V^T \\
&= V \begin{bmatrix} 0 & 0 \\ 0 & \sigma^2 \cdot I_{n-m} \end{bmatrix} V^T
\end{aligned}
$$

By permuting the columns of V, we get:

$$
\Sigma_e = V' \begin{bmatrix} \sigma^2 \cdot I_{n-m} & 0 \\ 0 & 0 \end{bmatrix} (V')^T
$$

which is a SVD of Σ_e. Thus, the first $m - n$ singular values are σ^2, and the last m singular values are 0. □

2.5 Miscellaneous

Now, we prove other lemmata that we will use in our security proof.

Lemma 2.6 (Jointly Gaussian Independence)**.** *Let* $\mathbf{s}$ *be a vector with entries* $s_i \sim N(0,1)$ *and* V *a Matrix with orthonormal rows. Then the entries of* $Z = V\mathbf{s}$ *are independent and follow* $\sim N(0,1)$*.*

Proof. First, we prove that for Z the entries $Z_i \sim N(0,1)$. Let $\mathrm{Cov}(Z) = \mathrm{Cov}((VS)(VS)^T) = V\Sigma V^T$, where Σ is the covariance matrix of S, and $\Sigma = I$. Because $VV^T = I$, $\mathrm{Cov}(Z) = I$. To prove the independence, note from the latter that $Z_i, Z_j, i \neq j$ are uncorrelated and from [joi] that all entries of Z are jointly Gaussian.
From Theorem 1 of [joi], we see that they are also independent. □

Lemma 2.7 (An Upper Bound on Quadratic Forms)**.** *For any vector* $\mathbf{x} \in \mathbb{R}^n$ *and a positive semi-definite symmetric matrix* $\Sigma \in \mathbb{R}^{n \times n}$ *it holds that:*

$$\mathbf{x}^T \Sigma^{-1} \mathbf{x} \leq \mathbf{x}^T \frac{1}{\sigma_{min}(\Sigma)} I \mathbf{x}$$

Proof. Because Σ is symmetric, it can be decomposed into matrices AA^T. Let $A = UDV^T$ be the singular value decomposition of A. Then, $\Sigma = UD^2U^T$. Note that D^2 is a diagonal matrix with the singular values of Σ on its diagonal.
Furthermore $D^2 = \sigma_{min}(\Sigma)I + (D^2 - \sigma_{min}(\Sigma)I)$.

$$\begin{aligned}
\mathbf{x}^T \Sigma^{-1} \mathbf{x} &= \mathbf{x}^T (UD^2U^T)^{-1} \mathbf{x} \\
&= \mathbf{x}^T U (D^2)^{-1} U^T \mathbf{x} \\
&= (U^T \mathbf{x})^T (D^2)^{-1} (U^T \mathbf{x}) \\
&\text{substitute } \mathbf{y} = U^T \mathbf{x} \\
&= \mathbf{y}^T (D^2)^{-1} \mathbf{y} \\
&= \mathbf{y}^T \left(\frac{1}{\sigma_{min}(\Sigma)} I + \left((D^2)^{-1} - \frac{1}{\sigma_{min}(\Sigma)} I \right) \right) \mathbf{y} \\
&= \mathbf{y}^T \frac{1}{\sigma_{min}(\Sigma)} I \mathbf{y} + \mathbf{y}^T \left((D^2)^{-1} - \frac{1}{\sigma_{min}(\Sigma)} I \right) \mathbf{y} \\
&\text{right term is negative semi-definite and } U \text{ is unitary} \\
&\leq (U^T \mathbf{x})^T \frac{1}{\sigma_{min}(\Sigma)} I (U^T \mathbf{x}) = \mathbf{x}^T \frac{1}{\sigma_{min}(\Sigma)} I \mathbf{x}
\end{aligned}$$

□

3 Gaussian Linear Real Secret Sharing

In order to get an intuition of our proposal to use Gaussian distributions for integer secret sharing and as a proof of concept, we first look at the continuous case and show how we can implement a real-valued secret sharing. We call this scheme *Gaussian Linear Real Secret Sharing* (GLRSS).

Let parties $\mathcal{P} = \{P_1, \ldots, P_k\}$ get l shares each ($lk = n$). If we desire a (t, n)-secret sharing scheme we construct it like this:
Let $M \in \mathbb{R}^{n \times t}$ be a full-rank Matrix with entries drawn i.i.d. from a Gaussian distribution and $\mathbf{r}$ a multivariate Gaussian $\in \mathbb{R}^t$. Then, a dealer can construct a share vector $\mathbf{s}$:

$$\begin{bmatrix} s_1 \\ \vdots \\ s_n \end{bmatrix} = \begin{bmatrix} M_1 \\ \vdots \\ M_n \end{bmatrix} \begin{bmatrix} r_1 \\ \vdots \\ r_t \end{bmatrix}$$

Party P_i will get some shares $\boldsymbol{s_i} = (s_{i_1}, ..., s_{i_l})^T$ where the $\boldsymbol{s_i}$s form an equal partition on $\mathbf{s}$. They will also receive the corresponding rows M_i of the Matrix M.
Define g as the gap between the reconstruction threshold t and the security threshold t', i.e. $t = t' + g$. In most cases, we want $g = 1$.

The secret $\mathbf{m} \in \mathbb{R}^g$ will be additively hidden by an extractor as a random Matrix multiplication, $G \in \mathbb{R}^{g \times t}$.

$$\mathbf{c} = G\mathbf{r} + \mathbf{m}$$

Each party also receives $(G, \mathbf{c})$.

Theorem 3.1. *Let m be in a publicly known interval $[-2^\lambda, 2^\lambda]$, $\mathbf{r} \sim N(0, \sigma_r^2 I)$, the entries of G, $G_{ij} \sim N(0, \sigma_G^2)$, and M be a matrix with, $M_{ij} \sim N(0, \sigma_M^2)$. If we choose $\sigma_r = 2^{2\lambda}$, $\sigma_G = 2^\lambda$, $\sigma_M = 2^{n+\lambda}$, then GLRSS is a $(t, n)^*$-secret sharing scheme, secure under non-adaptive attacks.*

Here, non-adaptive means that the attacker has to choose the parties it will corrupt *before* the protocol is executed, as defined in the SS-Game. We will discuss the adaptive case in Section 3.4.

Proof. In order to prove that GLRSS is a $(t, n)^*$-secret sharing scheme, we first have to prove the correctness of the scheme and then prove security.

3.1 Correctness

It is straightforward that parties with access to t shares can reconstruct the secret $\mathbf{m}$. Let $I \subseteq [n]$ be the set of indices that correspond to those shares. Let $\mathbf{s}_I$ be the vector that comprises the t shares and $M_I \in \mathbb{R}^{t \times t}$ be the composition of the respective rows that created the t shares. Since M_I has full rank (elements of M and thus M_I in $\mathbb{R}$), we can compute $M_I^{-1}\mathbf{s}_I = \mathbf{r}$. Then, the parties can compute $\mathbf{c} - G\mathbf{r}$ to obtain the secret $\mathbf{m}$.

Note that computers don't have infinite precision. Thus, we want to guarantee that for all subsets $I \subseteq [n]$, the lowest singular value is sufficiently lower bounded, i.e., that we have $\Pr[\sigma_t(M) \leq c] \leq \epsilon$ for some constant c and some negligible function ϵ that we can choose. This is because we want to ensure that the matrix M has full rank.

By [Nie21] for some specific M_I we have that:

$$\Pr[\sigma_t(M) \leq \sigma_M \epsilon t^{-1}] \leq \epsilon$$

We see that M_I can be different for each I. Concretely, there are $\binom{n}{t}$ many matrices $M_I \in \mathbb{R}^{t \times t}$. The binomial coefficient is maximal if $\binom{n}{n/2}$. Now, using a union-bound argument, we see that:

$$\Pr[\exists M_I : \sigma_t(M_I) \leq \sigma_M \epsilon t^{-1}] \leq \binom{n}{n/2}\epsilon \leq 2^n \epsilon$$

using the well-known fact about the central binomial coefficient that $\binom{n}{n/2} \leq 2^n$. Choosing $\epsilon = 2^{-(\lambda+n)}$ and $\sigma_M = 2^{\lambda+n}$ suffices for our purposes.

3.2 Security

Let $M_{I'} \in \mathbb{R}^{t' \times t}$ be the Matrix corresponding to the attacker obtaining the shares $\mathbf{s}_{I'} \in \mathbb{R}^{t'}$ with $t' < t$, $I' \subset [n]$, $|I'| = t'$. $M_{I'}$ is now a fat matrix, and we can't invert it to obtain $\mathbf{r}$.
We want to show that for two secrets in the interval $[-2^\lambda, 2^\lambda]$, the ciphertexts are indistinguishable. Let Σ be the Covariance matrix of $\mathbf{r}$, where $\Sigma = \sigma_r^2 I_t$. From Lemma 2.4 we get that the residual

distribution of $\mathbf{r}$ is $D_{\sqrt{\Sigma_e}}$ with

$$\begin{aligned}\Sigma_e &= V \begin{bmatrix} \sigma_r^2 \cdot I_{t-t'} & 0 \\ 0 & 0 \end{bmatrix} V^T \\ &= V \begin{bmatrix} \sigma_r^2 \cdot I_g & 0 \\ 0 & 0 \end{bmatrix} V^T\end{aligned}$$

where V is an orthonormal matrix depending on $M_{I'}$.
Obviously, $\mathbf{c}$ follows a Gaussian distribution with $\mathbb{E}[\mathbf{c}] = \mathbf{m}$. Now, we want to analyze the covariance matrix $\tilde{\Sigma}$ of $\mathbf{c}$.

$$\begin{aligned}\tilde{\Sigma} &= \mathrm{Var}[\mathbf{c}] = \mathrm{Var}[G\mathbf{r} + \mathbf{m}] = \mathrm{Var}[G\mathbf{r}] = \mathrm{Var}[G\mathbf{e}] \\ &= \mathbb{E}\left[(G\mathbf{e})(G\mathbf{e})^T\right] = G\mathbb{E}\left[\mathbf{e}\mathbf{e}^T\right] G^T = G\Sigma_e G^T \\ &= GV \begin{bmatrix} \sigma_r^2 \cdot I_g & 0 \\ 0 & 0 \end{bmatrix} V^T G^T \\ &= (GVP)(GVP)^T, P = \begin{bmatrix} \sigma_r \cdot I_g & 0 \\ 0 & 0 \end{bmatrix}\end{aligned}$$

GVP is the following matrix, where $\mathbf{g}_i$ is the ith row of G and $\mathbf{v}_i$ is the ith row of V.

$$\begin{bmatrix} \langle \mathbf{g}_1, \sigma_r \mathbf{v}_1 \rangle & \dots & \langle \mathbf{g}_1, \sigma_r \mathbf{v}_g \rangle & 0 & \dots & 0 \\ \vdots & & \vdots & \vdots & & \vdots \\ \langle \mathbf{g}_g, \sigma_r \mathbf{v}_1 \rangle & \dots & \langle \mathbf{g}_g, \sigma_r \mathbf{v}_g \rangle & 0 & \dots & 0 \end{bmatrix}$$

Now $(GVP)(GVP)^T$:

$$\begin{bmatrix} \sum_{j=1}^g \langle \mathbf{g}_1, \sigma_r \mathbf{v}_j \rangle^2 & \dots & \sum_{j=1}^g \langle \mathbf{g}_1, \sigma_r \mathbf{v}_j \rangle \langle \mathbf{g}_g, \sigma_r \mathbf{v}_j \rangle \\ \sum_{j=1}^g \langle \mathbf{g}_2, \sigma_r \mathbf{v}_j \rangle \langle \mathbf{g}_1, \sigma_r \mathbf{v}_j \rangle & \dots & \sum_{j=1}^g \langle \mathbf{g}_2, \sigma_r v_j \rangle \langle \mathbf{g}_g, \sigma_r \mathbf{v}_j \rangle \\ \vdots & & \vdots \\ \sum_{j=1}^g \langle \mathbf{g}_g, \sigma_r v_j \rangle \langle \mathbf{g}_1, \sigma_r \mathbf{v}_j \rangle & \dots & \sum_{j=1}^g \langle \mathbf{g}_g, \sigma_r \mathbf{v}_j \rangle^2 \end{bmatrix}$$

Now, let $G' = \frac{G}{\sigma_G} V_{1:g}$ denotes the first g columns of V. We see that:

$$\begin{aligned}(GVP)(GVP)^T &= \sigma_r^2 \sigma_G^2 \left[\left(G'V \begin{bmatrix} I_g & 0 \\ 0 & 0 \end{bmatrix} \right) \left(G'V \begin{bmatrix} I_g & 0 \\ 0 & 0 \end{bmatrix} \right)^T \right] \\ &= \sigma_r^2 \sigma_G^2 \left[\left(G'V_{1:g} \right) \left(G'V_{1:g} \right)^T \right]\end{aligned}$$

The entries of G' follow $N(0,1)$.

Using Lemma 2.6 and the independence of all entries in G' we get that $G'V_{1:g}$ and $(G'V_{1:g})^T$ are $g \times g$ matrices with entries i.i.d drawn from $N(0,1)$ For simplicity we say $\tilde{G} = G'V_{1:g}$.

Now, we want to analyze the singular values of the covariance matrix $\tilde{\Sigma}$.

$$\sigma_i(\tilde{\Sigma}) = \sigma_i((GVP)(GVP)^T) = \sigma_r^2\sigma_G^2 \cdot \sigma_i\left(\tilde{G}\tilde{G}^T\right)$$

Note that $\tilde{G}\tilde{G}^T$ is a so called Wishart-Matrix and by using [TV09] we get that for all $\epsilon \geq 0$:

$$\Pr[\sigma_g(\tilde{G}\tilde{G}^T) \geq \epsilon g^{-1}] \geq 1 - \epsilon$$

We can choose ϵ appropriately (e.g. $\epsilon = 2^\lambda$). If we combine our previous results, we get:

$$\Pr[\sigma_g(\tilde{\Sigma}) \geq \epsilon\sigma_r^2\sigma_G^2 g^{-1}] \geq 1 - \epsilon$$

Knowing the spectral bounds of $\tilde{\Sigma} = \mathrm{Cov}(G\mathbf{r} + \mathbf{m})$, we can calculate the statistical distance for two secrets $\mathbf{m}_0$, $\mathbf{m}_1$. The $G\mathbf{r}$ term acts as a mask for our secret such that we learn only negligible information about the secret from the ciphertext.
We see that:

$$\begin{aligned}
\tilde{\Sigma} &= (GVP)(GVP)^T \\
&\quad \text{substitute } GVP \text{ by its SVD to obtain} \\
&= UDU^T \\
D^{-1} &= \frac{1}{\sigma_{min}(\tilde{\Sigma})}I + (D^{-1} - \frac{1}{\sigma_{min}(\tilde{\Sigma})}I)
\end{aligned}$$

Note that $\frac{1}{\sigma_{min}}I$ is positive definite and $(D^{-1} - \frac{1}{\sigma_{min}})$ is negative semi-definite. We now want to analyze the statistical distance between the ciphertext of two secrets m_0 and m_1. Proposition 2.1. of [DMR18] shows that the statistical distance between two multivariate Gaussians $\mathbf{c}_0$ and $\mathbf{c}_1$ with $\mathbf{c}_0 \sim N(\mathbf{m}_0, \tilde{\Sigma}), \mathbf{c}_1 \sim N(\mathbf{m}_1, \tilde{\Sigma})$ is:

$$\mathrm{sd}(\mathbf{c}_0, \mathbf{c}_1) \leq \frac{1}{2}\sqrt{(\mathbf{m}_0 - \mathbf{m}_1)^T\tilde{\Sigma}^{-1}(\mathbf{m}_0 - \mathbf{m}_1)}$$

We can bound the statistical distance between $\mathbf{c}_0, \mathbf{c}_1$ using Lemma 2.7. Note that the term in the following square root achieves its maximum if the secrets lie at the edge of the publicly known interval, i.e. $|\mathbf{m}_0 - \mathbf{m}_1| = [2^{\lambda+1}, \ldots, 2^{\lambda+1}]^T = \mathbf{m}^*$.

Let's also set $2^{-\lambda} = \epsilon = \sigma_G^{-1}$. Then, for all secrets $\mathbf{m}_0, \mathbf{m}_1$:

$$\begin{aligned}
\mathrm{sd}(C_0, C_1) &\leq \frac{1}{2}\sqrt{(\mathbf{m}_0 - \mathbf{m}_1)^T \frac{1}{\sigma_{min}(\tilde{\Sigma})} I(\mathbf{m}_0 - \mathbf{m}_1)} \\
&\leq \frac{1}{2}\sqrt{(\mathbf{m}^*)^T \frac{1}{\sigma_{min}(\tilde{\Sigma})} I(\mathbf{m}^*)} \\
&\leq \frac{1}{2}\sqrt{\sum_{i=1}^{t} \frac{2^{\lambda+1} \cdot 2^{\lambda+1} \cdot g}{\epsilon \sigma_r^2 \sigma_G^2}} \\
&= \frac{1}{2}\sqrt{\frac{tg2^{2\lambda+2}}{\sigma_r^2}} \\
&= \sqrt{tg}\frac{2^{\lambda}}{\sigma_r}
\end{aligned}$$

This term is negligible in our security parameter λ if we choose $\sigma_r = 2^{2\lambda}$.
Now, stating that $\mathsf{Adv}_{\mathcal{D}}^{\text{ss-game}}(\mathcal{A}_{t',n}, \lambda)$ is negligible for any attacker $\mathcal{A}_{t',n}$ using Lemma 2.3, does not instantly work. Notice that the bound on the statistical distance only holds if the lowest singular value is big enough, as described in Remark 2.1. As stated above, we set $\epsilon = 2^{-\lambda}$ for the bound:

$$\Pr[\sigma_g(\tilde{\Sigma}) \leq \epsilon \sigma_r^2 \sigma_G^2 g^{-1}] \leq \epsilon$$

We call the event that the lowest singular value is too little "$\sigma_g(\tilde{\Sigma}) \leq c$". Recall that in the game SS-Game' it is guaranteed that the statistical distance $\mathrm{sd}(\mathbf{c}_0, \mathbf{c}_1)$ is independent of the random processes in the distribution algorithm D. In our case, we assume that $\sigma_g(\tilde{\Sigma}) \leq c$ does not happen, i.e., we have an upper bound of $\mathrm{sd}(\mathbf{c}_0, \mathbf{c}_1)$ that always holds. Using a union bound, we see that:

$$\begin{aligned}
&\mathsf{Adv}_{\mathcal{D}}^{\text{ss-game}}(\mathcal{A}_{t',n}, \lambda) \\
&\leq \Big|2 \cdot \Pr[SS - Game'_{\mathcal{S}}(\mathcal{A}_{t',n}, \lambda) \text{ returns true} \cup \text{"}\sigma_g(\tilde{\Sigma}) \leq c\text{"}] - 1\Big| \\
&\leq \Big|2 \cdot \Big(\Pr[SS - Game'_{\mathcal{S}}(\mathcal{A}_{t',n}, \lambda) \text{ returns true}] + \Pr[\text{"}\sigma_g(\tilde{\Sigma}) \leq c\text{"}]\Big) - 1\Big| \\
&= \Big|2 \cdot \Pr[SS - Game'_{\mathcal{S}}(\mathcal{A}_{t',n}, \lambda) \text{ returns true}] - 1\Big| + 2 \cdot \Pr[\text{"}\sigma_g(\tilde{\Sigma}) \leq c\text{"}] \\
&= \mathsf{Adv}_{\mathcal{S}}^{\text{ss-game}'}(\mathcal{A}_{t',n}, \lambda) + 2 \cdot \Pr[\text{"}\sigma_g(\tilde{\Sigma}) \leq c\text{"}] \\
&= \mathrm{sd}(\mathbf{c}_0, \mathbf{c}_1) + 2 \cdot \Pr[\text{"}\sigma_g(\tilde{\Sigma}) \leq c\text{"}]
\end{aligned}$$

This term is negligible in λ using Lemma 2.1 and Lemma 2.2. Note that we always assume that λ is big enough s.t. the constant factors cancel out. Conclusively, through proofing correctness and security, we have that GLRSS is a $(t, n)^*$-secret sharing scheme. □

3.3 Example

Let us have 20 parties $P_1, .., P_{20}$. Every party gets 1 share. The shared secret lies in a publicly known interval $[-2^{\lambda}, 2^{\lambda}]$. We want 10 parties to be able to reconstruct the secret ($t = 10$). The attacker compromises 9 parties and obtains $t' = 9$ shares. Given those shares, the ciphertexts $\mathbf{c}_0, \mathbf{c}_1$ where $\mathbf{c}_i = G\mathbf{r} + \mathbf{m_i}$, are statistically indistinguishable if $\sigma_r = 2^{2l}$ and $\sigma_G = 2^l$, i.e. the statistical difference between $\mathbf{c}_0, \mathbf{c}_1$ is negligible in λ.

3.4 Adaptive Case

Theorem 3.2. *Let m be on a publicly known interval $[-2^\lambda, 2^\lambda]$, $\mathbf{r} \sim N(0, \sigma_r^2 I)$, the entries of G, $G_{ij} \sim N(0, \sigma_G^2)$, and M be a matrix with, $M_{ij} \sim N(0, \sigma_M^2)$. If we choose $\sigma_r = 2^{2\lambda}$, $\sigma_G = 2^{n+\lambda}$, $\sigma_M = 2^{n+\lambda}$, then GLRSS is a $(t, n)^*$-secret sharing scheme, secure under adaptive attacks.*

We have previously considered the non-adaptive case. That means that the attacker has to choose the parties that it wants to corrupt before protocol execution. Now, we want to analyze the adaptive case, i.e., the attacker can see public information (other than the bound in which the secret lies), such as the ciphertext or rows of the Matrix M. One tactic is to choose the parties whose rows of the Matrix M result in a mask $G\mathbf{r}$ with minimal variance.

The proof is almost the same as for the non-adaptive case. There is one major change. For the bound of the lowest singular value, we had that:

$$\Pr[\sigma_g(\tilde{\Sigma}) \leq \epsilon \sigma_r^2 \sigma_G^2 g^{-1}] \leq \epsilon$$

Now, the adversary can choose any subset of rows of the matrix $M \in \mathbb{R}^{n \times t}$ to obtain some compound Matrix $M_{I'}$, i.e. $\tilde{\Sigma}$ is different for all of those. Concretely, there are $\binom{n}{t'}$ many matrices $M_{I'} \in \mathbb{R}^{t' \times t}$. Note that the binomial coefficient is maximal if $\binom{n}{n/2}$. We see, similarly to the reconstruction union bound, that:

$$\Pr[\exists \tilde{\Sigma} : \sigma_g(\tilde{\Sigma}) \leq \epsilon \sigma_r^2 \sigma_G^2 g^{-1}] \leq \binom{n}{n/2} \epsilon \leq 2^n \epsilon$$

Naturally, we have to set $\epsilon = 2^{-(n+\lambda)}$ instead of the previous $2^{-\lambda}$.

To account for this factor of 2^n, we have to adjust σ_G or σ_r. If we choose $\sigma_G = 2^{n+\lambda}$, for example, we achieve the desired statistical distance that is negligible in λ. We can make the same union bound argument as above and get that $\mathsf{Adv}_{\mathcal{D}}^{\text{ss-game}^*}(\mathcal{A}_{t',n}, \lambda)$ is negligible for the adaptive secret sharing game SS-game*. As a drawback, we get share sizes with additional n bits.

4 Gaussian Approximate Linear Integer Secret Sharing

Now, we want to discuss the integer case. This scheme is called *Gaussian Approximate Linear Integer Secret Sharing* (GALISS). In this work, we only discuss the proof for reconstruction, not the security proof, even though we give motivation on how such a proof might look like. We want to explicitly state that we **do not** guarantee the security of GALISS.

While working on this thesis, we discovered that integer secret sharing, using this construction, turns out to be more complicated than we initially thought. This is mainly because it is not guaranteed that the reconstruction matrix M_I with integer entries has an inverse M_I^{-1} with integer entries (e.g., $[2]^{-1} = [0.5]$). This only happens if M_I is a unimodular matrix, i.e. $\det(M_I) \in \{-1, 1\}$. So we must guarantee that for all possible reconstruction matrices $I \subseteq [n]$ of size $t \times t$, $\det(M_I) \in \{-1, 1\}$. To our knowledge, we cannot efficiently sample such a random matrix. A remedy would be to scale and round M_I^{-1} and then, when we reconstruct, get some scaled version of $\mathbf{r}$ with some minor errors. However, this violates linearity, and thus, we have to consider the integer version as an approximate linear integer sharing scheme. Let $M \in \mathbb{Z}^{n \times t}$, $\mathbf{r} \in \mathbb{Z}^t$, $\mathbf{g} \in \mathbb{Z}^t$. The dealer executes this protocol, for some secret $m \in [-2^\lambda, 2^\lambda]$:

- Sample the entries M from $\mathcal{D}_{\mathbb{Z}, \sigma_M^2}$.

- Check if M has full rank; otherwise, reject and sample again.
- Sample the entries of $\mathbf{r}$ and $\mathbf{g}$ from $\mathcal{D}_{\mathbb{Z},\sigma_r^2}$ and $\mathcal{D}_{\mathbb{Z},\sigma_g^2}$ respectively.
- Compute $\mathbf{s} = M\mathbf{r}$ and distribute the entries of $\mathbf{s}$ as shares to the parties.
- Compute $\mathbf{c} = \langle \mathbf{g}, \mathbf{r} \rangle + m$ and publish $\mathbf{c}, \mathbf{g}$.

We now do not sample from continuous Gaussians but from discrete Gaussians defined over lattices ($\mathbb{Z}^t$ in our case). Furthermore, note that we only share one secret at a time $\in [-2^\lambda, 2^\lambda]$.

4.1 Correctness / Reconstruction

First, we want to prove that reconstruction works and that the scheme is correct. The reconstruction algorithm does the following:

- Choose γ sufficiently large (see below).
- t parties jointly compute $M_I^{-1} \in \mathbb{R}^{t \times t}$, where $I \subseteq [n]$.
- M_I^* is the scaled and rounded inverse $M_I^* = \lfloor \gamma \cdot M_I^{-1} \rceil \in \mathbb{Z}^{t \times t}$.
- compute $\mathbf{r}^* = \lfloor \frac{M_I^* \mathbf{s}_I}{\gamma} \rceil$.
- Obtain the secret by computing $\mathbf{m} = \mathbf{c} - \langle \mathbf{g}, \mathbf{r}^* \rangle$.

Notice that now M_I^* is an integer matrix.

Proof. If we prove that $\mathbf{r}^* = \mathbf{r}$ correctness follows trivially. Now:

$$\begin{aligned} M_I^* &= \lfloor \gamma \cdot M_I^{-1} \rceil = \gamma \cdot M_I^{-1} + \Delta, \Delta \in [-\frac{1}{2}, \frac{1}{2}]^{t \times t} \\ M_I^* \mathbf{s}_I &= [\gamma \cdot M_I^{-1} + \Delta] M_I \mathbf{r} \\ &= \gamma \cdot M_I^{-1} M_I \mathbf{r} + \Delta M_I \mathbf{r} \\ &= \gamma \cdot \mathbf{r} + \Delta M_I \mathbf{r} \end{aligned}$$

Note that the following implication holds:

$$\|\Delta M_I \mathbf{r}\| < \frac{\gamma}{2} \Rightarrow \mathbf{r} = \lfloor \frac{M_I^* \mathbf{s}_I}{\gamma} \rceil$$

We want to choose γ s.t. the premise holds. Let $\Delta M_I = \mathbf{e}$. Then, for any column at index c and row at index r of $\mathbf{e}$:

$$-\frac{1}{2} \sum_{i=1}^{t} m_{ij} \leq e_{rc} \leq \frac{1}{2} \sum_{i=1}^{t} m_{ij}$$

Let $\mathbf{e}' = \Delta M_I \mathbf{r} = \mathbf{er}$. Then for all $k, 1 \leq k \leq t$:

$$\begin{aligned} e'_k &= \sum_{i=1}^{t} e_{ki} r_i \\ \iff (e'_k)^2 &\leq \left(\sum_{j=1}^{t} \left[\frac{1}{2} \sum_{i=1}^{t} m_{ij} \right] r_j \right)^2 \\ &= \frac{1}{4} \left(\sum_{j=1}^{t} r_j \left[\sum_{i=1}^{t} m_{ij} \right] \right)^2 \end{aligned}$$

We can analyze the norm of $\Delta M_I \mathbf{r}$:

$$\begin{aligned} \|\Delta M_I r\| &\leq \sqrt{\frac{1}{4} t \left(\sum_{j=1}^{t} r_j \left[\sum_{i=1}^{t} m_{ij} \right] \right)^2} \\ &= \frac{1}{2} \sqrt{t} \left| \left(\sum_{j=1}^{t} r_j \left[\sum_{i=1}^{t} m_{ij} \right] \right) \right| \end{aligned}$$

Choose γ as:

$$\gamma > \sqrt{t} \left| \left(\sum_{j=1}^{t} r_j \left[\sum_{i=1}^{t} m_{ij} \right] \right) \right|$$

such that the premise holds and we have that $\mathbf{r}^* = \mathbf{r}$. Note that the entries of $\mathbf{r}$ and M are upper bounded (up to some negligible probability), and thus, we can choose γ appropriately. To give an exact value for γ, we would need to know the variance of the normal distributions from which the entries of M and $\mathbf{r}$ are sampled. Those variances need to be determined in a possible security proof. □

4.2 Security

We do not guarantee that GALISS is secure. A possible security proof is up to future work on this topic. If a proof of security exists, it could look similar to the one for GLRSS. It has been shown that discrete Gaussians in the "smoothing regime" (comp. Def. 2.9) essentially behave like continuous Gaussians [Pei16] [Reg04].

5 Discussion and Comparison to existing protocols

We have successfully proposed two secret sharing schemes that use Gaussian distributions to hide a secret. We want to compare those protocols to existing ones and discuss interesting properties.

5.1 Gap Between Reconstruction and Security Threshold

Note that we can vary the gap g between the reconstruction and security thresholds. This way, we can share multiple secrets on behalf of having a bigger gap. In most cases, $g = 1$ will be the most useful.

If we, for example, want to share 2 secrets, we have to choose gap $g = 2$. Otherwise, an attacker with access to $t' = t - 1$ could have a higher chance of distinguishing between secrets. One possibility to share more than one secret while maintaining the same number of parties is to give the parties bundles of shares. Now, the dealer wants to share $g = 3$ secrets; there are $n_p = 10$ parties altogether, and $t_p = 5$ parties should be able to reconstruct the secret. Then one can set $n = n_p \cdot g = 30$, $t = t_p \cdot g = 15$. Note that $t_p - 1 = 4$ parties now have access to $t' = (t_p - 1) \cdot g = 12$ shares. Thus, we have a gap of 3 between the security threshold and the reconstruction threshold and still maintain the security that $t_p - 1$ parties (*not* $t - 1$ shares) learn a negligible amount of information about the secret. However, now the share sizes are bigger.

5.2 General Access Structure vs. Thresholds

While [Tho09] uses general access structures, i.e., the dealer can choose which parties should be able to reconstruct the secrets, we use a threshold secret sharing scheme, i.e., any t parties can reconstruct the secret.

The former has the advantage that, in practice, the dealer may be biased towards which parties the dealer trusts more and which are less reliable [Tho09]. The dealer can then come up with an access structure in which a smaller number of reliable parties and a larger number of not-so-reliable parties can reconstruct the secret. However, the construction of the scheme is also more complicated, as the matrix M with which [Tho09] [DT06] multiply their secret vector with has to be constructed carefully, and its computation is time-consuming. This problem does not arise with the threshold setting.

5.3 Statistical Security vs. Perfect Secrecy

Secret sharing schemes like Shamir's scheme or additive secret sharing guarantee perfect secrecy, meaning the distribution of the secret given the shares is the same as the distribution of the secret without the shares. Our scheme only achieves statistical security.

We want to provide an intuition of why having perfect secrecy in a linear integer secret sharing scheme is not achievable. The most important thing is that for a linear finite number secret sharing scheme, we have that the linearity of the shares is given concerning mod q where q is the size of the group. In this finite group, the secrets appear uniformly at random before protocol execution. When the shares are linearly added to the secret with some coefficient, they still seem uniformly distributed in this group. One of the main reasons for this is that we use modular arithmetic and that numbers bigger than q will be again projected to the finite group through the $\bmod$ operation. We can think of the operations as being "circular" over this group.

However, with integer secrets, $\mathbb{Z}$ is not "circular," i.e., a uniform distribution (not even properly defined on the integers) is not maintainable and we must rely on statistical security.

5.4 Why Do We Need to Restrict the Interval of the Secret?

The infinite nature of $\mathbb{Z}$ is also why we have to restrict the interval of the secret if we want to achieve statistical security.

The main idea in integer secret sharing schemes is that we hide the secret, i.e., some number in $\mathbb{Z}$ with some high variance mask that is unknown to a set of unqualified parties but known to a qualified set. If we have an interval where the secret lies, we must ensure that even for the secrets at the edges of the interval, the variance σ^2 of this mask is still large enough to hide the secrets under the mask. If we do not have bounds on the secrets, the secrets can be infinitely far apart, and therefore, we can't find a corresponding mask with bounded variance σ^2. In other words, the variance must be infinitely large.

5.5 Linearity

Because we only do linear operations in our scheme, we have that GLRSS is a linear secret sharing scheme, i.e., the secret is a linear combination of the shares. GALISS is not linear but approximately linear. That means it seems linear, depending on the application. Note that Shamir or additive secret sharing is also linear but over a finite group. As mentioned in the introduction, not all schemes are linear. Linearity has desirable properties that we discuss in Section 6. Table 1 represents which secret sharing schemes can achieve specific security definitions. Note again that linear secret sharing

	non-linear	linear
finite	perfect secrecy, e.g. [BI05]	perfect secrecy, e.g. Shamir's scheme
countable infinite	perfect secrecy, e.g. CRT schemes	statistical security, discussed in [Tho09]

Table 1: Achievable security definitions for finite/infinite and linear/non-linear schemes

schemes over a countable infinite domain can only achieve statistical security if the interval of the secret is restricted.

5.6 Verifiable Linear Integer Secret Sharing

So far, we have only considered an honest-but-curious scenario. That means the parties are interested in revealing the secret but don't cheat or deviate from the protocol. Several secret sharing schemes are *verifiable*, i.e., the parties can prove that they provided the correct input. Nevertheless, how can the other participants know that party i did not cheat when i wants to keep their input secret? Party i cannot just reveal its secret information in plain text.

This is not only a problem restricted to secret sharing but a general problem in cryptography. How can we prove something without revealing the "something"? The so-called zero-knowledge proof is the most popular protocol type to prove that a party provided a correct quantity or possesses some quantity without revealing the quantity.

Although a proof on how to make GALISS / GLRSS verifiable exceeds the scope of this thesis, several ideas on how to make integer secret sharing and related protocols verifiable have been provided in [Tho09]. Thorbek, for example, presents a distributed exponentiation protocol that uses his scheme in which the parties prove in zero-knowledge that their contributions are correct.

6 Applications

We will now discuss two possible applications for GLRSS and GALISS.

6.1 Distributed Exponentiation

Distributed exponentiation was motivated in the introduction. First, we want to show that for some number a and secret m, parties with access to t shares can compute a^m with their shares. We will present a successful protocol for GLRSS, which is linear, and show why this simple protocol does not easily extend to the approximate linear case (GALISS). This illustrates the disadvantages of an approximate linear secret sharing scheme. However, first, we want to show the correctness of GLRSS. The proof is similar to the one in [Tho09].

6.1.1 GLRSS

Let us first discuss the continuous/real case. Let $M_I \in \mathbb{R}^{t\times t}$ be the reconstruction matrix the t computing parties have access to. By assumption, M_I has full rank. Thus the parties can jointly compute M_I^{-1}.

$$M_I^{-1} = \begin{bmatrix} — & (\mathbf{m}_1)^T & — \\ & \vdots & \\ — & (\mathbf{m}_t)^T & — \end{bmatrix}$$

Let those r parties have access to l shares ($lr = t$). Then we can directly see that

$$\mathbf{s}_I = \sum_{1\le i\le r} (0,\dots,s_{i_1},\dots,0,\dots,s_{i_l},\dots,0)^T = \sum_{1\le i\le r} \mathbf{s}_{(i)}$$

and each i party possesses $\mathbf{s}_{(i)}$. Let $\mathbf{g}_x$ be the xth row of $G \in \mathbb{R}^{g\times t}$. Now, the secret can be reconstructed by:

$$\begin{aligned}\mathbf{m} &= \mathbf{c} - G(M_I^{-1}\mathbf{s}_I) = \mathbf{c} - (GM_I^{-1}) \sum_{1\le i\le r} \mathbf{s}_{(i)} \\ &= \mathbf{c} - \sum_{1\le i\le r} \left(\begin{bmatrix} \langle \mathbf{g}_1, \mathbf{m}_1\rangle & \langle \mathbf{g}_1, \mathbf{m}_2\rangle & \dots & \langle \mathbf{g}_1, \mathbf{m}_t\rangle \\ \vdots & \vdots & & \vdots \\ \langle \mathbf{g}_g, \mathbf{m}_1\rangle & \langle \mathbf{g}_g, \mathbf{m}_2\rangle & \dots & \langle \mathbf{g}_g, \mathbf{m}_t\rangle \end{bmatrix} s_{(i)} \right)\end{aligned}$$

Each party has access to M_I^{-1}, G and can thus compute:

$$\begin{bmatrix} \langle \mathbf{g}_1, \mathbf{m}_1\rangle & \langle \mathbf{g}_1, \mathbf{m}_2\rangle & \dots & \langle \mathbf{g}_1, \mathbf{m}_t\rangle \\ \vdots & \vdots & & \vdots \\ \langle \mathbf{g}_g, \mathbf{m}_1\rangle & \langle \mathbf{g}_g, \mathbf{m}_2\rangle & \dots & \langle \mathbf{g}_g, \mathbf{m}_t\rangle \end{bmatrix} \mathbf{s}_{(i)} = \sum_{1\le j\le l} \begin{bmatrix} \langle \mathbf{g}_1, \mathbf{m}_{i_j}\rangle s_{i_j} \\ \vdots \\ \langle \mathbf{g}_g, \mathbf{m}_{i_j}\rangle s_{i_j} \end{bmatrix}$$

Note that in our scheme we share secrets $m_1,\dots,m_g$. We can share secret m_z by computing:

$$m_z = c_z - \sum_{1\le i\le r} \left(\sum_{1\le j\le l} \langle \mathbf{g}_z, \mathbf{m}_{i_j}\rangle s_{i_j} \right)$$

We get that:

$$a^{m_z} = \frac{a^{c_z}}{a^{\sum_{1\leq i\leq r}\left(\sum_{1\leq j\leq l}\langle \mathbf{g}_z, \mathbf{m}_{i_j}\rangle s_{i_j}\right)}} = \frac{a^{c_z}}{\prod_{1\leq i\leq r} a^{\sum_{1\leq j\leq l}\langle \mathbf{g}_z, \mathbf{m}_{i_j}\rangle s_{i_j}}}$$

Let the parties provide $a_{i_j} = a^{s_{i_j}}$ and note that $\langle \mathbf{g}_z, \mathbf{m}_{i_j}\rangle = \alpha_{z,i_j}$ can be pre-computed by any participating party. Thus, we finally get that:

$$a^{m_z} = \frac{a^{c_z}}{\prod_{1\leq i\leq r}\prod_{1\leq j\leq l} a_{i_j}^{\alpha^z_{i_j}}}$$

Because all parts of the term are accessible to each party now, the parties can compute a^{m_z} for any of the secrets m_z.
In summary, the protocol looks as follows:

1. The dealer secret-shares any secret $\mathbf{m}$.
2. The parties agree on the secret at position z.
3. Every party computes $a_{i_j} = a^{s_{i_j}}$ for each of the shares s_{i_j} they have and posts them to the other parties.
4. Every party can compute a^{m_z} and $\alpha^z_{i_j} = \langle \mathbf{g}_z, \mathbf{m}_{i_j}\rangle$.
5. Every party can now compute a^{m_z} by the formula above.

6.1.2 GALISS

Let us now have a look at what the integer case looks like. The biggest difference is that we divide by γ in the end and round. As stated above, we show why we cannot simply extend the linear scheme to the approximate linear scheme.
Let $M_I \in \mathbb{Z}^{t\times t}$ be the reconstruction matrix the t computing parties have access to. By assumption, M_I has full rank. Thus, the parties can jointly compute M_I^* as described above. For GALISS, let's just consider sharing one secret m. Note that we can compute:

$$a^m = \frac{a^c}{a^{\langle g, \lfloor \frac{M_I^* \mathbf{s}_I}{\gamma} \rceil\rangle}} = \frac{a^c}{a^{\langle \mathbf{g}, \mathbf{r}\rangle}}$$

There is one problem now, namely that we have approximate linearity and not linearity anymore, i.e., we cannot find factors α_i s.t. $\langle g, \lfloor \frac{M_I^* s_I}{\gamma} \rceil\rangle = \sum_{i=1}^t \alpha_i s_i$. A remedy would be choosing a bigger γ:

$$\begin{aligned}\mathbf{g}^T M_I^* \mathbf{s}_I &= \mathbf{g}^T([\gamma \cdot M_I^{-1} + \Delta] M_I \mathbf{r}) \\ &= \mathbf{g}^T(\gamma \cdot M_I^{-1} M_I \mathbf{r} + \Delta M_I \mathbf{r}) \\ &= \gamma \mathbf{g}^T \cdot \mathbf{r} + \mathbf{g}^T \Delta M_I \mathbf{r}\end{aligned}$$

Choose γ such that this equality holds:

$$\|\mathbf{g}^T \Delta M_I \mathbf{r}\| < \frac{\gamma}{2} \Rightarrow \langle \mathbf{g}, \mathbf{r} \rangle = \lfloor \frac{\langle \mathbf{g}, M_I^* \mathbf{s}_I \rangle}{\gamma} \rceil$$

The problem boils down to solving:

$$a^{\lfloor \frac{\langle \mathbf{g}, M_I^* \mathbf{s}_I \rangle}{\gamma} \rceil} = a^{\langle \mathbf{g}, M_I^* \mathbf{s}_I \rangle \cdot \frac{1}{\gamma} + \delta} = \left(a^{\langle \mathbf{g}, M_I^* \mathbf{s}_I \rangle} \right)^{\frac{1}{\gamma}} \cdot a^{\delta}$$

for some $\delta \in [-\frac{1}{2}, \frac{1}{2}]$. Hence, $a^{\delta} \in [\frac{1}{\sqrt{a}}, \sqrt{a}]$. The parties could compute $\left(a^{\langle \mathbf{g}, M_I^* \mathbf{s}_I \rangle} \right)^{\frac{1}{\gamma}}$ similarly to the continuous case, but now we have a multiplicative overhead that we cannot get rid off by rounding.

6.2 RSA & co.

Distributed exponentiation is used in different protocols, like RSA, similarly to what [DT06] [Tho09] did. *In general*, linear integer secret sharing is a good choice for protocols where the modulus might not be prime or publicly known, as we analyzed in the introduction.

RSA and other protocols that depend on exponentiation are defined over cyclic groups in which specific properties hold. For RSA, we have that for a so-called generator a of the group $\mathbb{Z}_N^*$, the function $f(s) = a^s \mod N$ is bijective. Furthermore, it is conjectured to be exponentially hard to compute s from $f(s)$. If the so-called "RSA-assumption" holds, the attacker of our distributed exponentiation cannot obtain the shares of the parties that post their $a_{i_j} = a^{s_{i_j}}$'s, i.e., the attacker cannot obtain s_{i_j} from a_{i_j}.

However, GALISS does not seem to be an appropriate choice for RSA / distributed exponentiation, since there is a multiplicative overhead as seen above.

6.3 Multi-Party Threshold Cryptography

There has been a recent interest in multi-party threshold cryptography. The distributed exponentiation protocol is an example of threshold cryptography since it enables multiple parties to perform operations like encryption, decryption, or signing jointly. Secret sharing is used to split up secret information like private keys and distribute those shares amongst parties. A set of t (this is the threshold) parties can perform the operation while $t-1$ parties can not. Threshold cryptography finds application in cloud computing, secure financial transactions, and other sensitive multi-party protocols [Cac23]. Because of increased relevance, a NIST call has been opened [1]. NIST is short for "National Institute of Standards and Technology," an American authority that de-facto sets the standards of which protocols are used in practice.

Now, we will discuss threshold signature schemes (TSS) in further detail. They are of practical interest, for example, for consensus protocols in blockchain [Lee23]
For those unfamiliar with message signatures, a signature is a cryptographic tool to maintain the integrity of messages. When a sender sends a message to some receiver, a signature that ensures the integrity of the message is attached. Signatures are constructed so that it is exponentially hard for a third party to come up with some alternative message and its corresponding signature, i.e., the signature is only forgable with negligible probability.
In a (t, n) setup, a TSS would look similar to this:

[1] https://csrc.nist.gov/projects/threshold-cryptography

1. The dealer creates a verification key vk and a signing key sk.

2. While vk is public, the signing key is secret-shared amongst the participating n parties.

3. t parties can then come up with partial signatures that can be combined into a valid signature s for a message m. Parties that receive the message can then verify the correctness of the signature with the vk.

4. A set of $t' < t$ unqualified parties are, on the other hand, not able to forge a signature s' for some message $m' \neq m$

There is one exciting application of GALISS, which we will be working on beyond the scope of this thesis.

There is an interest in coming up with a lattice-based threshold signature scheme. Compared to protocols that depend on classical hardness assumptions, lattice-based problems like the Shortest Vector Problem (GapSVP) [BLP+13] are proven to be secure under quantum attackers. RSA-Signature protocols, for example, depend on a classical hardness assumption, namely that factoring large prime numbers is hard on ordinary computers. However, it has been proven that factoring large numbers is easy for quantum attackers (Shor's algorithm).

Why can GALISS or some variation of the schemes be interesting in improving lattice-based threshold schemes? On the one hand, if one chooses finite group secret sharing techniques like Shamir's scheme, one has to choose the modulus to be very high. This is because the Lagrange coefficients in the reconstruction might blow up the error noise [BS23][BGG+18]. On the other hand, other integer secret sharing techniques end up with big share sizes [BS23]. For those linear schemes, the share size is in $\Omega(n)$. We believe that this share size can be decreased in favor of GALISS being only *approximate* linear and want to analyze this further in future work.

7 Conclusion

In this thesis, we presented the two secret sharing techniques GLRSS and GALISS.

We discussed secret sharing and provided a formal definition of (t, n)-secret sharing. We learned that there are different attacker models and types of secret sharing schemes that can be versatile (e.g. [Tho09]) but also domain-specific (e.g. DNA-based secret sharing).

This work motivated integer secret sharing and drew applications like distributed exponentiation, threshold crypto, and multi-party computation on the integers.

After providing the reader with necessary preliminaries on cryptography, lattices, spectral analysis, and Gaussians, we stated the distribution and reconstruction algorithms of GLRSS. GLRSS is a preliminary stage of the integer case and demonstrates how we can use Gaussian distributions to hide secrets in the smooth continuous case. As with all secret sharing schemes, we proved the two most important properties of a secret sharing scheme: t parties can reconstruct the secret easily. In contrast, $t - 1$ parties learn negligible information about the secret.

Then we delved into the integer case and saw that this scheme cannot be linear but approximately linear. However, we saw that this scheme is still correct. The security of GALISS has to be discussed in future work.

Section 5 discussed interesting properties of the schemes, compared them to existing protocols, and talked about the limitations of the scheme.

In 6, we discussed applications of GLRSS and GALISS. By the example of distributed exponentiation, we saw the upsides of a linear scheme and the drawbacks of an approximate linear scheme.

Approximate linearity does not necessarily prevent distributed protocols from working, but we saw cases in which approximate linearity causes problems. This happened because distributed exponentiation requires linearity in the exponent, which caused us to have a multiplicative error that we cannot overcome by rounding.

Conclusively, we presented novel secret sharing schemes over $\mathbb{R}$ and $\mathbb{Z}$ that are, to our knowledge, the first linear integer secret sharing schemes over those structures that use Gaussian distributioned shares. We expect that this work helps to improve parameters of future cryptographic protocols.

References

[AB83] C. Asmuth and J. Bloom. A modular approach to key safeguarding. *IEEE Transactions on Information Theory*, 29(2):208–210, 1983.

[add] Summary of lecture on secret sharing. `https://www.cs.columbia.edu/~tal/4261/F19/secretsharingf19.pdf`. Accessed: 2024-02-29.

[Adh06] A. Adhikari. Dna secret sharing. In *2006 IEEE International Conference on Evolutionary Computation*, pages 1407–1411, 2006.

[AGHS13] Shweta Agrawal, Craig Gentry, Shai Halevi, and Amit Sahai. Discrete gaussian leftover hash lemma over infinite domains. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT (1)*, volume 8269 of *Lecture Notes in Computer Science*, pages 97–116. Springer, 2013.

[Bei11] Amos Beimel. Secret-sharing schemes: A survey. In Yeow Meng Chee, Zhenbo Guo, San Ling, Fengjing Shao, Yuansheng Tang, Huaxiong Wang, and Chaoping Xing, editors, *Coding and Cryptology*, pages 11–46, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[Bel97] Mihir Bellare. A note on negligible functions. Cryptology ePrint Archive, Paper 1997/004, 1997. `https://eprint.iacr.org/1997/004`.

[BGG+18] Dan Boneh, Rosario Gennaro, Steven Goldfeder, Aayush Jain, Sam Kim, Peter M. R. Rasmussen, and Amit Sahai. Threshold cryptosystems from threshold fully homomorphic encryption. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018*, pages 565–596, Cham, 2018. Springer International Publishing.

[BI05] Amos Beimel and Yuval Ishai. On the power of nonlinear secret-sharing. *SIAM Journal on Discrete Mathematics*, 19(1):258–280, 2005.

[BLP+13] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors, 2013.

[BS23] Katharina Boudgoust and Peter Scholl. Simple threshold (fully homomorphic) encryption from lwe with polynomial modulus. Cryptology ePrint Archive, Paper 2023/016, 2023. `https://eprint.iacr.org/2023/016`.

[Cac23] Christian Cachin. *Multi-Party Threshold Cryptography*, pages 65–69. Springer Nature Switzerland, Cham, 2023.

[CDLP14] Kai-Min Chung, Daniel Dadush, Feng-Hao Liu, and Chris Peikert. On the lattice smoothing parameter problem, 2014.

[CK93] Benny Chor and Eyal Kushilevitz. Secret sharing over infinite domains. *Journal of Cryptology*, 6:87–95, 1993.

[CSNN24] Arup Kumar Chattopadhyay, Sanchita Saha, Amitava Nag, and Sukumar Nandi. Secret sharing: A comprehensive survey, taxonomy and applications. *Computer Science Review*, 51:100608, 2024.

[DH76] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[DKL+23] Nico Döttling, Dimitris Kolonelos, Russell W. F. Lai, Chuanwei Lin, Giulio Malavolta, and Ahmadreza Rahimi. Efficient laconic cryptography from learning with errors. Cryptology ePrint Archive, Paper 2023/404, 2023. `https://eprint.iacr.org/2023/404`.

[DMR18] Luc Devroye, Abbas Mehrabian, and Tommy Reddad. The total variation distance between high-dimensional gaussians with the same mean. *arXiv preprint arXiv:1810.08693*, 2018.

[DT06] Ivan Damgard and Rune Thorbek. Linear integer secret sharing and distributed exponentiation. Cryptology ePrint Archive, Paper 2006/044, 2006. `https://eprint.iacr.org/2006/044`.

[HL] Howard Huang and Gijs Van Laer. Cs 600.442 – modern cryptography, lecture 2: One way functions. `https://www.cs.jhu.edu/~abhishek/classes/CS600-442-Fall2015/LectureS2.pdf`. Accessed: 2024-02-29.

[int] Shamir's secret sharing. `https://en.wikipedia.org/wiki/Shamir's_secret_sharing#Problem_of_using_integer_arithmetic`. Accessed: 2024-02-29.

[joi] Jointly gaussian. `https://inst.eecs.berkeley.edu/~ee126/fa22/notes/jg_note.pdf`. Accessed: 2023-12-20.

[Lau] Sven Laur. Mtat.07.003 cryptology ii; computational indistinguishability. `https://kodu.ut.ee/~swen/courses/crypto-ii/03-indistinguishability.pdf`. Accessed: 2024-02-29.

[Lee23] Kwangsu Lee. Decentralized threshold signatures for blockchains with non-interactive and transparent setup. Cryptology ePrint Archive, Paper 2023/1206, 2023. `https://eprint.iacr.org/2023/1206`.

[Mig82] Maurice Mignotte. How to share a secret? In *International Conference on the Theory and Application of Cryptographic Techniques*, 1982.

[Nie21] Zipei Nie. Matrix anti-concentration inequalities with applications, 2021.

[NMH+18] Yu Ning, Fuyou Miao, Wenchao Huang, Keju Meng, Yan Xiong, and Xingfu Wang. Constructing ideal secret sharing schemes based on chinese remainder theorem. In Thomas Peyrin and Steven Galbraith, editors, *Advances in Cryptology – ASIACRYPT 2018*, pages 310–331, Cham, 2018. Springer International Publishing.

[Pei16] Chris Peikert. A decade of lattice cryptography. *Found. Trends Theor. Comput. Sci.*, 10(4):283–424, mar 2016.

[PM14] Andrew J. Paverd and Andrew C. Martin. Modelling and automatically analysing privacy properties for honest-but-curious adversaries. 2014.

[Reg04] Oded Regev. New lattice-based cryptographic constructions. *J. ACM*, 51(6):899–942, nov 2004.

[RSA78] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, feb 1978.

[Sha79] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, nov 1979.

[Tho09] Rune Thorbek. *Linear Integer Secret Sharing*. Phd dissertation, University of Aarhus, May 2009. Available at `https://cs.au.dk/fileadmin/site_files/cs/PhD/PhD_Dissertations__pdf/Thesis-RIT.pdf`.

[TV09] Terence Tao and Van Vu. Random matrices: The distribution of the smallest singular values, 2009.