Threshold Linearly Homomorphic Encryption on $\mathbf{Z}/2^k\mathbf{Z}$

Guilhem Castagnos, Fabien Laguillaumie, Ida Tucker

Threshold Linearly Homomorphic Encryption on $\mathbf{Z}/2^k\mathbf{Z}$

Guilhem Castagnos[1], Fabien Laguillaumie[2], and Ida Tucker[3,4]

[1] Université de Bordeaux, CNRS, INRIA, IMB, UMR 5251, F-33400 Talence, France.
[2] LIRMM, Université de Montpellier, CNRS, France.
[3] IMDEA Software Institute, Madrid, Spain.
[4] Zondax AG.

Abstract. A threshold public key encryption protocol is a public key system where the private key is distributed among n different servers. It offers high security since no single server is entrusted to perform the decryption in its entirety. It is the core component of many multiparty computation protocols which involves mutually distrusting parties with common goals. It is even more useful when it is homomorphic, which means that public operations on ciphertexts translate to operations on the underlying plaintexts. In particular, Cramer, Damgård and Nielsen at Eurocrypt 2001 provided a new approach to multiparty computation from linearly homomorphic threshold encryption schemes. On the other hand, there has been recent interest in developing multiparty computations modulo 2^k for a certain integer k, that closely match data manipulated by a CPU. Multiparty computation would therefore benefit from an encryption scheme with such a message space that would support a distributed decryption.
In this work, we provide the first threshold linearly homomorphic encryption whose message space is $\mathbf{Z}/2^k\mathbf{Z}$ for any k. It is inspired by Castagnos and Laguillaumie's encryption scheme from RSA 2015, but works with a class group of discriminant whose factorisation is unknown.
Its natural structure *à la* Elgamal makes it possible to distribute the decryption among servers using linear integer secret sharing, allowing any access structure for the decryption policy. Furthermore its efficiency and its flexibility on the choice of the message space make it a good candidate for applications to multiparty computation.

Keywords: Class groups of quadratic fields, Linearly homomorphic encryption, Threshold cryptography

1 Introduction

Encryption protocols are the core of any communication architecture. They provide confidentiality, defined in terms of semantic security or indistinguishability of encryptions by Goldwasser and Micali [37]. On top of this security property, many applications require an "algebraic" property of the encryption scheme, in the sense that an operation on the ciphertexts translates into an operation

on the underlying plaintexts. An encryption protocol possessing this property is said to be *homomorphic*. While *fully* homomorphic encryption schemes allow any operation to be evaluated on ciphertexts, protocols that only allow linear transformations are also very useful and significantly more efficient.

The first linearly homomorphic encryption appears in Goldwasser and Micali's seminal work [37]. Then a line of factoring based schemes was developed, culminating with Paillier's scheme [49] which was then generalized by Damgård and Jurik [26], allowing to encrypt larger messages.

An alternative was proposed by Castagnos and Laguillaumie using class groups of quadratic fields in [17]. This allows to work with the additive group $\mathbf{Z}/q\mathbf{Z}$ as a message space where q is an odd prime, whereas Paillier and Damgård and Jurik' schemes work modulo N^s where N is an RSA integer. The case of message space $\mathbf{Z}/q^s\mathbf{Z}$ for an odd prime q, and more generally that of $\mathbf{Z}/N\mathbf{Z}$ with $N = \prod q_i^{s_i}$ for odd primes q_i, were sketched in the conclusion of [17]. This was further analyzed in [28] which gives a detailed construction and implementation. As a consequence, the Castagnos-Laguillaumie scheme allows to construct message spaces of any odd order N (with known factorization). There is a restriction however: many cryptographic applications and proofs require that this order N be relatively prime to the order of the underlying class group. This can only be ensured with high probability if each prime q_i dividing N is large enough to make $1/q_i$ negligible. Hence only relatively large values of odd integers N are possible in practice. The case of message spaces defined modulo 2^k were left open in these works.

Another elegant work, by Benhamouda, Herranz, Joye, Libert [3] (refining a scheme by Joye and Libert [41]), generalizes the Goldwasser-Micali cryptosystem using 2^k-th power residue symbols, and produces efficient protocols in terms of bandwidth and speed (for both encryption and decryption). It is proven secure under the quadratic residuosity assumption for RSA moduli $N = pq$, where the primes p and q have a special form. The message space of their scheme is the additive group $\mathbf{Z}/2^k\mathbf{Z}$, which is a very interesting feature, especially for the purpose of multi-party computation. Indeed, it has been used by Catalano, Di Raimondo, Fiore and Giacomelli [19] to design a new $2-$party protocol for secure computation over the ring $\mathbf{Z}/2^k\mathbf{Z}$. Their work follows a new line of secure computation modulo 2^k, initiated by Cramer, Damgård, Escudero, Scholl, and Xing in [23] who introduced a new information theoretic MAC that allows to authenticate messages in the ring $\mathbf{Z}/2^k\mathbf{Z}$ to achieve security against malicious adversaries. This choice is driven by the fact that modern CPU computations are performed in such a ring, and it allows protocol designers to directly apply optimizations that are often expensive to emulate modulo p or N.

On the other hand, several multi-party computation protocols, starting with the pioneering work of Cramer, Damgård and Nielsen [24], rely on threshold linearly homomorphic encryption. A $(t, n)-$threshold public key encryption (TPKE) scheme allows n parties to share the decryption key so that if t of them collaborate, they can decrypt ciphertexts, whereas $t - 1$ users learn nothing about the underlying plaintext. Katz and Yung proposed a threshold variant

of Goldwasser-Micali in [42], but this does not extend to a message space of order 2^k. Furthermore, it is an open problem to devise an efficient threshold variant of Benhamouda, Herranz, Joye, Libert's scheme. In a nutshell, This scheme uses an RSA integer $N = pq$ where $p \equiv 1 \pmod{2^k}$. A ciphertext for $m \in \mathbf{Z}/2^k\mathbf{Z}$ is $c = y^m x^{2^k} \in \mathbf{Z}/N\mathbf{Z}$ for a random x, and a public y, which is a fixed non quadratic residue, with Jacobi Symbol 1. Decryption is done modulo p, by removing the 2^k-th power using an exponentiation to the power $(p-1)/2^k$ and then finding m thanks to an easy discrete logarithm computation using Pohlig-Hellman's algorithm in the subgroup of $(\mathbf{Z}/p\mathbf{Z})^\times$ of order 2^k. As a result, lots of operations are done modulo the secret prime p, which prevents an efficient adaptation in a multiparty setting.

A solution would be to design an Elgamal version of this scheme, that fits the CL framework [17] of a DDH group with an easy DL subgroup. In this framework, one works with a cyclic group G isomorphic to $H \times F$ where H and F are subgroups of G of respective unknown order s and known order q, with q and s co-prime. The group H consists of $q-$th powers, and in F discrete logarithms are easy to compute. This makes it possible to encode messages $m \in \mathbf{Z}/q\mathbf{Z}$ in f^m where f is a generator of F. Then f^m is hidden by a random $q-$th power.

To make Benhamouda et al's scheme fit the CL framework, the idea would be to use two primes $p, q \equiv 1 \pmod{2^k}$ and encode the message in the exponent of an element $f \in (\mathbf{Z}/N\mathbf{Z})^\times$ of order 2^k both modulo p and q. A ciphertext for m could then be of the form $(g^r, f^m \mathsf{pk}^r)$, with $\mathsf{pk} = g^{\mathsf{sk}}$. During decryption, after recovering f^m, the discrete logarithm computation could then be done modulo the public N, and only one exponentiation would have to be distributed among the parties. However, this simple solution has some drawbacks due to the fact that this element f must be public and seems hard to generate without knowing the factorization. As a consequence, such a variant would rely on *ad hoc* security assumptions that include this element f. Moreover, it would be less efficient than the scheme that we propose in this work, at least in terms of bandwidth. Devising a variant without a trusted dealer would also be very complicated. So this question remains open:

Is it possible to design an efficient threshold linearly homomorphic encryption with message space $\mathbf{Z}/2^k\mathbf{Z}$?

Our contributions. In this work, we first propose a new linearly homomorphic encryption (LHE) scheme with message space of order 2^k that solves the aforementioned issues (Section 4). This LHE has an Elgamal structure as it follows the CL framework, with an element f of order 2^k that is used to encrypt the messages in the exponent. We emphasize that this element can be generated from public parameters. Thanks to its Elgamal shape, it can be converted into the first threshold linearly homomorphic encryption with message space $\mathbf{Z}/2^k\mathbf{Z}$.

The part of the decryption which involves the secret key uses an exponentiation to that secret key in a group of unknown order. We use linear integer secret sharing schemes (LISS), introduced in [27], to share the secret key over

the integers. This allows to set up a scheme allowing any access structure for the decryption policy and in particular a threshold decryption (Section 5).

Furthermore, we suggest how to add robustness and a distributed setup to our scheme. We also sketch several application domains: multiparty computation, homomorphic secret sharing and lossy trapdoor functions (Section 6).

The security of our schemes relies on a hard subgroup membership (HSM) assumption, which is a natural adaptation of the assumption used in the CL framework. One could also design a variant based on the DDH assumption.

The setup of our schemes is flexible and allows to encrypt messages modulo 2^k for any integer k, and in particular natural choices such as $k = 1, 32, 64, 128$ or even larger values. We show that our schemes are efficient by reporting timings from an implementation in SageMath.

Technical Overview and Challenges. Our construction uses class groups of imaginary quadratic fields like the encryption schemes modulo an odd prime q of [17,55]. One of the challenges is to stay within the CL framework, while working modulo a power of 2. As we will see, plugging $q = 2^k$ does not work.

In the original framework, a class group with a cyclic subgroup of order q is generated. This is done by considering two class groups, $Cl(\Delta_K)$ with discriminant $\Delta_K = -pq$ and $Cl(\Delta)$ with discriminant $-pq^3$. In this case, there is a surjection from $Cl(\Delta)$ to $Cl(\Delta_K)$, and the kernel of which is precisely the required subgroup of order q.

However, as usual in number theory, moving from an odd prime q to 2 or 2^k is not an easy task. Firstly, setting $q = 2^k$ in the construction above does not always give a cyclic subgroup of order 2^k. Further difficulties arise from the fact that in class groups, knowing the factorization of the discriminant allows to compute square roots, and decide if elements are squares. And as we will see, this allows to completely break a scheme which uses an Elgamal in the exponent with a subgroup of order 2^k.

We solve this issue by constructing a discriminant from an RSA integer N of unknown factorization. But other technical reasons make the choice of this discriminant tricky, and lead to arithmetic conditions on the primes composing N (which have no negative impact in practice). We thus have to delve into the genus theory associated to class groups of quadratic fields, introduced by Gauss, to select discriminants that make it possible to securely work with the group of squares of cardinality $2^k s$ where s is odd (Section 3). Indeed, we need to carefully handle the fact that some genera can leak information on discrete logarithms. Controlling the 2-Sylow subgroup of $Cl(\Delta_K)$ and expliciting the shape of the kernel of the surjection from $Cl(\Delta)$ to $Cl(\Delta_K)$ when the conductor is equal to 2^k allows to find a element f of the group of squares of order 2^k which does not depend on the factorization of N.

Relying on the factorization assumption implies slightly larger elements (compared to the original CL scheme), but the timings that we provide in Table 2 of Section 4.3 from a non-optimized implementation of our protocol with SageMath, show that the scheme is actually very efficient.

2 Background

2.1 Threshold public key encryption

In a threshold PKE (TPKE) scheme, the decryption key is divided into a number of key shares which are distributed to multiple decryption servers, according to a certain access structure. To decrypt a message, each server creates its own decryption share, and these shares can be publicly combined to result in a full decryption.

Definition 1. *A monotone access structure on $\{1,\ldots,n\}$ is a non-empty collection $\mathbb{A} \subseteq \{1,\ldots,n\}$ such that $\emptyset \notin \mathbb{A}$, and such that for all $A \in \mathbb{A}$, and all sets B such that $A \subseteq B \subseteq \{1,\ldots,n\}$ it holds that $B \in \mathbb{A}$.*
For a positive integer $t < n$, the threshold-t access structure $T_{t,n}$ is the collection of sets $A \subseteq [n]$ for which $|A| > t$. The sets in $\mathbb{A}$ are called qualified*, whereas the sets outside A which should not be able to obtain any information about the secret are called* forbidden*.*

Definition 2 (Threshold PKE). *Let $P = \{P_1,\ldots,P_n\}$ be a set of parties and let $\mathbb{A}$ be an access structure. A threshold PKE scheme for a message space $\mathcal{M}$ and access structure $\mathbb{A}$ is a tuple of PPT algorithms $\mathsf{TPKE} = (\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{PartDec}, \mathsf{FinalDec})$ with the following syntax.*

$\mathsf{Setup}(1^\lambda, \mathbb{A}) \to (\mathsf{pp}, \mathsf{ek}, \mathbf{sk})$ *Takes as input a security parameter 1^λ and an access structure $\mathbb{A}$. It outputs public parameters pp, an encryption key ek, and a vector of n secret-key shares $\mathbf{sk} = (\mathsf{sk}_1,\ldots,\mathsf{sk}_n)$.*
Party P_i is given the share sk_i that allows deriving decryption shares for any ciphertext.

$\mathsf{Encrypt}(\mathsf{ek}, m) \to c$ *On input the encryption key ek and a plaintext $m \in \mathcal{M}$, outputs a ciphertext ct.*

$\mathsf{PartDec}(\mathsf{pp}, \mathsf{sk}_i, \mathsf{ct}) \to \mu_i \cup \perp$ *Takes as input the public parameters pp, a secret-key share sk_i, and a ciphertext ct. It outputs a partial decryption share μ_i.*

$\mathsf{FinalDec}(\mathsf{pp}, \{\mu_i\}_{i\in S}) \to m \cup \perp$ *Given pp and a subset $S \subset \{1,\ldots,n\}$ with decryption shares $\{\mu_i\}_{i\in S}$, this algorithm outputs either a plaintext m or $\perp$.*

We require a TPKE scheme to satisfy the following correctness, and security requirements.

Definition 3 (Decryption correctness). *We say that a TPKE scheme for an access structure $\mathbb{A}$ satisfies decryption correctness if for all λ, and all qualified sets S, the following holds. For $(\mathsf{pp}, \mathsf{ek}, \mathbf{sk}) \leftarrow \mathsf{Setup}(1^\lambda, \mathbb{A})$, $\mathsf{ct} \leftarrow \mathsf{Encrypt}(\mathsf{ek}, m)$, $\mu_i \leftarrow \mathsf{PartDec}(\mathsf{pp}, \mathsf{sk}_i, \mathsf{ct})$ for $i \in S$, $\Pr[\mathsf{FinalDec}(\mathsf{pp}, \{\mu_i\}_{i\in S}) = m] = 1 - \mathsf{negl}(\lambda)$.*

Definition 4 (following [34]) is a classical extension of semantic security for an encryption scheme to the threshold case. The attacker actively (but non-adaptively) corrupts a set S of servers outside $\mathbb{A}$, gets their secret keys, and can ask for partial decryptions of ciphertexts for which he already knows the corresponding plaintext. The idea is that partial decryptions give no information about the private keys of non-corrupted users.

Definition 4 (T-ind-cpa-security). *We say a* TPKE *scheme for an access structure* $\mathbb{A}$ *is adaptive chosen plaintext (*T-ind-cpa*) secure if for any large enough* $\lambda \in \mathbf{N}$*, and any PPT adversary* $\mathcal{A}$ *the experiments* $\mathsf{Expt}_{\mathcal{A},\mathsf{TPKE},0}$ *and* $\mathsf{Expt}_{\mathcal{A},\mathsf{TPKE},1}$ *of Fig. 1 are computationally indistinguishable.*

Choose structure: On input 1^λ, the adversary $\mathcal{A}$ chooses an access structure $\mathbb{A}$.
Set up: The challenger $\mathcal{C}$ runs $(\mathsf{pp}, \mathsf{ek}, \mathbf{sk}) \leftarrow \mathsf{Setup}(1^\lambda, \mathbb{A})$ and sends pp, ek to $\mathcal{A}$.
Choose set: $\mathcal{A}$ outputs a set S such that $S \notin \mathbb{A}$, and $\mathcal{C}$ sends the set of secret keys $\{\mathsf{sk}_i\}_{i \in S}$ to $\mathcal{A}$.
Partial decryption queries: $\mathcal{A}$ queries partial decryption on encryptions of plaintexts m_j of his choice. For $\mathsf{ct}_j = \mathsf{Encrypt}(\mathsf{pp}, \mathsf{ek}, m_j)$, it then gets $\mathsf{PartDec}(\mathsf{pp}, \mathsf{sk}_i, \mathsf{ct}_j)$ for $i \notin S$
Choose challenge: $\mathcal{A}$ outputs a pair of challenge messages $m_0, m_1 \in \mathcal{M}$.
Challenge: $\mathcal{C}$ computes $\mathsf{ct}_b \leftarrow \mathsf{Encrypt}(\mathsf{ek}, m_b)$ and sends ct_b to $\mathcal{A}$.
Partial decryption queries: Again, $\mathcal{A}$ queries partial decryption on encryptions of plaintexts m_j of his choice : it gets $\mathsf{ct}_j = \mathsf{Encrypt}(\mathsf{pp}, \mathsf{ek}, m_j)$ and $\mathsf{PartDec}(\mathsf{pp}, \mathsf{sk}_i, \mathsf{ct}_j)$ for $i \notin S$
Guess: $\mathcal{A}$ outputs a bit b', which is the output of the experiment.

Fig. 1. Experiment $\mathsf{Expt}_{\mathcal{A},\mathsf{TPKE},\mathsf{b}}$

Linearly Homomorphic Threshold Encryption. This primitive is particularly useful for applications to multi-party computation.

Definition 5. *Consider a* TPKE *with message space* $(\mathcal{M}, +)$*. A linearly homomorphic* TPKE *scheme additionally has the following* evaluation *algorithms:*

$\mathsf{EvalAdd}(\mathsf{pp}, c_1, c_2) \to c^*$ *Takes as input* pp *and two ciphertexts* c_1 *and* c_2*, and outputs a new ciphertext* c^* *which decrypts to* $m_1 + m_2$ *where each* c_i*,* $i \in \{1, 2\}$ *decrypts to* m_i*.*

$\mathsf{EvalScal}(\mathsf{pp}, c, \alpha) \to c^*$ *Takes as input* pp*, a ciphertexts* c *which decrypts to* m*, and a scalar* α*, and outputs a new ciphertext* c^* *which decrypts to* $\alpha \cdot m$*.*

Informally, evaluations should be correct, meaning that decryption should lead to the correct plaintext message $m_1 + m_2$ *(resp.* αm*).*

2.2 Linear integer secret sharing

In the threshold setting for groups of unknown orders, key generation schemes share the secret decryption key using the linear integer secret sharing (LISS) primitive of Damgård and Thorbek [27], which is similar to linear secret sharing schemes except that it works over $\mathbf{Z}$.

They show that any integer span program (ISP) as defined in [25] can be used to build a secure LISS scheme. Roughly speaking, an ISP is specified by a

matrix with integer entries, and these entries are used as coefficients in the linear combinations that produce the shares from secret and randomness. They also show that any LISS scheme can be used to build a distributed exponentiation protocol, which is what we will use in our threshold decryption.

Goal. Let $P = \{1, \ldots, n\}$ denote the n shareholders and D the dealer. Let $\mathbb{A}$ be a monotone access structure on P. The dealer D wants to share a secret s in a publicly known interval $[-2^l, 2^l]$ with the shareholders, such that any set $A \in \mathbb{A}$ can reconstruct s, but any set $A \notin \mathbb{A}$ gets no (or negligible) information on s.

Distributing the secret. To this end, D uses a distribution matrix $\mathbf{M} \in \mathbf{Z}^{d \times e}$ and a distribution vector $\boldsymbol{\rho} = (s, \rho_2, \ldots, \rho_e)^\top$, where s is the secret, and the ρ_i's are integers sampled uniformly at random in $[-2^{l_0+\lambda}, 2^{l_0+\lambda}]$ for $2 \leq i \leq e$, where l_0 is a constant that is part of the description of the scheme.

The dealer D computes a vector $\mathbf{s} \in \mathbf{Z}^d$ of share units as:

$$\mathbf{s} = (s_1, \ldots, s_d)^\top = \mathbf{M} \cdot \boldsymbol{\rho}.$$

Let $\psi : \{1, \ldots, d\} \mapsto P$ be a surjective function. Shareholder $\psi(i)$ is given the i-th share unit, and is said to *own* the i-th row in $\mathbf{M}$. For a set of shareholders $A \subset P$, $\mathbf{M}_A \in \mathbf{Z}^{d_a \times e}$ denotes the restriction of $\mathbf{M}$ to the rows jointly owned by A, while d_A denotes the number of these rows.

Likewise, $s_A \in \mathbf{Z}^{d_A}$ denotes the restriction of $\mathbf{s} \in \mathbf{Z}^d$ to the coordinates jointly owned by the parties in A. Shareholder j's share consists of $s_{\psi^{-1}(j)} \in \mathbf{Z}^{d_j}$, so that it receives $d_j = |\psi^{-1}(j)|$ out of the $d = \sum_{j=1}^{n} d_j$ share units. The *expansion rate* $\mu = d/n$ is the average number of share units per player.

To construct LISS schemes, Damgård and Thorbek [27] used integer span programs [25], which were originally used to construct black-box secret sharing which does not extend shares in the ring of integers.

Definition 6 (Integer Span Program (ISP) [25]). *The tuple $\mathcal{M} = (\mathbf{M}, \psi, \boldsymbol{\epsilon})$ is called an* integer span program *(ISP), if $\mathbf{M} \in \mathbf{Z}^{d \times e}$ and the d rows of $\mathbf{M}$ are labeled by a surjective function $\psi : \{1, \ldots, d\} \mapsto \{1, \ldots, n\}$. Finally, $\boldsymbol{\epsilon} = (1, 0, \ldots, 0)^\top \in \mathbf{Z}^e$ is called the target vector. The size of $\mathcal{M}$ is the number of rows d of $\mathbf{M}$.*

Definition 7. *Let $\mathbb{A}$ be a monotone access structure and let $\mathcal{M} = (\mathbf{M}, \psi, \boldsymbol{\epsilon})$ be an ISP. Then $\mathcal{M}$ is an ISP for $\mathbb{A}$ if for all $A \subset \{1, \ldots, n\}$ the following conditions hold:*

- *If $A \in \mathbb{A}$, there is a reconstruction vector $\boldsymbol{\lambda} \in \mathbf{Z}^{d_A}$ such that $\boldsymbol{\lambda}^\top \cdot \mathbf{M} = \boldsymbol{\epsilon}^\top$.*
- *If $A \notin \mathbb{A}$, there exists $\boldsymbol{\kappa} = (\kappa_1, \ldots, \kappa_e)^\top \in \mathbf{Z}^e$ such that $\mathbf{M}_A \cdot \boldsymbol{\kappa} = \mathbf{0} \in \mathbf{Z}^d$, and $\boldsymbol{\kappa}^\top \cdot \boldsymbol{\epsilon} = 1$. The vector $\boldsymbol{\kappa}$ is called a* sweeping vector for A.
  We also define $\kappa_{\max} = \max_{A \notin \mathbb{A}}(\|\kappa\|_\infty)$.

In other words, the rows owned by a qualified set must include the target vector in their span, while for a forbidden set, there must exist a sweeping vector

which is orthogonal to all rows of the set, but has inner product 1 with the target vector. We also say that $\mathcal{M}$ *computes* $\mathbb{A}$.

Damgård and Thorbek [27] showed that from an ISP $\mathcal{M} = (\mathbf{M}, \psi, \boldsymbol{\epsilon})$ which computes the access structure $\mathbb{A}$, a statistically private LISS scheme for $\mathbb{A}$ can be obtained with $\mathbf{M}$ as the share generating matrix and $l_0 = l + \lceil \log_2(\kappa_{max}(e-1)) \rceil + 1$, where l is the length of the secret.

Then LISS can be obtained from Cramer-Fehr [25] or Benaloh-Leichter [2]. Although this later case was designed to work over finite groups, Damgård and Thorbek generalized it to share integers using access structures consisting of any monotone Boolean formula. Thanks to results of Valiant [56], LISS schemes can therefore be constructed for any threshold access structure. From a monotone Boolean function f, Damgård and Thorbek's technique from Benaloh-Leichter results allows binary share distribution matrices in $\{0,1\}^{d \times e}$ such that $d, e = O(\mathsf{size}(f))$ and which have at most $\mathsf{depth}(f) + 1$ non-zero entries, so that each share unit has magnitude $O(2^{l_0+\lambda}\mathsf{depth}(f))$. Valiant's results, improved by [38] gives a monotone formula of size $O(n^{1+\sqrt{2}})$ and depth $O(\log n)$ for the majority function (from which any threshold-t function can be built). This reduces the average share size to $O(n^{\sqrt{2}}(l_0 + \lambda + \log\log(n)))$ bits.

Lemma 1 ([53, Lemma 3.1]). *Let $l_0 = l + \lceil \log_2(\kappa_{max}(e-1)) \rceil + 1$. Consider a secret to be shared, $s \in [-2^l, 2^l]$, and $\boldsymbol{\rho}$ randomly sampled from $[-2^{l_0+\lambda}, 2^{l_0+\lambda}]^e$ conditionally on $\langle \boldsymbol{\rho}, \boldsymbol{\epsilon} \rangle = s$, then the LISS scheme derived from $\mathcal{M}$ is private. For any arbitrary $s, s' \in [-2^l, 2^l]$ and any forbidden set of shareholders $A \in [n]$, the two distributions $\{s_A = \mathbf{M}_A \cdot \boldsymbol{\rho} \mid \boldsymbol{\rho} \leftarrow U([-2^{l_0+\lambda}, 2^{l_0+\lambda}]^e)$ s.t. $\langle \boldsymbol{\rho}, \boldsymbol{\epsilon} \rangle = s\}$, and $\{s'_A = \mathbf{M}_A \cdot \boldsymbol{\rho} \mid \boldsymbol{\rho} \leftarrow U([-2^{l_0+\lambda}, 2^{l_0+\lambda}]^e)$ s.t. $\langle \boldsymbol{\rho}, \boldsymbol{\epsilon} \rangle = s'\}$ are $2^{-\lambda}$ close.*

2.3 Class Groups

Class Groups. Given a non square integer $\Delta < 0$, $\Delta \equiv 0, 1 \pmod 4$, called discriminant, the imaginary quadratic order of discriminant Δ, denoted $\mathcal{O}_\Delta$ is the ring $\mathbf{Z}[(\Delta + \sqrt{\Delta})/2]$. The associated class group $Cl(\Delta)$ is defined as the quotient of the group of invertible fractional ideals of $\mathcal{O}_\Delta$ by the subgroup of principal ideals. Precise definitions of these objects can be found in e.g., [11].

In a nutshell, the class group $Cl(\Delta)$ is a finite Abelian group, with an efficiently computable group law and a compact representation of elements. Elements are classes of ideals, with a unique reduced representative. The order of $Cl(\Delta)$, the class number, denoted $h(\Delta)$ is close to $\sqrt{|\Delta|}$.

Historically, with the works of Lagrange and Gauss, the class group $Cl(\Delta)$ was defined using the language of positive definite binary quadratic forms of discriminant Δ. Let $a, b, c \in \mathbf{Z}$ such that $a > 0$ and $\Delta = b^2 - 4ac$, we will denote for short $f := (a, b, c)$ the positive definite binary quadratic form over the integers, $f(X, Y) = aX^2 + bXY + cY^2$. Such a form is said to be primitive if a, b and c are relatively prime. In the following, we will just call "forms" the primitive positive definite binary quadratic forms over the integers. Two forms f and g are said to equivalent if $g(X, Y) = f(AX + BY, CX + DY)$ for integers A, B, C, D such that $AD - BC = 1$.

The class group $Cl(\Delta)$ is isomorphic to the set of forms modulo this equivalence relation. In fact, it is more natural to work with forms for algorithmic purposes: the class of the form (a, b, c) corresponds to the class of the $\mathcal{O}_\Delta-$ideal $a\mathbf{Z} + \frac{-b+\sqrt{\Delta}}{2}\mathbf{Z}$. Moreover, the definition of the unique representative of the class is more natural when working with forms: it is the reduced form (a, b, c), which satisfies $-a < b \leqslant a$, $a \leqslant c$ and if $a = c$ then $b \geqslant 0$. A reduced form satisfies $a \leqslant \sqrt{|\Delta|/3}$. As a result, elements of $Cl(\Delta)$ can be represented by (a, b) using $\log_2(|\Delta|)$ bits. Dobson *et al.* recently proposed in [32] an elegant method to reduce this representation to $3/4 \log_2(|\Delta|)$ bits.

Computations in $Cl(\Delta)$ can be performed with a reduction algorithm for forms, devised by Lagrange (which corresponds to lattice reduction in dimension 2), and Gauss' composition of forms (which corresponds to product of ideals). More recently, efficient algorithms have been proposed for practical implementation by Shanks (cf. [40]). The neutral element of $Cl(\Delta)$ is the class of the (reduced) principal form: $(1, b, (b - \Delta)/4)$ where $b = \Delta \mod 2$.

Class Group-based Cryptography. Class group cryptography dates back to the late 80s with the first key exchange in the class group of ideals of maximal orders of imaginary quadratic fields, and related protocols that can be found in Buchmann and Williams' work [12] or McCurley's [46]. After several years, a family of class group based cryptosystems, NICE, was designed using class groups of non-maximal orders [51]. The area remained dormant for another decade until a serious cryptanalysis of this whole family of NICE cryptosystems was proposed [16]. Since then, there has actually been a high regain of interest in class groups to design new advanced cryptosystems, especially for secure multi-party computation. Built upon Castagnos and Laguillaumie's linearly homomorphic encryption scheme (CL) [17], projective hash functions relying on class groups allowed to design efficient inner product functional encryption [18], 2-party and fully-threshold ECDSA signatures [14,15,58,29].

The main advantage of class-group cryptography is that it is well-suited when multi-party protocols require a one-time transparent (or public-coin) setup with minimal interaction among parties. For instance, [54] presented a scalable distributed randomness generator with enhanced security and transparent setup that relies on a variant of the CL encryption scheme. The verifiable random functions from [57] take advantage of an exponentiation in a group of unknown order without trusted setup, as well as accumulators in [45], and succinct non-interactive arguments of knowledge in [13,44].

Another advantage is that the underlying algorithmic problems are harder that equivalent problems in $(\mathbf{Z}/N\mathbf{Z})^\times$ or $(\mathbf{Z}/p\mathbf{Z})^\times$. Indeed, the current best known algorithms to solve the discrete logarithm problem in the class group of ideals of order of imaginary quadratic fields, or to compute the class number have a sub-exponential complexity of complexity $L_{|\Delta|}(1/2, o(1))$ (cf. [4]). This means that elements in the class group are asymptotically *smaller*, and this actually matters in practice for a given security parameter. For example, a 112-bit (resp. 256-bit) security determinant will be of size 1348 bits (resp. 5971 bits), while an RSA modulus will be of size 2048 bits (resp. 15360 bits).

Genus Theory, Squares and Square Roots. We now give a quick introduction on genus theory and properties of squares of the class group. A comprehensive exposition of the subject with a historical perspective can be found in [22]. The theory of quadratic forms was originally motivated by the representation problem: given $m \in \mathbf{Z}$ and a quadratic form f, are there integers $(x, y) \in \mathbf{Z}^2$ such that $f(x, y) = m$? A first remark is that all the forms in an equivalent class represent the same numbers. Genus theory aims at characterizing primes represented by quadratic forms of a fixed discriminant Δ. A genus consists of classes of forms that represent the same classes of numbers in $(\mathbf{Z}/\Delta\mathbf{Z})^\times$. Genera are related to squares of the class group: Gauss proved that the genus of the principal form, the principal genus, corresponds to the subgroup of squares in the class group $Cl(\Delta)$. Moreover, genera can be identified by values of some characters.

Before going into more details, let us make a parallel with the well-known properties of squares in $(\mathbf{Z}/N\mathbf{Z})^\times$ where $N = pq$ is an RSA integer. Given $x \in (\mathbf{Z}/N\mathbf{Z})^\times$, there are 4 possible values for the Legendre symbols $((x/p), (x/q))$ which gives a partition in 4 sets of $(\mathbf{Z}/N\mathbf{Z})^\times$. One could speak of 4 "genera". The value $(1, 1)$ corresponds to squares of $(\mathbf{Z}/N\mathbf{Z})$ which is the "genus" of 1. Given the factorization of N, one can thus identify the "genus" of an element x. But without it, one can only compute the Jacobi symbol of x. Given an element x of Jacobi symbol 1, the quadratic residuosity problem asks to decide if x is in the principal genus with symbol $(1, 1)$ or in the genus with symbol $(-1, -1)$ without knowing the factorization of N.

For a class group $Cl(\Delta)$, the situation is similar: there are $2^{\mu-1}$ genera, where μ is related to the number of odd primes factor of the discriminant. One can define μ "assigned characters", whose joint values determine the genus (the product of all characters is always one, so we indeed have $2^{\mu-1}$ genera). The characters are for the majority Legendre symbols with respect to the odd prime factors of the discriminant. Let us describe in more details the setting that we will use to define our cryptosystem, where $\Delta = -8N$ and $N = pq$ is an RSA integer. In this case, where $\Delta = 0 \mod 4$ and $2N = 2 \mod 4$ (see [22, Prop. 3.11, Th. 3.15]), it holds that $\mu = 3$, and there are 4 genera. If f is a quadratic form, the first two assigned characters are respectively

$$\chi_p(f) := \left(\frac{a}{p}\right) \text{ and } \chi_q(f) := \left(\frac{a}{q}\right)$$

where a is any integer represented by f, respectively prime to p, prime to q. The third one, is

$$\chi_8(f) := (-1)^{(a^2-1)/8} \text{ or } \chi_{-8} := \chi_{-4} \cdot \chi_8(f) := (-1)^{(a-1)/2} \cdot (-1)^{(a^2-1)/8},$$

depending if $N \equiv 3 \mod 4$ or $N \equiv 1 \mod 4$, where a is any odd integer represented by f. The genus of the class of a form f is thus identified by $(\chi_p(f), \chi_q(f), \chi_8(f))$ or $(\chi_p(f), \chi_q(f), \chi_{-8}(f))$ depending on N modulo 4. The subgroup of squares of $Cl(\Delta)$ is the subgroup of forms of genus with symbol $(1, 1, 1)$, the three other genera have symbols $(-1, -1, 1)$, $(-1, 1, -1)$, $(1, -1, -1)$.

Given the complete factorization of Δ (thus of N), one can identify in polynomial time the genus of an element of the class group (see also [43, Theorem 6.3]). Without it, only χ_8 or χ_{-8} and the Jacobi symbol relative to N, *i.e.*, the product $\chi_p \cdot \chi_q$ can be computed.

The situation of elements of order $\leqslant 2$ of the class group is similar. In the general case, there are also $2^{\mu-1}$ such elements and there are classes of forms $(a, 0, c)$, (a, a, c) and (a, b, a) [22, Lemma 3.10, Prop. 3.11]). As a result, finding these elements is equivalent to factoring the discriminant. For example, the discriminant of the form $(a, 0, c)$ is $-4ac$.

Computing square roots can also be done efficiently given the factorization of the discriminant [43, Theorem 6.10]. The algorithm, due to Gauss, uses reduction of ternary forms, and the factorization of Δ is needed to extract square roots modulo Δ.

Two Classgroups. Starting with the NICE family of cryptosystems, the idea of using the relationship between two class groups has enabled many developments. Let us first consider Δ_K a fundamental negative discriminant: this means that either $\Delta_K \equiv 1 \pmod 4$ and Δ_K is square-free or $\Delta_K = 4m$ where m is square-free and $m \equiv 2, 3 \pmod 4$. This discriminant defines the maximal order $\mathcal{O}_{\Delta_K}$ of the quadratic field $Q(\sqrt{\Delta_K})$. Now let us consider a non fundamental discriminant $\Delta_\ell := \Delta_K \ell^2$ where ℓ is called a conductor. Then, there exists a surjective map $\varphi_\ell : Cl(\Delta_\ell) \to Cl(\Delta_K)$, moreover, for $\Delta_K < -4$, the kernel of this surjection is isomorphic to

$$\left(\mathcal{O}_{\Delta_K}/\ell\mathcal{O}_{\Delta_K}\right)^\times / \left(\mathbf{Z}/\ell\mathbf{Z}\right)^\times .$$

This isomorphism is used in [22, Theorem 7.24] to establish that for $\Delta_K < -4$,

$$h(\mathcal{O}_{\Delta_\ell}) = h(\mathcal{O}_{\Delta_K}) \cdot \ell \cdot \prod_{p|\ell} \left(1 - \left(\frac{\Delta_K}{p}\right)\frac{1}{p}\right). \tag{1}$$

3 A Class Group with a Cyclic Subgroup of order 2^k

In this section, we show how to generate a class group $Cl(\Delta)$ that contains a cyclic subgroup of order 2^k, inspired by the CL cryptosystem of [17], that builds a subgroup of order a prime q by using eq. 1 with a conductor $\ell = q$, and a fundamental discriminant Δ_K divisible by q. Unfortunately, the situation is not as simple as setting $\ell = 2^k$: as usual, working with 2 instead of an odd prime induces a lot of technicalities.

3.1 Choice for Δ_K

Let us begin with the generation of the fundamental discriminant Δ_K. Firstly, as we shall see in Subsection 3.4, we cannot reach any security in our applications if computing square roots in the class group is easy. As mentioned in Subsection 2.3 this means Δ_K must be hard to factor. As a consequence we will construct a

discriminant Δ_K from an RSA integer, $N = pq$, which is a first difference with the CL encryption.

Secondly, we will need to work with a subgroup of $Cl(\Delta)$ of odd order. This subgroup will be isomorphic to the subgroup of squares of $Cl(\Delta_K)$. The subgroup of squares has cardinality $\hat{s} := h(\Delta)/2^{\mu-1}$ where $2^{\mu-1}$ is the number of elements of order $\leqslant 2$. If we ensure that the 2−Sylow of $Cl(\Delta)$ is restricted to the elements of order $\leqslant 2$, then the subgroup of squares will correspond to the odd-part and $\hat{s}$ will be odd as required. This is done by ensuring that elements of order 2 are not squares, thereby imposing conditions on the prime factors of Δ_K.

Several choices are possible to construct Δ_K from N. In order for the conditions on the prime p and q to be the less restrictive possible, we choose to work with a fundamental discriminant $\Delta_K := -8N$. The next lemma gives the conditions that ensures that the 2−Sylow is restricted to elements of order 2.

Lemma 2. *Consider two distinct odd primes p and q of same bit-size, with values modulo 8 and Legendre symbols chosen according to Table 1. Let $N = pq$, and consider the fundamental discriminant $\Delta_k = -8N$. Then the 2−Sylow subgroup of $Cl(\Delta_K)$ is isomorphic to $\mathbf{Z}/2\mathbf{Z} \times \mathbf{Z}/2\mathbf{Z}$.*

$p \mod 8$	$q \mod 8$	(p/q)	(q/p)
1	3	-1	-1
1	5	-1	-1
3	1	-1	-1
3	5	$*$	
3	7	-1	1
5	1	-1	-1
5	3	$*$	
5	5	$*$	
5	7	-1	-1
7	3	1	-1
7	5	-1	-1

Table 1. Choices of (p, q) such that the 2−Sylow of $Cl(-8pq)$ is isomorphic to $\mathbf{Z}/2\mathbf{Z} \times \mathbf{Z}/2\mathbf{Z}$. The star $*$ means that there is no restriction on the values (p/q) and (p/q).

Proof. If $N = pq$, and $\Delta_k = -8N$, as seen in Subsection 2.3, $\mu = 3$, so there are $2^{3-1} - 1 = 3$ elements of order 2 in $Cl(\Delta_K)$. Looking at forms of the type $(a, 0, c)$ of discriminant $-4ac = -8N$, we find the following ones :

$$f_2 := (2, 0, N); f_p := (p, 0, 2q); f_q := (q, 0, 2p).$$

By hypothesis, $N > 2$, $2q > p$ and $2p > q$ so these three distinct forms are reduced, and their classes gives the 3 elements of order 2 of $Cl(\Delta_k)$.

We now compute the genus of f_2, f_p and f_q. For this, we need the value of χ_p, χ_q and χ_8 or χ_{-8} depending of the value of $N \mod 4$. Let us see in details the case of f_2. We have $\chi_p(f_2) = \left(\frac{a}{p}\right)$ where a is an integer represented by f_2

prime to p. One can choose $a = 2 = f_2(1,0)$. As a result

$$\chi_p(f_2) = \left(\frac{2}{p}\right) = (-1)^{\frac{p^2-1}{8}},$$

which gives 1 if $p \equiv 1,7 \pmod 8$, and -1 if $p \equiv 3,5 \pmod 8$.

Likewise, $\chi_q(f_2) = (2/q)$, whose value is determined by $q \pmod 8$.

If $N \equiv 3 \pmod 4$, to compute $\chi_8(f_2)$ we need an odd integer represented by f_2. We can choose $f_2(0,1) = N$. We then have

$$\chi_8(f_2) = (-1)^{\frac{N^2-1}{8}},$$

and again this value depends only on $N \mod 8$. If $N \equiv 1 \pmod 4$, we can also take N to evaluate $\chi_{-8}(f_2)$, which also depends only on $N \mod 8$.

For f_p, we use $f_p(0,1) = 2q$ to evaluate $\chi_p(f_p) = \left(\frac{2}{p}\right) \cdot \left(\frac{q}{p}\right)$. We also have $\chi_q(f_p) = \left(\frac{p}{q}\right)$ using $f_p(1,0) = p$, and the value of χ_8 and χ_{-8} are also determined using p. The genus of f_p can thus be determined by the values of $p \mod 8$ and the Legendre symbols $\left(\frac{p}{q}\right)$ and $\left(\frac{q}{p}\right)$. Note that by the law of quadratic reciprocity, these Legendre symbols are equal if $p \equiv 1 \pmod 4$ or $q \equiv 1 \pmod 4$ and $\left(\frac{p}{q}\right) = -\left(\frac{q}{p}\right)$ if $p \equiv q \equiv 3 \pmod 4$. The determination of the genus of f_q is similar to the one of f_p by exchanging the roles of q and p.

Now in order to have a $2-$Sylow subgroup isomorphic to $\mathbf{Z}/2\mathbf{Z} \times \mathbf{Z}/2\mathbf{Z}$, f_p, f_q and f_2 must all not be squares, which means that their genus must not have symbols $(1,1,1)$. As shown above, this only depends on the values of $p, q \pmod 8$ and of the relative Legendre symbols of p and q. By inspection of these values, we fill the Table 1 which gives all possibilities ensuring that f_p, f_q and f_2 are all not squares. □

3.2 Choice for Δ

We now want to construct a non fundamental discriminant such that $Cl(\Delta)$ contains a cyclic subgroup of order 2^k by using eq. 1. We will therefore consider as conductor ℓ a power of 2. As $\Delta_K = -8N$, we get $\left(\frac{\Delta_K}{2}\right) = 0$, and denoting, $\Delta = \ell^2 \Delta_K$, the class number $h(\Delta) = \ell h(\Delta_K)$, *i.e*, the kernel of the surjection φ_ℓ has order ℓ.

If we set $\ell = 2^k$, we thus get a subgroup of $Cl(\Delta)$ of order 2^k, and one can prove that this subgroup is cyclic, in our case where $\Delta_K = -8N$. Unfortunately, a similar computation to that of the proof of the next theorem shows that generators of this subgroup are not squares: the character χ_{-4} is equal to -1. This would break all our security assumptions, as the value of this character would leak the parity of discrete logarithms.

We thus set $\ell = 2^{k+1}$, and as a consequence, the kernel of φ_ℓ has order 2^{k+1}, and we can work in its cyclic subgroup of squares, of order 2^k. Note that other choices of Δ_K depending on N lead to similar constructions.

Theorem 1. *Let $\Delta_K = -8N$ with $N = pq$ as in Lemma 2. Let $\Delta = 2^{2k+2} \cdot \Delta_K$, then the class of $f := (2^{2k}, 2^{k+1}, 1 + 8N)$ is a square of order 2^k in $Cl(\Delta)$.*

Proof. Let $\ell = 2^{k+1}$ be the conductor. The strategy of the proof is as follows: we use the fact that $\ker \varphi_\ell$ is isomorphic to

$$G_\ell := (\mathcal{O}_{\Delta_K}/\ell\mathcal{O}_{\Delta_K})^\times / (\mathbf{Z}/\ell\mathbf{Z})^\times .$$

We will exhibit a system of representatives of this quotient group, and an element of order 2^k. As the isomorphism to $\ker \varphi_\ell$ is explicit, we will apply it to this element to get f of the same order 2^k in $Cl(\Delta)$. Then a computation of the assigned characters will show that f is a square.

The first step is to establish a system of representatives of the group G_ℓ. As $\Delta_K = 4m$, with $m := -2N$, $\mathcal{O}_{\Delta_K} = \mathbf{Z} + \mathbf{Z}\sqrt{m} \equiv \mathbf{Z}[X]/(X^2 - m)$ and $\mathcal{O}_{\Delta_K}/\ell\mathcal{O}_{\Delta_K} \equiv \mathbf{Z}/\ell\mathbf{Z}[X]/(X^2 - m)$. First observe that when $\ell = 2$, we have $m \equiv 0 \mod \ell$ and the invertible elements of $\mathcal{O}_{\Delta_K}/2\mathcal{O}_{\Delta_K}$ are therefore 1 and $1 + X$. For $k \geqslant 0$, we will then have

$$(\mathcal{O}_{\Delta_K}/\ell\mathcal{O}_{\Delta_K})^\times = \left\{a + bX, (a, b) \in (\mathbf{Z}/\ell\mathbf{Z})^\times \times \mathbf{Z}/\ell\mathbf{Z}\right\}$$

To get the group G_ℓ, we identify $a + bX$ with $ac + bcX$ for $c \in (\mathbf{Z}/\ell\mathbf{Z})^\times$. We then have the system of representatives:

$$\{1 + bX, b \in \mathbf{Z}/\ell\mathbf{Z}\} .$$

Now we show that $1 + 2X$ is of order 2^k in this group (one could prove that $1+X$ is of order 2^{k+1} and this group is cyclic, but considering $1+2X$ is sufficient for our applications). For this we first prove that for $k \geqslant 2$,

$$(1 + 2X)^{2^{k-1}} = 1 + 2^k X + 2^k X^2 \in \mathbf{Z}/2^{k+1}\mathbf{Z}[X]. \tag{2}$$

This can been shown by induction: the equality is clear for $k = 2$. Now suppose, that there exists a polynomial Q, s.t. $(1 + 2X)^{2^{k-2}} = 1 + 2^{k-1}X + 2^{k-1}X^2 + 2^k Q(X)$. Squaring both sides, we indeed get that $(1+2X)^{2^{k-1}} = 1+2^k X+2^k X^2$ modulo 2^{k+1}, which proves eqn. 2.

As a result we get that for $k \geqslant 2$, $(1 + 2X)^{2^{k-1}} \equiv 1 + 2^k m + 2^k X$ which is equivalent to $1+(1+2^k)^{-1}2^k X = 1+2^k X \neq 1$ in the group G_ℓ. But $(1+2^k X)^2 = 1$ which proves that $1+2X$ is of order 2^k for $k \geqslant 2$. It is straightforward to verify that $1 + 2X$ is also of order 2^k for $k = 0, 1$.

The next step is to map $1 + 2X$ in $\ker \varphi_\ell$ using the explicit isomorphism. This isomorphism consists in taking a representative α of the class of $1 + 2X$ in $\mathcal{O}_{\Delta_K}$, to compute a basis of the ideal $\alpha\mathcal{O}_{\Delta_K}$ and then to move it to $Cl(\Delta)$ by considering the class of the ideal $\alpha\mathcal{O}_{\Delta_K} \cap \mathcal{O}_\Delta$. The element $1 + 2X$ corresponds to the quadratic integer $\alpha := 1 + 2\sqrt{m} = 1 + \sqrt{\Delta_K}$. Following [10, Prop. 2.9], one writes $\alpha = \frac{x+y\sqrt{\Delta_K}}{2}$ with $x = y = 2$. Then applying the Extended Euclidean

algorithm on $y = 2$ and $(x + y\Delta_K)/2 = 1 + \Delta_K$ one gets $\kappa = 1 - 4N$, $\lambda = -1$, $\mu = 1$ s.t. $\kappa y + \lambda(x + y\Delta_K)/2 = \mu$. The ideal $\alpha\mathcal{O}_{\Delta_K}$ then corresponds to the form (a, b, c) where $a = N(\alpha)/\mu = 1 - \Delta = 1 + 8N$; and $b \equiv -\kappa x - \lambda(x + y)\Delta_K/2 \pmod{2a}$. One gets $b \equiv -2 - 8N \equiv 8N \pmod{2a}$, and $c = 2N$.

We then move this form to $Cl(\Delta)$ following [39, Algorithm 2] as a is odd so prime to the conductor 2^{k+1}. We get the form $(1 + 8N, 2^{k+4}N, 2^{2k+3}N)$. We then reduce this form, first by normalizing the b coefficient modulo $2a$: $2^{k+4}N - 2^{k+1}a = -2^{k+1}$, and computing the new value of c, we get 2^{2k}. As a result, this normalization gives the form $(1+8N, -2^{k+1}, 2^{2k})$ which is equivalent to the form $f = (2^{2k}, 2^{k+1}, 1+8N)$. Note that this form is reduced if $2^{2k} < 1+8N$ which will be the case in our applications.

The final step of the proof is to prove that f is a square. In $Cl(\Delta)$, the assigned characters are $\chi_8, \chi_{-4}, \chi_p, \chi_q$. Using $f(1, 0) = 2^{2k}$ which is a square, one gets that $\chi_p(f) = \chi_q(f) = 1$. Using the odd integer $f(0, 1) = 1 + 8N$, $\chi_{-4}(f) = (-1)^{4N} = 1$ and $\chi_8(f) = 1$ as $1 + 8N \equiv 1 \pmod 8$. □

3.3 The Gen_{2^k} algorithm

We depict our group generator in Algorithm 1. We first select a fundamental discriminant $\Delta_K := -8N$ as in Lemma 2. This ensures that the $2-$Sylow subgroup of $Cl(\Delta_K)$ has order 4 and $h(\Delta_K) = 4\hat{s}$ where $\hat{s}$ is odd, and $\hat{s}$ is the cardinality of the subgroup of squares of $Cl(\Delta_K)$.

Algorithm 1: Gen_{2^k}

Input: 1^λ
Result: pp
—
sample two random distinct $\eta(\lambda)$-bit primes p, q according to Table 1
$N := pq$
$\Delta_K := -8N$
$\Delta := 2^{2k+2} \cdot \Delta_K$
$f := (2^{2k}, 2^{k+1}, 1 + 8N) \in Cl(\Delta)$
sample r a random square of $Cl(\Delta)$
$h := r^{2^k} \in Cl(\Delta)$
compute $\tilde{s}$ an upper bound $h(\Delta_K)$
return $\mathsf{pp} := (f, h, \tilde{s})$

We then consider the class group $Cl(\Delta)$ of the non maximal order of discriminant $\Delta := 2^{2k+2} \cdot \Delta_K$ as in Lemma 1. This setting ensures that the class of the form $f := (2^{2k}, 2^{k+1}, 1 + 8N)$ generates a subgroup F of order 2^k of the group of squares of $Cl(\Delta)$.

For the discriminant Δ, the parameter μ equals 4 and there are $h(\Delta)/2^{\mu-1} = 2^{k+1} \cdot h(\Delta_K)/8 = 2^k \cdot \hat{s}$ squares in $Cl(\Delta)$ (cf. [22, Prop. 3.11]). We then consider

h a random 2^k-power of a square of $Cl(\Delta)$. By construction, the order of h, denoted s, is odd as it divides $\hat{s}$. We denote H the subgroup generated by h. Denoting G the cycling subgroup of the squares of $Cl(\Delta)$ of order $2^k s$, we have the isomorphism $G \simeq F \times H$ and F (resp. H) is the subgroup of 2^k-roots of unity of G (resp. the 2^k-th powers of G). Finally, we denote $\tilde{s}$ a known upper bound for s: the order s is *unknown* in our generator, but $\tilde{s}$ can be computed from an upper bound on the class number of $Cl(\Delta_K)$: $h(\Delta_K) < \frac{1}{\pi} \log |\Delta_K| \sqrt{|\Delta_K|}$, or obtain a slightly better bound using the analytic class number formula (cf [46]).

Size of p and q. The bitsize $\eta(\lambda)$ of the primes p and q is chosen such that the best algorithms for factoring $N := pq$ take 2^λ time. This ensures that computing s via the class number of $Cl(\Delta_K)$ takes more that 2^λ time as known algorithms for computing class numbers have worse complexities. In practice, for 112 bits (resp. 128 bits) of security, we take N of 2048 bits (resp. 3072 bits).

3.4 Assumptions

The semantic security of our linearly homomorphic encryption (and its threshold variant) relies on the following hard subgroup membership assumption, which is a natural extension of the HSM assumption underlying CL encryption [18]. In a nutshell, in the group G, we assume that it is hard to distinguish elements of the subgroup H, the 2^k-th powers, from random elements. As we shall see, in our particular context, this assumption implies the factorization assumption.

Definition 8 (HSM_{2^k} assumption). *Let $\mathcal{A}$ be an adversary for the HSM_{2^k} problem, its advantage is defined as:*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{HSM}_{2^k}}(\lambda) := \Big| 2 \cdot \Pr[b = b^* \ : \ \mathsf{pp} := (f, h, \tilde{s}) \xleftarrow{\$} \mathsf{Gen}_{2^k}(1^\lambda), x \hookleftarrow \mathcal{D}_{\mathcal{H}}, u \xleftarrow{\$} U((\mathbf{Z}/2^k\mathbf{Z})^\times), b \xleftarrow{\$} \{0,1\}, z_0 := h^x f^u, z_1 := h^x, b^* \leftarrow \mathcal{A}(\mathsf{pp}, z_b)] - 1 \Big|$$

where $\mathcal{D}_H$ is a distribution over the integers such that the distribution $\{h^x, \ x \hookleftarrow \mathcal{D}_H\}$ is at distance less than $2^{-\lambda}$ from the uniform distribution in H. The HSM_{2^k} assumption holds if for any probabilistic polynomial-time adversary $\mathcal{A}$, its advantage is negligible.

Relations with factoring and computing the class number. Let $\Delta_K = -8N$, as defined in Algo. 1. The class number $n := h(\Delta_K)$ is an integer multiple of s, the unknown order of h. So computing n allows to break the HSM_{2^k} assumption by checking if $z_b^n = 1$ or not.

The knowledge of n also allows to find the elements of order 2 of $Cl(\Delta_K)$, and, as shown in the background section, these elements gives the factorisation of Δ_K and thus of N. So computing n allows to factor N.

Conversely, there is no known method of computing $h(\Delta_K)$ given the factorisation of N, and best algorithms for computing $h(\Delta_K)$ have worse complexities than algorithms to factor N. However, the factorization of N allows to break

HSM_{2^k}. As already mentioned in Subsection 2.3, computing square roots and deciding if an element is a square in $Cl(\Delta)$ is feasible in polynomial-time if the factorisation of the discriminant is available. The HSM_{2^k} problem asks to distinguish between an element of the form $z_0 = h^x f^u$ for a random odd integer u and an element of the form $z_1 = h^x$, for a random x. As h is a 2^k-th power of a square, it is a square, and at least one of its square roots is a square itself, even if $k = 1$. This is not the case for $h^x f^u$. By construction, f is a square whose square roots are *not* squares: if there exists a square a such that $f = a^2$ then a would be a square of order 2^{k+1}, and we get a contradiction with the fact that the group of squares of $Cl(\Delta)$ has order $2^k \cdot \hat{s}$ where $\hat{s}$ is odd. So to distinguish the two cases, one has to compute a square root of the challenge element, and by inspection, compute the genera of the forms, check whether there exists a square root that is a square (and in this case the attacker outputs $b^\star = 1$) or not (and in this case the attacker outputs $b^\star = 0$).

It can be shown that the factorization allows in fact to compute a *partial discrete logarithm* of any $y = g^x$ in the class group, *i.e.*, the value $x \mod 2^k$ (and gives therefore a trapdoor to decrypt a ciphertext).

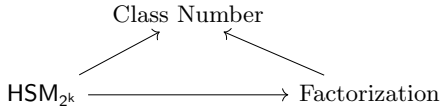

Fig. 2. Relations between the algorithmic assumptions underlying our protocols

4 Linearly Homomorphic Encryption Scheme on $\mathbf{Z}/2^k\mathbf{Z}$

4.1 Description of the new encryption scheme

Let $\mathcal{D}_H$ (resp. $\mathcal{D}_G$) be a distribution over the integers, such that $\{x \bmod s : x \xleftarrow{\$} \mathcal{D}_H\}$ (resp. $\{x \bmod 2^k s : x \xleftarrow{\$} \mathcal{D}_G\}$) is δ-close to the uniform distribution in $\{1, \ldots, s\}$ (resp. $\{1, \ldots, 2^k s\}$), where $\delta \leq 2^{-\lambda}$.

The distribution $\mathcal{D}_G$ is only used in the security proof and the distribution $\mathcal{D}_H$ can be instantiated by sampling x uniformly in $\{1, \ldots \tilde{s} \cdot 2^{\lambda+2}\}$ using the upper bound $\tilde{s}$ on s (cf. [17, Appendix C]).

Our linearly homomorphic encryption scheme on $\mathbf{Z}/2^k\mathbf{Z}$ is described in Figure 3, where the key generation algorithm takes as input the public parameters $\mathsf{pp} := (f, h, \tilde{s})$ that come from the Gen_{2^k} algorithm. There is no condition on the value of k, typical values are $32, 64$ or 128.

Correctness and Decryption The correctness of the protocol comes from the fact $c_2 \cdot c_1^{-\mathsf{sk}} = f^m \cdot \mathsf{pk}^r \cdot (h^r)^{-sk} = f^m \cdot (h^{\mathsf{sk}})^r \cdot (h^r)^{-\mathsf{sk}} = f^m$. To recover m from f^m, one has to compute a discrete logarithm. In this case, this discrete logarithm computation is trivial since f generates a subgroup of order 2^k. Pohlig-Hellman algorithm makes it possible to recover m by extracting m bit by bit. The algorithm to retrieve the discrete logarithm is described in Fig. 4. It consists mainly of $O(k^2)$ squaring in the class group.

Algorithm 2: KeyGen

Input: pp
Result: $(\mathsf{pk}, \mathsf{sk})$
—
sample $\mathsf{sk} \xleftarrow{\$} \mathcal{D}_H$
$\mathsf{pk} := h^{\mathsf{sk}}$
return $(\mathsf{pk}, \mathsf{sk})$

Algorithm 3: Encrypt

Input: $\mathsf{pp}, \mathsf{pk}, m \in \mathbf{Z}/2^k\mathbf{Z}$
Result: ciphertext (c_1, c_2)
—
sample $r \xleftarrow{\$} \mathcal{D}_H$
$c_1 := h^r$
$c_2 := f^m \mathsf{pk}^r$
return (c_1, c_2)

Algorithm 4: Decrypt

Input: $\mathsf{pp}, \mathsf{sk}, (c_1, c_2)$
Result: $m \in \mathbf{Z}/2^k\mathbf{Z} \cup \{\perp\}$
—
$M := c_2 \cdot c_1^{-\mathsf{sk}}$
if $M \notin F$ **then**
  return $\perp$
end
return $\log_f(M)$

Algorithm 5: EvalAdd

Input: $\mathsf{pp}, \mathsf{pk}, (c_1, c_2), (c'_1, c'_2)$
Result: ciphertext (c''_1, c''_2)
—
$c''_1 := c_1 \cdot c'_1$
$c''_2 := c_2 \cdot c'_2$
sample $r \xleftarrow{\$} \mathcal{D}_H$
return $(c''_1 \cdot h^r, c''_2 \cdot \mathsf{pk}^r)$

Algorithm 6: EvalScal

Input: $\mathsf{pp}, \mathsf{pk}, (c_1, c_2), \alpha$
Result: ciphertext (c'_1, c'_2)
—
$c'_1 := c_1^\alpha$
$c'_2 := c_2^\alpha$
sample $r \xleftarrow{\$} \mathcal{D}_H$
return $(c'_1 \cdot h^r, c'_2 \cdot \mathsf{pk}^r)$

Fig. 3. Linearly homomorphic encryption scheme with message space $\mathbf{Z}/2^k\mathbf{Z}$

4.2 Security of the encryption scheme

Semantic security

Theorem 2. *The scheme described in Figure 3 is semantically secure under chosen plaintext attacks* $(\mathsf{ind} - \mathsf{cpa})$ *if the* HSM_{2^k} *assumption holds.*

Proof. The proof proceeds as a sequence of games, starting with the real $\mathsf{ind}-\mathsf{cpa}$ experiment and ending in a game where the ciphertext statistically hides the random bit b chosen by the challenger. We denote S_i the event 'adversary $\mathcal{A}$ outputs $b = b^*$ in Game i'.

In Game 1, instead of sampling sk from $\mathcal{D}_H$, it is sampled from $\mathcal{D}_G$. The rest of the experiment is unchanged, so the only difference from $\mathcal{A}$'s view is the distribution of $\mathsf{pk} := h^{\mathsf{sk}}$. The distribution $\mathcal{D}_H$ is chosen such that $\{h^x : x \xleftarrow{\$} \mathcal{D}_H\}$ is δ-close to the uniform distribution in H. Furthermore, since s divides $2^k s$, sampling x in the previous expression also yields a distribution δ-close to the uniform distribution in H, so $|\Pr[S_1] - \Pr[S_0]| \leq 2\delta$.

Algorithm 7: Pohlig-Hellman

Input: $\mathsf{pp}, M \in F$
Result: m such that $M = f^m$
—
$m := 0$;
$\tilde{f} := f^{2^{k-1}}$;
for $i = 0$ *to* $k-1$ **do**
  if $(f^{-m}M)^{2^{k-1-i}} = \tilde{f}$ **then**
    $m := m + 2^i$;
  end
end
return m

Fig. 4. Pohlig-Hellman algorithm to compute $\log_f(M)$

In Game 2, the challenge ciphertext is computed as $c_1 := h^r$ and $c_2 := f^{m_b} c_1^{\mathsf{sk}}$, where $r \xleftarrow{\$} \mathcal{D}_H$. As $\mathsf{pk}^r = c_1^{\mathsf{sk}}$ this game is identical to the previous one, *i.e*,

$$\Pr[S_1] = \Pr[S_2].$$

In Game 3, the challenger additionally samples $u \xleftarrow{\$} (\mathbf{Z}/2^k\mathbf{Z})^\star$ uniformly at random. It sets $c_1 := h^r f^u$, and $c_2 := f^{m_b} c_1^{\mathsf{sk}}$. Now if $\mathcal{A}$ could distinguish game 2 from game 3, one could use $\mathcal{A}$ to solve the HSM_{2^k} problem, by setting c_1 to be the HSM_{2^k} challenge. Hence, denoting $\epsilon_{\mathsf{HSM}_{2^k}}$ the maximum advantage of any polynomial time adversary for the HSM_{2^k} problem, $\mathcal{A}$'s success probability in Game 2 and Game 3 can not differ by more than $\epsilon_{\mathsf{HSM}_{2^k}}$. This implies that

$$|\Pr[S_3] - \Pr[S_2]| \leq \epsilon_{\mathsf{HSM}_{2^k}}.$$

We now demonstrate that in game 3, the challenge bit b is perfectly hidden from $\mathcal{A}$'s view. Since $G \simeq H \times F$, the element $c_1 = h^r f^u$ information theoretically fixes the value of $(u \bmod 2^k)$ and of $(r \bmod s)$ from $\mathcal{A}$'s view. Furthermore $\mathcal{A}$ receives $c_2 = f^{m_b + u \cdot \mathsf{sk}} \mathsf{pk}^r$. Given c_1 and pk, the value of pk^r is information theoretically fixed, hence an unbounded adversary could infer $(m_b + u \cdot \mathsf{sk} \bmod 2^k)$.

Since sk is sampled from $\mathcal{D}_G$, the distribution followed by $(\mathsf{sk} \bmod 2^k s)$ is at negligible distance $\delta \leq 2^{-\lambda}$ of the uniform modulo $2^k s$. Furthermore, since s and 2^k are co-prime, $(\mathsf{sk} \bmod 2^k)$ is δ-close to the uniform modulo 2^k and is independent of $(\mathsf{sk} \bmod s)$. So even if $\mathsf{pk} = h^{\mathsf{sk}}$ fixes the value of $(\mathsf{sk} \bmod s)$, that of $(\mathsf{sk} \bmod 2^k)$ remains δ-close to $U(\mathbf{Z}/2^k\mathbf{Z})$ from $\mathcal{A}$'s view. Finally since u is invertible modulo 2^k, $(u \cdot \mathsf{sk} \bmod 2^k)$ is also δ-close to $U(\mathbf{Z}/2^k\mathbf{Z})$, and perfectly masks m_b. Therefore $|\Pr[S_2] - 1/2| \leq \delta$. Combining the probability equations, we conclude the proof with the following inequality:

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{ind\text{-}cpa}}(\lambda) \leq \epsilon_{\mathsf{HSM}_{2^k}} + 3\delta.$$

□

We note that a DDH assumption can also be used in the group G similarly to [17], which would lead to slightly different encryption scheme (using $g = hf$ instead of h).

Circuit privacy for linear functions The ciphertexts of our encryption protocol guarantees circuit privacy, in the sense that ciphertexts obtained through the homomorphic evaluation process are indistinguishable from fresh encryptions of the resulting message. This property is very useful in multi-party computation (see [19,20] for instance). More precisely, the definition is as follows.

Definition 9 (Circuit privacy for linear functions). *We say that a linearly homomorphic encryption* LHE *is private if there exists a probabilistic polynomial-time simulator* Sim *such that for any* $\lambda \in \mathbf{N}$*, for any* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\lambda, \mathsf{pp})$*, any pair of messages* m_1*,* m_2 *in the message space, and two ciphertexts* c_1 *and* c_2 *of* m_1 *and* m_2 *respectively, and any scalar* α*, the statistical distances between* $\mathsf{LHE.EvalAdd}(\mathsf{pp}, \mathsf{pk}, c_1, c_2)$ *and* $\mathsf{Sim}(1^\lambda, \mathsf{pp}, \mathsf{pk}, m_1 + m_2)$ *and between* $\mathsf{LHE.EvalScal}(\mathsf{pp}, \mathsf{pk}, c_1, \alpha)$ *and* $\mathsf{Sim}(1^\lambda, \mathsf{pp}, \mathsf{pk}, \alpha m_1)$ *are negligible.*

Theorem 3. *The scheme of Fig. 3 is circuit private for linear functions.*

Proof. For both pair of distributions, the simulator just encrypts the message it has as input ($m_1 + m_2$ or αm_1). The randomization applied during homomorphic evaluations (LHE.EvalAdd and LHE.EvalScal) ensures that the distributions are statistically close. □

4.3 Experiments

We have implemented our encryption protocol using Sagemath with calls to the PARI native C Library [50] for the operations in class groups. All benchmarks were done on a standard laptop (Intel Core i5-6267U @ 2.90GHz). Our experiments have been run for security levels of $\lambda = 112$ and 128 bits. The RSA modulus N has therefore respective sizes of 2048 bits and 3072 bits. The bit size of the ciphertexts is $2 \times \frac{3}{4}(5 + 2k + \ell_N)$ where ℓ_N is the bit size of N. The crucial part of KeyGen, Encrypt and Decrypt are exponentiations in class groups where the exponent is upper bounded by $\tilde{s} \approx \sqrt{N}$.

These timings show that even a straightforward implementation is practical, and an optimized C implementation of our system would drastically improve the running times.

5 Threshold Encryption on $\mathbf{Z}/2^k\mathbf{Z}$ with Trusted Setup

5.1 A TPKE scheme from class groups

In this subsection we adapt the PKE scheme from the previous section to the threshold setting. The threshold decryption relies on the LISS construction from Damgård and Thorbek [27] based on [2]. Let n be the number of servers, from the

k	λ (bits)	ciphertext (bits)	Setup	KeyGen	Encrypt	Decrypt
32	112	3176	0.037	0.101	0.096	0.101
	128	4712	0.231	0.212	0.214	0.222
64	112	3272	0.086	0.098	0.098	0.118
	128	4808	0.201	0.217	0.219	0.243
128	112	3464	0.076	0.103	0.105	0.178
	128	5000	0.398	0.230	0.230	0.309

Table 2. Bit size and running time of our homomorphic encryption in seconds.

threshold access structure $\mathbb{A}$, the dealer generates a share-generating matrix $\mathbf{M} \in \{0,1\}^e$, where $e \in O(n^{1+\sqrt{2}})$ which computes the Boolean formula associated to $\mathbb{A}$ as well as a surjective function $\psi : \{1,\ldots,d\} \mapsto P$ as defined in Subsection 2.2.

Our new threshold encryption protocol with message space of order 2^k is described in Fig. 5. We omit the $\mathsf{EvalAdd}$ and $\mathsf{EvalScal}$ algorithm that are exactly the same as the ones for our linearly homomorphic encryption scheme (Algorithms 5 and 6 of Fig. 3).

Theorem 4. *The scheme described in Fig. 5 achieves* $\mathsf{T\text{-}ind\text{-}cpa}$*-security under the* $\mathsf{ind-cpa}$ *security of the non-threshold scheme of Fig. 3.*

Proof. This theorem is a direct corollary of the privacy of the LISS and the $\mathsf{ind\text{-}cpa}$ of the non-threshold encryption scheme.

From an attacker against the $\mathsf{T\text{-}ind\text{-}cpa}$-security $\mathcal{A}$, we construct an attacker against the $\mathsf{ind\text{-}cpa}$ security of the basic scheme, which receives public parameters pp and a public key $\mathsf{pk} = h^x$ for an unknown x.

After $\mathcal{A}$ chooses an access structure $\mathbb{A}$, he is fed with pp and pk as ek. He chooses a set S outside $\mathbb{A}$, and waits for the corresponding secret keys. They are simulated after the computation of a sharing of 0, *i.e.*, the distribution vector is $\boldsymbol{\rho} = (0, \rho_1, \ldots, \rho_d)^T$ and the shares are $\mathbf{s} = (s_1, \ldots, s_d)^\top = \mathbf{M} \cdot \boldsymbol{\rho}$, where $\mathbf{M}$ is the matrix corresponding to the access structure $\mathbb{A}$. $\mathcal{A}$ receives the shares belonging to the servers in S.

Now, $\mathcal{A}$ can query partial decryptions: he queries the oracle on plaintext m and server i. The message m is encrypted as $\mathsf{ct} = \mathsf{Encrypt}(\mathsf{pp}, \mathsf{ek}, m) = (\mathsf{ct}_1, \mathsf{ct}_2)$. We must simulate the contributions that this honest party i computes, namely $\mathbf{d}_i := (c_1^{s_j})_{j \in \psi^{-1}(i)}$, from $\mathsf{pk}^r = \mathsf{ct}_1^x$. This is done as in [27] for the distribution of an exponentiation. Let $\boldsymbol{\kappa_S}$ be the sweeping vector of Def. 7 for S. Now, let R be a row in the distribution matrix $\mathbf{M}$ belonging to the honest server P_i and let s_j be one component of the server's share we computed from this row. Had we used $\boldsymbol{\rho}' = \boldsymbol{\rho} + x\boldsymbol{\kappa_S}$ instead of $\boldsymbol{\rho}$, then the share component coming from R would have been $s'_j = (\boldsymbol{\rho} + x\boldsymbol{\kappa_S})R = s_j + x\boldsymbol{\kappa_S}R$ instead. The observation is now that because we know ct_1^x and s, we can compute $\mathsf{ct}_1^{s'_j}$ even though we do not

Algorithm 8: Setup

Input: $1^\lambda, \mathbb{A}$
Result: $\mathsf{pp}, \mathsf{ek}, \mathbf{sk}$
—
generate $pp := (f, h, \tilde{s}) \xleftarrow{\$} \mathsf{Gen}_{2^k}(1^\lambda)$
sample $\mathsf{sk} \xleftarrow{\$} \{1, \ldots, 2^{\lambda+2}\tilde{s}\}$
$\mathsf{ek} := h^{\mathsf{sk}}$
Set $e, d \in O(n^{1+\sqrt{2}})$
Compute the matrix $\mathbf{M} \in \{0,1\}^{d \times e}$ that computes $\mathbb{A}$ // Benaloh-Leichter
sample $(\rho_2, \ldots, \rho_e) \xleftarrow{\$} [2^{l_0+\lambda}, 2^{l_0+\lambda}]^{e-1}$
$\boldsymbol{\rho} := (\mathsf{sk}, \rho_2, \ldots, \rho_e)^\top$
for $i \in \{1, \ldots, n\}$ **do**
  $\mathsf{sk}_i := (\mathbf{M}_j \cdot \boldsymbol{\rho})_{j \in \psi^{-1}(i)} \in \mathbf{Z}^{d_i}$
end
$\mathbf{sk} := (\mathsf{sk}_i)_{1 \leqslant i \leqslant n}$
return $\mathsf{pp}, \mathsf{ek}, \mathbf{sk}$

Algorithm 9: Encrypt

Input: $\mathsf{ek}, m \in \mathbf{Z}/2^k\mathbf{Z}$
Result: ciphertext $\mathsf{ct} := (c_1, c_2)$
—
sample $r \xleftarrow{\$} \mathcal{D}_H$
$c_1 := h^r$
$c_2 := f^m \mathsf{ek}^r$
return (c_1, c_2)

Algorithm 10: PartDec

Input: $\mathsf{pp}, \mathsf{sk}_i, \mathsf{ct}$
Result: $\mathbf{d}_i \in G \cup \{\perp\}$
—
parse ct as (c_1, c_2)
parse sk_i as $(s_j)_{j \in \psi^{-1}(i)}$
$\mathbf{d}_i := (c_1^{s_j})_{j \in \psi^{-1}(i)}$
return $\mathbf{d}_i$

Algorithm 11: FinalDec

Input: $\mathsf{pp}, \{\mathbf{d}_i\}_{i \in \mathcal{S}}$ for $\mathcal{S} \in \mathbb{A}$
Result: $m \in \mathbf{Z}/2^k\mathbf{Z} \cup \{\perp\}$
—
parse $\mathcal{S}$ as $(j_1, \ldots, j_t)$
compute $\boldsymbol{\lambda}_\mathcal{S} := (\boldsymbol{\lambda}_{j_1}^\top, \ldots, \boldsymbol{\lambda}_{j_t}^\top)^\top \in \{-1, 0, 1\}^{d_\mathcal{S}}$ such that

$$\boldsymbol{\lambda}_\mathcal{S} \cdot \mathbf{M}_{\psi^{-1}(\mathcal{S})} = (1, 0, \ldots, 0)^\top$$

where $d_\mathcal{S} := \sum_{i \in \mathcal{S}} d_i$ and $\boldsymbol{\lambda}_{j_i} := (\lambda_{j_1,1}, \ldots, \lambda_{j_1,d_{j_i}})^\top$ for all $i = 1, \ldots, t$.
compute $d := \prod_{i \in [t]} \prod_{k \in [d_{j_i}]} d_i^{\lambda_{j_i,k}}$
$M := c_2 \cdot d^{-1}$
if $M \notin F$ **then**
  return $\perp$
end
return $\log_f(M)$ using Algo. 7

Fig. 5. TPKE scheme with message space 2^k

know x. Concretely, we simulate the contribution from P by

$$\mathsf{ct}_1^{s_j}(\mathsf{ct}_1^x)^{\boldsymbol{\kappa_S} R} = \mathsf{ct}_1^{s_j + x\boldsymbol{\kappa_S} R} = \mathsf{ct}_1^{s'_j}.$$

After this partial decryption phase, $\mathcal{A}$ outputs his messages m_0 and m_1 which are forwarded to the ind-cpa challenger which answers with a challenge ciphertext $c^\star$ of one of this two messages $m_{b^\star}$, which is given to $\mathcal{A}$. After another series of queries for partial decryptions, answered in the same way, $\mathcal{A}$ outputs a bit b, which is set as the output of the ind-cpa adversary.

To see that the simulation is correct, we see that the simulated shares are statistically indistinguishable from the real shares by the privacy of the LISS scheme. Second, honest parties always output the correct value ct_1^x, by correctness of the LISS scheme. Finally, given ct_1, ct_1^x, the simulated contributions from honest parties are statistically indistinguishable, since the vector we use for the simulated sharing is $\boldsymbol{\rho}' = \boldsymbol{\rho} + x\boldsymbol{\kappa_S}$ which is statistically close to a uniformly chosen sharing vector for x. The advantage of the ind-cpa attacker will therefore be that of the T-ind-cpa attacker. □

In terms of efficiency, the only difference between the thresold scheme and the encryption scheme of Section 4 is in the decryption algorithms (encryption is exactly the same). Therefore, the additional costs come from the LISS, and translate in an additional number of exponentiations in the class group. Exact numbers depend on the considered access structure. For a concrete example, taking the access structure construction for a 2-out-of-3 policy (cf [53, Example 3.4 p. 26]), the shares have roughly the same bitsize as the secret key and we get 2 exponentiations (in total) for PartDec and a negligible extra cost in FinalDec consisting of multiplication and inversion since the reconstruction vector has components in $\{-1, 0, 1\}$. As a result, PartDec takes twice the classical decryption time (or the same with parallelisation).

5.2 Extensions

Some extensions and improvements (in terms of security or functionality) are possible for our threshold encryption scheme: we suggest few of them.

- *Robustness:* It informally captures that no malicious adversary can prevent a honest majority from decrypting a valid ciphertext. It can be achieved in our context by using Σ-protocols proving equality of discrete logarithms in groups of unknown orders to prove the validity of decryption shares.
- *Removing the trusted dealer:* It is one of the most interesting feature that can be achieved, especially compared to a potential Elgamal version of Benhamouda *et al.*'s scheme. It first requires to generate in a distributed manner an RSA modulus satisfying the needed congruences. Many efficient techniques can be employed, such as [21], secure against any subset of maliciously colluding parties. The class group can then be computed publicly and the factorization ignored. To share the secret key without trusted dealer, it is

possible to use verifiable linear integer secret sharing [53] and techniques from [35] to distribute key generation for discrete-log based cryptosystems. In addition, zero-knowledge arguments for several relations (well-formdness of a key or equality of discrete logarithms) in group of unknown orders will be needed and can be found in the literature [5,15,58,29].

– *CCA security:* Using techniques of [30] for their chosen ciphertext secure threshold cryptosystems from the Decision Composite Residuosity (DCR) assumption, it should be possible to make our threshold encryption scheme CCA secure (even though we would be loosing the crucial homomorphic encryption), as well as adaptively secure (i.e., secure against an adversary who dynamically corrupts servers throughout the protocol), and non-interactive (i.e., decryption servers do not interact amongst themselves but rather contribute, each, a single message). Note that several building blocks need to be adapted: for example, a Trapdoor Σ-protocol showing that an element is a 2^k-th power. As shown at the end of Subsection 3.4, the factorization of the discriminant could be the trapdoor of such a protocol.

6 Applications

We here discuss future work, and provide intuition for some of the many applications we see to our scheme.

6.1 Secure multi-party computation

The goal here is to devise an MPC protocol (for dishonest majority) that works over $\mathbf{Z}/2^k\mathbf{Z}$, and provides better (bandwidth) efficiency than current solutions.

The topic of malicious MPC for $\mathbf{Z}/2^k\mathbf{Z}$ has drawn significant attention since 2018, when Cramer et al. revelled their $\mathsf{SPD}\mathbf{Z}_{2^k}$ protocol [23] which aims at solving this issue.

Computations modulo 2^k, closely match what happens in a CPU, thereby allowing protocol designers to take advantage of tricks already known there. Typical examples being comparison operations and bitwise operations which seem to be easier modulo 2^k (and harder to emulate modulo p).

The solution from [23] follows a blueprint that is by now standard for many fast (maliciously) secure MPC protocols. The protocol phase is divided in two stages. An offline (slow) phase where some precomputation is done without knowing the actual inputs of the computation; and a very fast, information theoretic, phase which requires knowing the inputs and takes advantage of the data computed offline.

The offline stage consists, mainly, in creating sharings of many triplets of the form $[a], [b], [ab]$, where a and b are random in $\mathbf{Z}/2^k\mathbf{Z}$. These triplets are used to speed up the online phase.

The computations on the input data executed in the online phase require performing additions and multiplications. To add two shared secrets $[x]$, $[y]$, players simply add their shares non interactively. Multiplication is less straightforward.

In order to compute $[xy]$ *quickly*, given a (yet unused) triplet $[a]$, $[b]$, $[ab]$, players proceed as follows. First jointly open $[x]-[a]=c$ and $[y]-[b]=d$. Then, without further interaction, each player can compute: $[xy]=cd+[a]d+[b]c+[ab]$. Since the online phase is very fast (essentially the same for all protocols following this blueprint) the question is *how to improve efficiency of the offline stage?*

In SPDZ_{2^k} they use oblivious transfer, which is fast but expensive in terms of bandwidth consumption. One way of reducing bandwidth would be to rely on homomorphic encryption (and indeed the original SPDZ protocol uses somewhat homomorphic encryption for degree two polynomials to compute triplets). The issue is how to do this in the $\mathbf{Z}/2^k\mathbf{Z}$ setting, since we don't know many homomorphic encryption schemes that cope well with this setting.

To our knowledge, two solutions exist to this problem, and both have issues. The first is a protocol due to Orsini et al. [48], which presents significant efficiency gains with respect to [23], but remains very complex. The second, much simpler, is due to Catalano et al. [19]. Their protocol relies on the Joye-Libert encryption scheme, but has lower bandwidth gain and only works in the two party case. Indeed, though the Joye-Libert protocol allows for a message space of order 2^k, it is unclear how to enhance it with threshold decryption. Hence each player in the [19] protocol has their own public and secret key pair, and computing multiplications is performed via a protocol *à la* Gilboa [36] which entails a number of zero-knowledge proofs – hence the small gain in bandwidth consumption.

How does our scheme help? Our encryption scheme both allows for a message space of order 2^k, and for threshold decryption. Given both these properties, one can easily generate triplets as follows. Each player P_i chooses a random a_i, a random b_i, encrypts them $\mathsf{Encrypt}_{\mathsf{pk}}(a_i)$, $\mathsf{Encrypt}_{\mathsf{pk}}(b_i)$ and broadcasts these values. Every player homomorphically adds the shares it sent and received to obtain encryptions $\mathsf{Encrypt}_{\mathsf{pk}}(a)$, $\mathsf{Encrypt}_{\mathsf{pk}}(b)$, where $a=\sum_i a_i$, and $b=\sum_i b_i$. Then, using a trick from Catalano et al. [20], every player can multiply the underlying plaintexts to obtain $\mathsf{Encrypt}_{\mathsf{pk}}(ab)$. Finally each player P_i uses the partial decryption algorithm with its secret key sk_i to obtain an additive share c_i of ab.

6.2 Homomorphic secret sharing (HSS)

Homomorphic secret sharing is a form of secret sharing that allows parties to non-interactively perform computations on shared private inputs. HSS can be viewed as a distributed variant of homomorphic encryption: in HSS multiple parties are given a share of the inputs, and, without further interaction, they each perform (non interactively) homomorphic evaluations over these inputs to obtain a share of the desired output. HSS can be used instead of fully/somewhat homomorphic encryption in many scenarios, including low-communication MPC (e.g. [9]), private querying to remote databases (e.g. [7]), methods of succinctly generating correlated randomness (e.g. [6]), and more. Using our TPKE scheme, combined with recent techniques introduced by Orlandi et al [47], one should

be able to devise the first HSS protocol that efficiently performs computations modulo 2^k, without requiring a correctness/efficiency trade-off, and without the need to restrict the size of the shared inputs.

Details. Based on the breakthrough work by Boyle et al. [8], Fazio et al. [33] provided a blueprint to build an HSS scheme for an expressive class of programs[5]. Precisely, from the [8] protocol, which was based on (circular secure) Elgamal encryption, [33] abstract the key ingredients required of an encryption scheme to build an HSS.

1. The encryption scheme must be both message and key (linearly) homomorphic over a finite quotient.
2. Given an encryption of a small integer w, and substractive secret sharings $\langle x \rangle$ and $\langle \mathsf{sk}x \rangle$ (where sk is the decryption key), there must be a non-interactive method for parties to compute multiplicative shares of the group element g^{xw}, which lives in the ciphertext space of the encryption scheme.
3. A non-interactive technique to convert the multiplicative sharing of g^{xw} into an additive sharing of xw which lives in the plaintext space of the encryption scheme.

Our TPKE scheme naturally satisfies item 1, as it is linearly homomorphic, and threshold decryption can provide us the aforementioned key homomorphic property.

Regarding item 2, we leverage the Elgamal-like structure of our TPKE. Consider a ciphertext $(h^r, f^w \mathsf{pk}^r)$ encrypting w, where $\mathsf{pk} = h^{\mathsf{msk}}$. For each memory value x in the RMS program, the value of x and of $\mathsf{sk}x$ are each held as an additive secret sharing across parties (let us denote P_i's shares $\langle x \rangle_i$ and $\langle \mathsf{sk}x \rangle_i$). P_i's computes its' multiplicative share of f^{wx} as $g_i := (h^r)^{-\langle \mathsf{sk}x \rangle_i} (f^w \mathsf{pk})^{\langle x \rangle_i}$.

Item 3 has for long been the tricky part of the protocol. An ingenious distributed discrete logarithm (DDLog) protocol was first suggested by [8]. In their protocol, to obtain substractive shares of $z := xw$, parties P_0 and P_1, respectively owning shares g_0, g_1 such that $g_0 = g_1 g^{xw}$, agree upon some distinguished element $\tilde{g}$ that is not too far away from g_0, g_1 in terms of multiplications by g. If they find such a $\tilde{g}$, then party i can compute the distance of g_i from $\tilde{g}$ by brute force: by multiplying $\tilde{g}$ by g repeatedly, and seeing how many multiplications it takes to get to g_i. If $\tilde{g}$ isn't too far away, this should not be too inefficient. The primary challenge is agreeing upon a common point $\tilde{g}$. [8] had the parties first fix a set of random, distinguished points in the group; party i then finds the closest point in this set to g_i. As long as both parties find the same point, this will lead to a correct share conversion. To make this process efficient, the distance d between successive points can't be too large, as running time will be $O(d)$. But this induces an inherent $\approx 1/d$ probability of failure, in case a point lies between the original two shares and parties fail to agree. Furthermore, Dinur et al [31]

[5] Restricted Multiplication Straight-line Programs. This class captures polynomial-size branching programs, which includes arbitrary logspace computations and NC1 circuits.

showed that if one could do better than $1/d$ error probability in $O(\sqrt{d})$ steps, the algorithm could be used to improve the cost of finding discrete logarithms in an interval, a well-studied problem which is believed to be hard.

In recent work, Orlandi et al [47] overcame this barrier by leveraging the easy discrete logarithm subgroup present in the Paillier framework. As we also have such a setup, we benefit from their technique, and parties can both agree on a distinguished point and efficiently find the distance of a multiplicative share from that point, without requiring a correctness/efficiency trade-off. The high level idea (applied to our group elements), is that both g_0 and g_1 can be seen as elements of the coset $X_{g_0} := \{g_0, g_0 f, \ldots, g_0 f^{2^k-1}\}$. If both parties agree on a point $\tilde{h}$ in this set, then there exists z such that $\tilde{h} = g_0 f^z$, and so P_0 can efficiently compute $\log_F(\tilde{h} \cdot g_0^{-1}) = z$. Furthermore, since $g_0 = g_1 f^{xw}$, P_1 can efficiently compute $\log_F(\tilde{h} \cdot g_1^{-1}) = z + xw$. And it holds that $z + xw - z = xw$ as desired.

Now to agree on the point $\tilde{h}$ the parties compute the smallest element from X_{g_0}. This may be done using the surjection φ_{2^k} from $Cl(\Delta)$ to the class group of the maximal order $Cl(\Delta_K)$. Finally, we note that in prior work, the size of the shared inputs had to be bounded, either for efficiency, as in [8], or for correctness of computations in [47]. The upper bound in Orlandi et al's protocol ensures that no wrap around occurs modulo N. On the other hand, using our TPKE scheme with message space 2^k, we can set the order of the message space to be the modulus desired for practical computations, and potentially avoid such constraints.

6.3 Lossy trapdoor functions (LTDFs)

Lossy trapdoor functions, introduced by Peikert and Waters [52], are families of functions where injective functions are computationally indistinguishable from lossy functions, which lose many bits of information about their input. Among many interesting applications, LTDFs are known to imply chosen-ciphertext-secure PKE [52] or deterministic encryption [1] for instance.

Huge efficiency gains were obtained by Joye and Libert [41] over previous constructions from linearly homomorphic scheme, by leveraging the 2^k order of their message space in order to batch evaluation and process k-bit blocks of the input at once.

Applying both techniques to our linearly homomorphic PKE of Figure 3 would yield an efficient LTDF which supports evaluation over k-bit blocks at once. This allows for compact outputs of the functions. We note however that, due to the Elgamal-like structure of our underlying PKE, trapdoors and function descriptions would be larger than in the Joye-Libert LTDF: for inputs of size n, our trapdoors would require an extra n/k integers, while our function descriptions would require an extra n/k elements in H.

Acknowledgements. We thank Dario Catalano for helpful early discussions about these results and the anonymous reviewers for their comments in improving the work. This work was partially supported by the French ANR SANGRIA

project (ANR-21-CE39-0006) and the French PEPR Cybersecurité SecureCompute project (ANR-22-PECY-0003). This work also recieved funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program under projects PICOCRYPT (grant agreement No. 101001283), and TERMINET (grant agreement No. 957406), by the Spanish Government under projects SCUM (ref. RTI2018-102043-B-I00), by the Madrid Regional Government under project BLOQUES (ref. S2018/TCS-4339), and by a grant from Nomadic Labs and the Tezos foundation.

References

1. M. Bellare, A. Boldyreva, and A. O'Neill. Deterministic and efficiently searchable encryption. *CRYPTO 2007*, Vol. 4622 of *LNCS*, pp. 535–552. Springer, 2007.
2. J. C. Benaloh and J. Leichter. Generalized secret sharing and monotone functions. *CRYPTO'88*, Vol. 403 *LNCS*, pp. 27–35. Springer, 1990.
3. F. Benhamouda, J. Herranz, M. Joye, and B. Libert. Efficient cryptosystems from 2^k-th power residue symbols. *Journal of Cryptology*, 30(2):519–549, 2017.
4. J.-F. Biasse, M. J. Jacobson, and A. K. Silvester. Security estimates for quadratic field based cryptosystems. *ACISP 10*, Vol. 6168 *LNCS*, 233–247. Springer, 2010.
5. A. R. Block, J. Holmgren, A. Rosen, R. D. Rothblum, and P. Soni. Time- and space-efficient arguments from groups of unknown order. *CRYPTO 2021*, Vol. 12828 of *LNCS*, pp. 123–152, 2021 Springer, 2021
6. E. Boyle, G. Couteau, N. Gilboa, Y. Ishai, L. Kohl, and P. Scholl. Efficient pseudorandom correlation generators: Silent OT extension and more. *CRYPTO 2019*, Vol. 11694 of *LNCS*, pp. 489–518. Springer, 2019.
7. E. Boyle, N. Gilboa, and Y. Ishai. Function secret sharing. *EUROCRYPT 2015*, Vol. 9057 of *LNCS*, pp. 337–367. Springer, 2015.
8. E. Boyle, N. Gilboa, and Y. Ishai. Breaking the circuit size barrier for secure computation under DDH. *CRYPTO 2016*, *LNCS* 9814, 509–539. Springer, 2016.
9. E. Boyle, N. Gilboa, and Y. Ishai. Group-based secure computation: Optimizing rounds, communication, and computation. *EUROCRYPT 2017*, Vol. 10211 of *LNCS*, pp. 163–193. Springer, 2017.
10. J. Buchmann, C. Thiel, and H. Williams. *Short Representation of Quadratic Integers*, pp. 159–185. Springer, Dordrecht, 1995.
11. J. Buchmann and U. Vollmer. *Binary Quadratic Forms: An Algorithmic Approach*. Algorithms and Computation in Mathematics. Springer Berlin 2007.
12. J. Buchmann and H. C. Williams. A key-exchange system based on imaginary quadratic fields. *Journal of Cryptology*, 1(2):107–118, 1988.
13. B. Bünz, B. Fisch, and A. Szepieniec. Transparent SNARKs from DARK compilers. *EUROCRYPT 2020*, Vol. 12105 of *LNCS*, pp. 677–706. Springer, 2020.
14. G. Castagnos, D. Catalano, F. Laguillaumie, F. Savasta, and I. Tucker. Two-party ECDSA from hash proof systems and efficient instantiations. *CRYPTO 2019*, Vol. 11694 *LNCS*, 191–221. Springer, 2019.
15. G. Castagnos, D. Catalano, F. Laguillaumie, F. Savasta, and I. Tucker. Bandwidth-efficient threshold EC-DSA. *PKC 2020*, Vol. 12111 *LNCS*, 266–296. Springer, 2020.
16. G. Castagnos and F. Laguillaumie. On the security of cryptosystems with quadratic decryption: The nicest cryptanalysis. *EUROCRYPT 2009*, Vol. 5479 of *LNCS*, pp. 260–277. Springer, 2009.

17. G. Castagnos and F. Laguillaumie. Linearly homomorphic encryption from DDH. *CT-RSA 2015*, Vol. 9048 *LNCS*, pp. 487–505. Springer, 2015.
18. G. Castagnos, F. Laguillaumie, and I. Tucker. Practical fully secure unrestricted inner product functional encryption modulo p. *ASIACRYPT 2018*, vol. 11273 of *LNCS*, 733–764. Springer, 2018.
19. D. Catalano, M. Di Raimondo, D. Fiore, and I. Giacomelli. Mon$\mathbb{Z}_{2^k}$a: Fast maliciously secure two party computation on $\mathbb{Z}_{2^k}$. *PKC 2020*, Vol. 12111 *LNCS*, pp. 357–386. Springer, 2020.
20. D. Catalano and D. Fiore. Using linearly-homomorphic encryption to evaluate degree-2 functions on encrypted data. *CCS 2015*, pp. 1518–1529. ACM, 2015.
21. M. Chen, C. Hazay, Y. Ishai, Y. Kashnikov, D. Micciancio, T. Riviere, A. Shelat, M. Venkitasubramaniam, and R. Wang. Diogenes: Lightweight scalable rsa modulus generation with a dishonest majority. In *2021 IEEE Symposium on Security and Privacy (SP)*, pp. 590–607, 2021.
22. D. Cox. *Primes of the Form $x^2 + ny^2$: Fermat, Class Field Theory, and Complex Multiplication*. Pure and Applied Mathematics, Wiley, 2014
23. R. Cramer, I. Damgård, D. Escudero, P. Scholl, and C. Xing. SPD $\mathbb{Z}_{2^k}$: Efficient MPC mod 2^k for dishonest majority. *CRYPTO 2018*, Vol. 10992 of *LNCS*, pp. 769–798. Springer, 2018.
24. R. Cramer, I. Damgård, and J. B. Nielsen. Multiparty computation from threshold homomorphic encryption. *EUROCRYPT 2001*, Vol. 2045 of *LNCS*, pp. 280–299. Springer, 2001.
25. R. Cramer and S. Fehr. Optimal black-box secret sharing over arbitrary Abelian groups. *CRYPTO 2002*, Vol. 2442 of *LNCS*, pp. 272–287. Springer, 2002.
26. I. Damgård and M. Jurik. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. *PKC 2001*, Vol. 1992 of *LNCS*, pp. 119–136. Springer, 2001.
27. I. Damgård and R. Thorbek. Linear integer secret sharing and distributed exponentiation. *PKC 2006*, Vol. 3958 of *LNCS*, pp. 75–90. Springer, 2006.
28. P. Das, M. J. Jacobson Jr., and R. Scheidler. Improved efficiency of a linearly homomorphic cryptosystem. In *Codes, Cryptology and Information Security*, pp. 349–368. Springer International Publishing, 2019.
29. Y. Deng, S. Ma, X. Zhang, H. Wang, X. Song, and X. Xie. Promise ς-protocol: How to construct efficient threshold ecdsa from encryptions based on class groups. *ASIACRYPT 2021*, pp. 557–586, Cham, Springer, 2021
30. J. Devevey, B. Libert, K. Nguyen, T. Peters, and M. Yung. Non-interactive CCA2-secure threshold cryptosystems: Achieving adaptive security in the standard model without pairings. *PKC 2021*, Vol. 12710 of *LNCS*, pp. 659–690. Springer, 2021.
31. I. Dinur, N. Keller, and O. Klein. An optimal distributed discrete log protocol with applications to homomorphic secret sharing. *CRYPTO 2018*, vol. 10993 of *LNCS*, 213–242. Springer, 2018.
32. S. Dobson, S. Galbraith, and B. Smith. Trustless unknown-order groups. *Mathematical Cryptology*, 1(1):1–15, 2021.
33. N. Fazio, R. Gennaro, T. Jafarikhah, and W. E. Skeith. Homomorphic secret sharing from paillier encryption. *Provable Security*, pp. 381–399, Springer, 2017.
34. P.-A. Fouque, G. Poupard, and J. Stern. Sharing decryption in the context of voting or lotteries. *FC 2000*, Vol. 1962 of *LNCS*, pp. 90–104. Springer, 2001.
35. R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Secure distributed key generation for discrete-log based cryptosystems. *Journal of Crypto.*, 20(1):51–83, 2007.
36. N. Gilboa. Two party RSA key generation. *CRYPTO'99*, Vol. 1666 of *LNCS*, pp. 116–129. Springer, 1999.

37. S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.
38. S. Hoory, A. Magen, and T. Pitassi. Monotone circuits for the majority function. *APPROX 2006*, Vol. 4110 of *LNCS*, pp. 410–425. Springer, 2006.
39. D. Hühnlein, M. J. Jacobson Jr., S. Paulus, and T. Takagi. A cryptosystem based on non-maximal imaginary quadratic orders with fast decryption. In*EUROCRYPT'98*, Vol. 1403 of *LNCS*, pp. 294–307. Springer, 1998.
40. M. J. Jacobson and A. J. van der Poorten. Computational aspects of nucomp. *Algorithmic Number Theory*, pp. 120–133, Springer, 2002.
41. M. Joye and B. Libert. Efficient cryptosystems from 2^k-th power residue symbols. *EUROCRYPT 2013*, Vol. 7881 of *LNCS*, pp. 76–92. Springer, 2013.
42. J. Katz and M. Yung. Threshold cryptosystems based on factoring. *ASIACRYPT 2002*, Vol. 2501 of *LNCS*, 192–205. Springer, 2002.
43. J. Lagarias. Worst-case complexity bounds for algorithms in the theory of integral quadratic forms. *Journal of Algorithms*, 1(2):142 – 186, 1980.
44. R. W. F. Lai and G. Malavolta. Subvector commitments with application to succinct arguments. *CRYPTO 2019*, Vol. 11692 *LNCS*, 530–560. Springer, 2019.
45. H. Lipmaa. Secure accumulators from euclidean rings without trusted setup. *ACNS 12*, Vol. 7341 of *LNCS*, pp. 224–240. Springer, 2012.
46. K. S. McCurley. Cryptographic key distribution and computation in class groups. *NATO Advanced Study Inst. on Number Theory and Applications*, Kluwer, 1989.
47. C. Orlandi, P. Scholl, and S. Yakoubov. The rise of paillier: Homomorphic secret sharing and public-key silent OT. *EUROCRYPT 2021*, Vol. 12696 of *LNCS*, pp. 678–708. Springer, 2021.
48. E. Orsini, N. P. Smart, and F. Vercauteren. Overdrive2k: Efficient secure MPC over $\mathbb{Z}_{2^k}$ from somewhat homomorphic encryption. *CT-RSA 2020*, Vol. 12006 of *LNCS*, pp. 254–283. Springer, 2020.
49. P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. *EUROCRYPT'99*, Vol. 1592 *LNCS*, pp. 223–238. Springer, 1999.
50. PARI Group, Univ. Bordeaux. *PARI/GP version* `2.11.4`, 2020.
51. S. Paulus and T. Takagi. A new public-key cryptosystem over a quadratic order with quadratic decryption time. *Journal of Cryptology*, 13(2):263–272, Mar. 2000.
52. C. Peikert and B. Waters. Lossy trapdoor functions and their applications. *40th ACM STOC*, 187–196. ACM Press, 2008
53. R. Thorbek. *Linear Integer Secret Sharing*. PhD thesis, Department of Computer Science - University of Aarhus, 2009.
54. S. A. K. Thyagarajan, G. Castagnos, F. Laguillaumie, and G. Malavolta. Efficient cca timed commitments in class groups. *ACM CCS '21*, pp. 2663–2684, 2021.
55. I. Tucker. *Functional encryption and distributed signatures based on projective hash functions, the benefit of class groups*. PhD thesis, Université de Lyon, 2020.
56. L. Valiant. Short monotone formulae for the majority function. *Journal of Algorithms*, 5(3):363–366, 1984.
57. B. Wesolowski. Efficient verifiable delay functions. *Journal of Cryptology*, 33(4):2113–2147, 2020.
58. T. H. Yuen, H. Cui, and X. Xie. Compact zero-knowledge proofs for threshold ECDSA with trustless setup. *PKC 2021*, Vol. 12710 of *LNCS*, pp. 481–511. Springer, 2021.