# Encryption Switching Protocols Revisited: Switching Modulo p

Guilhem Castagnos, Laurent Imbert, Fabien Laguillaumie

# Encryption Switching Protocols Revisited: Switching modulo $p$

Guilhem Castagnos[1], Laurent Imbert[2] and Fabien Laguillaumie[2,3]

[1] IMB UMR 5251, Université de Bordeaux, LFANT/INRIA
[2] CNRS, Université Montpellier/CNRS LIRMM
[3] Université Claude Bernard Lyon 1, CNRS/ENSL/INRIA/UCBL LIP

(This is an extended version of the paper published at CRYPTO 2017)

**Abstract.** At CRYPTO 2016, Couteau, Peters and Pointcheval introduced a new primitive called *encryption switching protocols*, allowing to switch ciphertexts between two encryption schemes. If such an ESP is built with two schemes that are respectively additively and multiplicatively homomorphic, it naturally gives rise to a secure 2-party computation protocol. It is thus perfectly suited for evaluating functions, such as multivariate polynomials, given as arithmetic circuits. Couteau et al. built an ESP to switch between Elgamal and Paillier encryptions which do not naturally fit well together. Consequently, they had to design a clever variant of Elgamal over $\mathbf{Z}/n\mathbf{Z}$ with a costly shared decryption.
In this paper, we first present a conceptually simple generic construction for encryption switching protocols. We then give an efficient instantiation of our generic approach that uses two well-suited protocols, namely a variant of Elgamal in $\mathbf{Z}/p\mathbf{Z}$ and the Castagnos-Laguillaumie encryption which is additively homomorphic over $\mathbf{Z}/p\mathbf{Z}$. Among other advantages, this allows to perform all computations modulo a prime $p$ instead of an RSA modulus. Overall, our solution leads to significant reductions in the number of rounds as well as the number of bits exchanged by the parties during the interactive protocols. We also show how to extend its security to the malicious setting.

## 1 Introduction

Through interactive cryptographic protocols, secure multi-party computation (MPC) allows several parties to compute the image of a pre-agreed function of their private inputs. At the end of the interaction, anything that a party (or a sufficiently small coalition of parties) has learned from the protocol could have been deduced from its public and secret inputs and outputs. In other words, the adversary's view can be efficiently forged by a simulator that has only access to the data publicly known by the adversary. This important area of research emerged in the 80's with the works of Yao [46] and Goldreich, Micali and Wigderson [22]. Formal security notions can be found in [34,4,8]. Initially considered as a theoretical subject due to overly inefficient protocols, MPC has

nowadays reached a reasonable complexity and has became relevant for practical purposes [6] especially in the 2-party case [40,33,31]. Several techniques may be used to design secure multi-party computation. Some recently proposed solutions use or combine tools from oblivious transfer [3,30], secret sharing with pre-processing [37,16], garbled circuits [31], homomorphic encryption [11,15], and somewhat or fully homomorphic encryption [5,2].

In [10], Couteau, Peters and Pointcheval formalized an innovative technique to securely compute functions between two players, thanks to interactive cryptographic protocols called *encryption switching protocols* (ESP). This mechanism permits secure 2-party computations against semi-honest adversaries (honest-but-curious) as well as malicious adversaries, i.e. opponents which might not follow the specifications of the protocol. Couteau et al.'s proposal relies on a pair of encryption schemes $(\Pi_+, \Pi_\times)$ which are respectively additively and multiplicatively homomorphic and which share a common message space. Furthermore, there exists switching algorithms to securely convert ciphertexts between $\Pi_+$ and $\Pi_\times$. More precisely, there exists a protocol $\mathsf{Switch}_{+\to\times}$ which takes as input an encryption $C_m^+$ of a message $m$ under $\Pi_+$, and returns a ciphertext $C_m^\times$ of the same message $m$ under $\Pi_\times$. Symmetrically, there exists a second protocol $\mathsf{Switch}_{\times\to+}$ which computes a ciphertext for $m$ under $\Pi_+$ when given a ciphertext for $m$ under $\Pi_\times$. The advantage of this construction is that it benefits from the intrinsic efficiency of multiplicatively homomorphic encryption like Elgamal [18] or additively homomorphic encryption like Paillier [38]. In [10], Couteau et al. present a natural construction for secure 2-party computation from any ESP.

**Applications.** Two-party computation is the most important application of an ESP. In [11], an MPC protocol is built from only an additively homomorphic encryption scheme which is a natural alternative to an ESP. The round complexity of their protocol is in $\mathcal{O}(d)$, where $d$ is the depth of the circuit $\mathcal{C}$ to be evaluated, and if we suppose that the multiplicative gates can be evaluated in parallel at each level. With an ESP, gathering the additive and multiplicative gates separately would imply a dramatic improvement. Fortunately, the result by Valiant, Skyum, Berkowitz and Rackoff from [45, Theorem 3], states that for any circuit $\mathcal{C}$ of size $s$ and degree $d$ computing a polynomial $f$, there is another circuit $\mathcal{C}'$ of size $\mathcal{O}(s^3)$ and depth $\mathcal{O}(\log(s)\log(d))$ which computes the same polynomial $f$. Moreover, Allender, Jiao, Mahajan and Vinay showed that the circuit $\mathcal{C}'$ is by construction layered (see [1]), in the sense that it is composed of layers whose gates are all the same and alternatively $+$ and $\times$. Roughly speaking, $\mathcal{C}'$ is of the form $(\sum\prod)^{\mathcal{O}(\log(s)\log(d))}$ where $\sum$ has only additive gates and $\prod$ has only multiplicative gates. In other words, the polynomial $f$ can be written as a composition of $\mathcal{O}(\log(s)\log(d))$ polynomials written in a sparse representation. The ESP allows to treat each $\sum$ and $\prod$ independently, so that the number of switches and therefore the number of rounds is essentially $\mathcal{O}(\log(s)\log(d))$, instead of $\mathcal{O}(d)$ for [11]. Any enhancement of an ESP will naturally improve any protocol which requires to evaluate on encrypted data a polynomial given in the

form of a sum of monomials. Especially it is well-suited to oblivious evaluation of multivariate polynomials [36,27,42] or private disjointness testing [47].

**Related works.** The idea of switching between ciphertexts for different homomorphic encryption schemes was first proposed by Gavin and Minier in [20] in the context of oblivious evaluation of multivariate polynomials. They proposed to combine a variant of Elgamal over $(\mathbf{Z}/N\mathbf{Z})^*$ (where $N$ is an RSA modulus) with a Goldwasser-Micali encryption protocol [23]. Unfortunately, as noticed by Couteau et al. [10], their design contains a serious flaw which renders their scheme insecure (the public key contains a square root of unity with Jacobi symbol $-1$, which exposes the factorization of $N$). Another attempt was proposed in [44] with a compiler designed to embed homomorphic computation into C programs to operate on encrypted data. The security of this construction relies on a very strong assumption since switching between the encryption schemes is done using a secure device which decrypts and re-encrypts using the secret key. In [29], Lim, Tople, Saxena and Chang proposed a primitive called *switchable homomorphic encryption* implemented using Paillier and Elgamal, in the context of computation on encrypted data. Again, this proposal uses an insecure version of Elgamal, which does not satisfy the indistinguishability under a chosen plaintext attack. It is indeed very difficult to design two compatible encryption schemes from unrelated protocols like Paillier and Elgamal. Couteau et al. managed to tune Elgamal so that it can switch with Paillier, but their construction remains fairly expensive. In particular, they constructed a variant of Elgamal over $(\mathbf{Z}/n\mathbf{Z})^*$, where $n$ is an RSA modulus, which is the same as the Paillier modulus. As Elgamal is secure only in the subgroup $\mathbf{J}_n$ of $(\mathbf{Z}/n\mathbf{Z})^*$ of elements of Jacobi symbol $+1$, they need a careful encoding of the group $(\mathbf{Z}/n\mathbf{Z})^*$. The security relies on the DDH assumption in $\mathbf{J}_n$ and the quadratic residuosity assumption in $(\mathbf{Z}/n\mathbf{Z})^*$. Because their Elgamal variant does not support a simple 2-party decryption (a Paillier layer has to be added to Elgamal in order to simulate a threshold decryption), the switching protocols are intricate and specific to their construction.

**Our contributions and overview of our results.** In this paper, we first propose a generic ESP inspired by Couteau et al's solid basis. Our construction relies on the existence of an additively homomorphic encryption $\Pi_+$ and a multiplicatively homomorphic encryption $\Pi_\times$ which support a 1-round threshold decryption and achieve classical security properties (IND-CPA and zero-knowledge of the 2-party decryption). Because the message spaces must be compatible, we suppose that $\Pi_+$ works over a ring $\mathcal{R}$ and $\Pi_\times$ over a monoid $\mathcal{M}$ with $\mathcal{R}\cap\mathcal{M} = \mathcal{R}^*$ where $\mathcal{R}^*$ is the set of invertible elements of $\mathcal{R}$. A major issue when designing an ESP is to embed the zero message[4] into the message space for $\Pi_\times$, while preserving the homomorphic and security properties. In Section 4.2, we propose

[4] The zero message has to be taken into account since it can arise easily by homomorphically subtracting two equivalent ciphertexts of the same message.

a generic technique to do so, inspired by the approach employed in [10]. Contrary to their construction, our switching protocols over $\mathcal{R}^*$ (i.e. without the zero-message) are symmetrical, i.e. both $\mathsf{Switch}_{+\to\times}$ and $\mathsf{Switch}_{\times\to+}$ follow the same elementary description given in Figure 3. This is possible for two reasons: first because we suppose that both $\Pi_+$ and $\Pi_+$ admit a single round 2-party decryption, and second because they both possess a $\mathsf{ScalMul}$ algorithm which takes as input a ciphertext of $m$ and a plaintext $\alpha$ and outputs a ciphertext of $\alpha \times m$ (which is why we consider a ring as the message space for $\Pi_+$ instead of an additive group).

Besides, they are very efficient: as detailed in Section 5.3, they only require 2 rounds, whereas Couteau et al.'s $\mathsf{Switch}_{\times\to+}$ needs 6. Our full switching protocols work over $\mathcal{R}^* \cup \{0\}$. They are built on top of the switching protocols over $\mathcal{R}^*$ (i.e. without 0), plus some additional tools like 2-party re-encryption, encrypted zero test, and a 2-party protocol to homomorphically compute a product under $\Pi_+$ (see Figure 1). Our security proofs are simpler than Couteau et al's. In terms of round complexity, the savings are substantial: our full ESP protocols require 7 and 4 rounds respectively, whereas Couteau et al's ESP need 7 and 11.

In a second part, we propose an efficient instantiation of our generic protocol over the field $\mathbf{Z}/p\mathbf{Z}$. Working over $\mathbf{Z}/p\mathbf{Z}$ has several advantages compared to $\mathbf{Z}/n\mathbf{Z}$ (for an RSA modulus $n$): it means true message space equality, instead of computational equality. It also means faster arithmetic by carefully choosing the prime $p$. Our instantiation combines a variant of Elgamal together with the Castagnos-Laguillaumie additively homomorphic encryption from [9]. Because Elgamal is only secure in the subgroup of squares modulo $p$, our variant over $\mathbf{Z}/p\mathbf{Z}^*$, denoted $\mathsf{Eg}^*$, encodes the messages into squares and adds the encryption of a witness bit (i.e. the Legendre Symbol) under Goldwasser-Micali [23] for its homomorphic properties modulo 2. For $\Pi_+$, we use a variant of the Castagnos-Laguillaumie encryption scheme ($\mathsf{CL}$) described in [9, Section 4]. We work over (subgroups of) the class group of an order of a quadratic field of discriminant $\Delta_p = -p^3$. Computations are done in this class group. The elements are represented by their unique reduced representative, i.e. by two integers of size $\sqrt{|\Delta_p|}$. Thus, an element of the class group requires $3 \log p$ bits. Under slightly different security assumptions, it is possible to further reduce the size of the elements and to achieve a better bit complexity. We discuss these implementation options in Section 5.3 and compare their costs with the ESP from Couteau et al. [10]. Our ESP protocol reduces the round complexity by a factor of almost 3 in the $\times \to +$ direction, while remaining constant in the other direction. Using the variant of $\mathsf{CL}$ optimized for size, the bit complexity is also significantly reduced in the $\times \to +$, while remaining in the same order of magnitude in the other.

We also propose improvements on $\mathsf{CL}$ that can be on independent interest. That system makes exponentiations in a group whose order is unknown but where a bound is known. We show that using discrete Gaussian distribution instead of uniform distribution improves the overall computational efficiency of the scheme. Moreover in order to use our generic construction, we devise a 2-party decryption for $\mathsf{CL}$.

Eventually we discuss in Section 6 how to adapt our generic construction and our instantiation against malicious adversaries.

## 2 Cryptographic Building Blocks

In this section, we recall some classical definitions and operations that will be useful in the rest of the paper.

### 2.1 Homomorphic encryption schemes.

In Section 3, we will give a definition of *Encryption Switching Protocols (ESP)*, previously proposed in [10]. An ESP allows to switch a ciphertext under an encryption protocol $\Pi_1$ into a ciphertext under another encryption protocol $\Pi_2$, and vice versa. ESP require the protocols $\Pi_1$ and $\Pi_2$ to be (partially) homomorphic. In this paper, we consider ESP between an additively homomorphic encryption $\Pi_+$ and a multiplicatively homomorphic encryption $\Pi_\times$.

In Definitions 1 and 2 below, we define $\Pi_+$ and $\Pi_\times$ formally in a generic context. An additive homomorphic encryption is most commonly defined over a group. In our setting, $\Pi_+$ is defined over a ring $\mathcal{R}$ to guarantee that for $m, m' \in \mathcal{R}$, the product $m \times m'$ is well defined. For genericity $\Pi_\times$ is defined over an algebraic structure with a single associative binary operation (denoted $\times$) and an identity element; i.e. a monoid. By doing so, our definition encapsulates encryption schemes over $(\mathbf{Z}/pq\mathbf{Z})^* \cup \{0\}$ (with $p, q$ primes) such as [10], as well as our instantiation over $\mathbf{Z}/p\mathbf{Z}$ presented in Section 5.

**Definition 1 (Additively homomorphic encryption).** *Let* $(\mathcal{R}, +, \times, 1_\mathcal{R}, 0_\mathcal{R})$ *be a ring. An* additively homomorphic encryption scheme *over the message space* $\mathcal{R}$ *is a tuple* $\Pi_+ = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt}, \mathsf{Hom}_+, \mathsf{ScalMul})$ *such that:*

$\mathsf{Setup}$ *is a PPT algorithm algorithm which takes as input a security parameter* $1^\lambda$ *and outputs public parameters* $\mathsf{params}$ *(these public parameters will be omitted in the algorithms' inputs).*

$\mathsf{KeyGen}$ *is a PPT algorithm taking public parameters as inputs and outputting a pair of public and secret key* $(pk, sk)$*.*

$\mathsf{Encrypt}$ *is a PPT algorithm which takes as input some public parameters, a public key* $pk$ *and a message* $m \in \mathcal{R}$*, and outputs an encryption* $c$*.*

$\mathsf{Decrypt}$ *is a PPT algorithm which takes as input public parameters, a public key* $pk$ *(omitted in* $\mathsf{Decrypt}$*'s input), a secret key* $sk$ *and a ciphertext* $c$*, and outputs a message* $m \in \mathcal{R}$*.*

$\mathsf{Hom}_+$ *is a PPT algorithm which takes as inputs some public parameters, a public key* $pk$ *and two ciphertexts* $c$ *and* $c'$ *of* $m \in \mathcal{R}$ *and* $m' \in \mathcal{R}$ *respectively, and outputs a ciphertext* $c''$ *such that* $\Pi_+.\mathsf{Decrypt}(sk, c'') = m + m' \in \mathcal{R}$*.*

$\mathsf{ScalMul}$ *is a PPT algorithm which takes as inputs some public parameters, a*

*public key $pk$, a ciphertext $c$ of $m \in \mathcal{R}$ and a plaintext $\alpha \in \mathcal{R}$, and outputs a ciphertext $c'$ such that $\Pi_+.\mathsf{Decrypt}(sk, c') = \alpha \times m \in \mathcal{R}$.*

*Remark 1.* A generic algorithm for computing $\Pi_+.\mathsf{ScalMul}(pk, c, \alpha)$ is given by $\mathsf{2Mul}_+(c, \Pi_+.\mathsf{Encrypt}(\alpha))$, where $\mathsf{2Mul}_+$ is an interactive PPT algorithm which computes homomorphically the product of two ciphertexts for $\Pi_+$. $\mathsf{2Mul}_+$ is defined more formally in Section 2.3. For our instantiation we provide a non-interactive, more efficient version over $\mathbf{Z}/p\mathbf{Z}$ (see Section 5).

**Definition 2 (Multiplicatively homomorphic encryption).** *Let $(\mathcal{M}, \times, 1_{\mathcal{M}})$ be a monoid. A* multiplicatively homomorphic encryption scheme *over the message space $\mathcal{M}$ is $\Pi_\times = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt}, \mathsf{Hom}_\times, \mathsf{ScalMul})$ such that:*

$\mathsf{Setup}$, $\mathsf{KeyGen}$, $\mathsf{Encrypt}$ *and* $\mathsf{Decrypt}$ *as in Definition 1 except that* $\mathsf{Encrypt}$ *and* $\mathsf{Decrypt}$ *receives the input messages from $\mathcal{M}$ instead of $\mathcal{R}$.*

$\mathsf{Hom}_\times$ *is a PPT algorithm which takes as input some public parameters, a public key $pk$ and two ciphertexts $c$ and $c'$ of $m \in \mathcal{M}$ and $m' \in \mathcal{M}$ respectively, and outputs a ciphertext $c''$ such that $\Pi_\times.\mathsf{Decrypt}(sk, c'') = m \times m' \in \mathcal{M}$.*

$\mathsf{ScalMul}$ *is a PPT algorithm which takes as inputs some public parameters, a public key $pk$, a ciphertext $c$ of $m \in \mathcal{M}$ and a plaintext $\alpha \in \mathcal{M}$, and outputs a ciphertext $c'$ such that $\Pi_\times.\mathsf{Decrypt}(sk, c') = \alpha \times m \in \mathcal{M}$.*

*Remark 2.* A generic algorithm for computing $\Pi_\times.\mathsf{ScalMul}(pk, c, \alpha)$ is given by $\Pi_\times.\mathsf{Hom}_\times(pk, c, c')$, where $c' = \Pi_\times.\mathsf{Encrypt}(pk, \alpha)$. In Section 5, we provide a more efficient version over $(\mathbf{Z}/p\mathbf{Z})^*$.

The above encryption schemes must be correct in the usual sense. Moreover, we consider as a security requirement the indistinguishability under a chosen plaintext attack (IND-CPA). We refer the reader to e.g, [25] for the standard definition of IND-CPA.

### 2.2 One round 2-Party Decryption.

A crucial feature of the encryption protocols which are used in the ESP is the fact that they support a 2-party decryption (threshold cryptosystems were introduced in [17]). These encryption schemes are equipped with a $\mathsf{Share}$ procedure that is run by a trusted dealer, which works as follows: it takes as input a pair of keys $(sk, pk)$ obtained from the $\mathsf{KeyGen}$ algorithm and produces two shares $sk_A$ and $sk_B$ of the secret key $sk$. It outputs $(sk_A, sk_B)$ and an updated public part still denoted $pk$. Its decryption procedure is an interactive protocol denoted $\mathsf{2Dec}$ which takes as inputs the public parameters, a ciphertext $c$, and the secret key of each participant $sk_A$ and $sk_B$ and outputs a plaintext $m$ which would have been obtained as $\mathsf{Decrypt}(sk, c)$.

Contrary to the classical definition of threshold decryption, we suppose that the protocol is in a *single* round. The protocol $\mathsf{2Dec}(pk, c; sk_A; sk_B)$ is supposed as follows: Alice starts the protocol and sends her information in one flow to

Bob which ends the computation and gets the plaintext. This is because in our context, we do not decrypt plaintexts but plaintexts which are masked by a random element. For example, protocols whose decryption only performs exponentiations with secret exponents gives one round 2-party decryption by sharing the exponentiations. This is the case for many cryptosystems.

The semantic security is adapted from the standard IND-CPA notion by giving the adversary one of the two secret keys, as well as a share decryption oracle which simulate the party with the other secret key. A formal definition can be found for instance in [41,11].

We need as an additional security requirement the notion of zero-knowledge defined in Appendix A, which means that no information on the secret keys are leaked during an interaction with a curious adversary. Cryptosystems like Elgamal [17] or Paillier [19] satisfy all these properties. We will propose a variant of Elgamal and a variant of Castagnos-Laguillaumie [9] that satisfy also these properties in Section 5.

### 2.3 Homomorphically computing a product with $\Pi_+$.

A core routine of our protocol is the computation of a $\Pi_+$-encryption of a product $XY$ given $\Pi_+$-encryptions of $X$ and $Y$ (this is why we assume that $\Pi_+$ has a ring $\mathcal{R}$ as message space). We describe in Fig. 1 a protocol which is implicitly used in [10]. It is a simplified variant of a protocol proposed by Cramer, Damgård and Nielsen from [11]: the main difference comes from the fact that the result of this 2-party computation is obtained only by one of the user, who can forward the result to the other. This leads to the use of a single randomness on Alice's side, instead of one on each side. We will denote by $\mathsf{2Mul}_+(pk, C_X^+, C_Y^+; sk_A; sk_B)$ a call to this protocol. Again, this protocol will be a 2-round protocol since the shared decryption is single round, and the first ciphertext can be sent along with the shared decryption. This protocol has to be zero-knowledge in the sense similar to those of Def. 5 and 7 (we do not write down this definition which can be readily adapted).

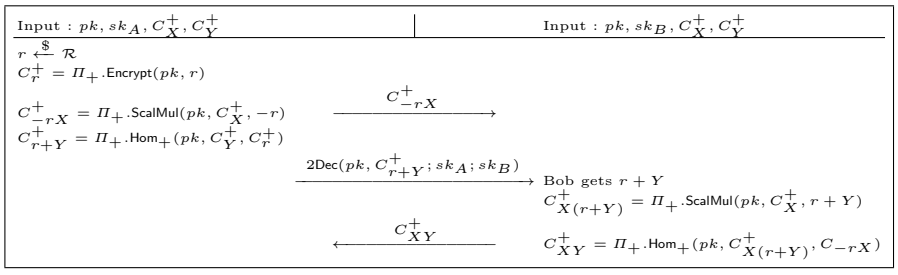


**Fig. 1.** $\mathsf{2Mul}_+$: 2-party protocol to compute $C_{XY}^+$ from $C_X^+$ and $C_Y^+$

**Theorem 1.** *Let $\Pi_+$ be an additively homomorphic encryption scheme with a zero-knowledge one round 2-party decryption. Then, the protocol described in Fig. 1 is correct and zero-knowledge.*

*Proof.* The correctness follows from the correctness of the encryption scheme and its homomorphic properties. Let us prove first that it is zero-knowledge for Alice. We describe a simulator $\mathsf{Sim}$ whose behavior is indistinguishable from Alice's behavior in front of an adversarial Bob. The simulator receives as input the public key $pk_+$ and will set $\mathsf{Sim}_{\mathsf{Share}}$ as follows: it calls out $\mathsf{Sim}^{2d}_{\mathsf{Share}}$ procedure of the zero-knowledge property of 2-party decryption for $\Pi_+$ with $pk_+$ as input. It obtains a simulated share $sk_B = x^+_B$ and feeds the adversary with it. When $\mathsf{Sim}$ is requested for the 2-party computation of $C^+_{XY}$ from $C^+_X$ and $C^+_Y$, it receives a pair of $((C^+_X, C^+_Y), \bar{C})$ where $\bar{C}$ is a ciphertext of $XY$, it does the following to simulate $C^+_{-rX}, C^+_{r+Y}$ and $C_A$:

- It picks $R$ at random in the plaintext space and sets $C^+_{r+Y} = \mathsf{Encrypt}(pk, R)$.
- Then it uses the simulator for the zero-knowledge for Alice of the 2-party decryption $\mathsf{Share}^{2d}_A(C^+_{r+Y}, R, x^+_B)$ so that Bob decrypts $R$ (which is equivalent to $\mathsf{Decrypt}(sk, C^+_{r+Y})$).
- Eventually, it sets $C_{-rX} = \mathsf{Hom}_+(\bar{C}, \mathsf{ScalMul}(C^+_X, -R))$. This ciphertext encrypts $XY - RX$ so that Bob's final $\mathsf{Hom}_+$ evaluation will cancel out the $RX$ part and lead to $\bar{C}$.

The simulated view is the same as a genuine one with $R = r + Y$, which means that they are indistinguishable, and the protocol is zero-knowledge for Alice. The protocol is obviously zero-knowledge for Bob: Bob's contribution is simulated by just sending $\bar{C}$. □

### 2.4 2-Party re-Encryption.

The final tool we need to build our encryption switching protocol is an interactive 2-party protocol to re-encrypt a ciphertext from an encryption scheme $\Pi_+$ intended to $pk$ into a ciphertext of the same encryption scheme of the same message, but intended to another key $pk'$. This protocol is depicted in Fig. 2. Note that the initial ciphertext to be transformed is not known to Bob. This protocol readily extends to the multiplicative case, which is useless for our purpose. With a proof similar to the proof of Theorem 1, we showed that

**Theorem 2.** *Let $\Pi_+$ be an additively homomorphic encryption scheme with a zero-knowledge one round 2-party decryption then the protocol described in Fig. 2 is correct and zero-knowledge.*

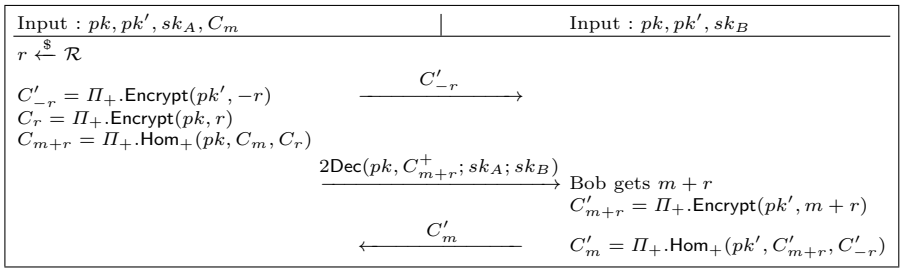


**Fig. 2.** 2-party $\mathsf{ReEnc}_+$

## 3 Encryption Switching Protocols

The global scenario is established as follows: two semantically secure threshold homomorphic encryption schemes, one additive, and the other multiplicative, are at the disposal of two players. A public key is provided for each protocols, and the matching secret key is shared among the players by a trusted dealer. Ideally, these two encryption schemes should have the same plaintext space, which is assumed to be a ring or a field. An encryption switching protocols makes it possible to interactively transform a ciphertext from a source encryption scheme into a ciphertext for the other encryption scheme (the target one) and *vice versa*. The formal definitions are given in the following paragraphs.

**Definition 3 (Twin ciphertexts).** *Let $\Pi_+$ and $\Pi_\times$ be two different encryption schemes with plaintext and ciphertext spaces respectively $\mathcal{M}_+, \mathcal{C}_+$ and $\mathcal{M}_\times, \mathcal{C}_\times$. If $C^+_m \in \mathcal{C}_+$ and $C^\times_m \in \mathcal{C}_\times$ are two encryptions of the same message $m \in \mathcal{M}_+ \cap \mathcal{M}_\times$, they are said to be* twin ciphertexts.

We will say that two ciphertexts from the same encryption scheme which decrypt to the same plaintext are *equivalent*.

**Definition 4 (Encryption Switching Protocols).** *An encryption switching protocol (ESP) between $\Pi_+$ and $\Pi_\times$, denoted $\Pi_+ \rightleftharpoons \Pi_\times$, is a protocol involving three parties: a trusted dealer $\mathcal{D}$ and two users $A$ and $B$. It uses common* Setup *and* KeyGen *algorithms to set the message space between $\Pi_+$ and $\Pi_\times$ and keys. It is a pair of interactive protocols* (Share, Switch) *defined as follows:*

- $\mathsf{Share}((pk_+, sk_+), (pk_\times, sk_\times)) \to (pk, sk_A, sk_B)$*: It is a protocol (run by $\mathcal{D}$) which takes as input two pairs of keys $(pk_+, sk_+)$ and $(pk_\times, sk_\times)$ produced from $\Pi_+.\mathsf{KeyGen}$, $\Pi_\times.\mathsf{KeyGen}$ and* Setup*. It outputs the shares $sk_A$ (sent to $A$) and $sk_B$ (sent to $B$) of $(sk_+, sk_\times)$ and updates the public key $pk$.*
- $\mathsf{Switch}_{\mathsf{way}}((pk, sk_A, c), (pk, sk_B, C)) \to C'$ *or* $\perp$*: It is an interactive protocol in the direction* $\mathsf{way} \in \{+ \to \times, \times \to +\}$ *which takes as common input the public key and a ciphertext $C$ under the source encryption scheme and as secret input the secret shares $sk_A$ and $sk_B$. The output is a twin ciphertext $C'$ of $C$ under the target encryption scheme or $\perp$ if the execution encountered problems.*

**Correctness** An encryption switching protocols $\Pi_+ \rightleftharpoons \Pi_\times$ is *correct* if for any $\lambda \in \mathbf{N}$, $(\mathsf{params}_+, \mathsf{params}_\times) \leftarrow \mathsf{Setup}(1^\lambda)$, for any pair of keys $(pk_+, sk_+) \leftarrow \Pi_+.\mathsf{KeyGen}(1^\lambda, \mathsf{params}_+)$ and $(pk_\times, sk_\times) \leftarrow \Pi_\times.\mathsf{KeyGen}(1^\lambda, \mathsf{params}_\times)$, for any shares $(pk, sk_A, sk_B) \leftarrow \mathsf{Share}((pk_+, sk_+), (pk_\times, sk_\times))$, for any twin ciphertext pair $(C_m^+, C_m^\times)$ of a message $m \in \mathcal{M}_+ \cap \mathcal{M}_\times$,

$$\Pi_+.\mathsf{Decrypt}(sk_+, \mathsf{Switch}_{\times\to+}((pk, sk_A, C_m^\times), (pk, sk_B, C_m^\times))) = m$$
$$\Pi_\times.\mathsf{Decrypt}(sk_\times, \mathsf{Switch}_{+\to\times}((pk, sk_A, C_m^+), (pk, sk_B, C_m^+))) = m.$$

**Zero-knowledge** An ESP has to satisfy a notion of zero-knowledge similar to the notion of zero-knowledge for threshold decryption (see def. 7). This property means that an adversary will not learn any other information on the secret share of a participant that he can learn from his own share, the input, and the output of the protocol.

**Definition 5.** *An encryption switching protocols $\Pi_+ \rightleftharpoons \Pi_\times$ is* zero-knowledge for $A$ *if there exists an efficient simulator* $\mathsf{Sim} = (\mathsf{Sim}_{\mathsf{Share}}, \mathsf{Sim}_A)$ *which simulates the sharing phase and the player $A$.*

*The subroutine* $\mathsf{Sim}_{\mathsf{Share}}$ *takes as input a public key $(pk_+, pk_\times)$ and outputs $(pk', sk'_B)$ that simulates the public key obtained from the* Share *algorithm and Bob's share of the secret key.*

*The subroutine* $\mathsf{Sim}_A$ *takes as input a direction* $\mathsf{way} \in \{+ \to \times, \times \to +\}$*, a source ciphertext $C$, a twin ciphertext $\bar{C}$ and a flow* flow*. It emulates the output of an honest player $A$ would answer upon receiving the flow* flow *when running the protocol* $\mathsf{Switch}_{\mathsf{way}}((pk, sk_A, C), (pk, sk_B, C))$ *without $sk_A$ but possibly $sk_B$, and forcing the output to be a ciphertext $C'$ which is equivalent to $\bar{C}$.*

*Then, for all $\lambda \in \mathbf{N}$, for any parameters* $(\mathsf{params}_+, \mathsf{params}_\times) \leftarrow \mathsf{Setup}(1^\lambda)$*, for any pairs of keys* $(pk_+, sk_+) \leftarrow \Pi_+.\mathsf{KeyGen}(1^\lambda, \mathsf{params}_+)$ *and* $(pk_\times, sk_\times) \leftarrow \Pi_\times.\mathsf{KeyGen}(1^\lambda, \mathsf{params}_\times)$*,* $(pk, sk_A, sk_B) \leftarrow \mathsf{Share}((pk_+, sk_+), (pk_\times, sk_\times))$ *or for any* simulated *share* $(pk', sk'_B) \leftarrow \mathsf{Sim}_{\mathsf{Share}}(pk)$*, and for any adversary $\mathcal{D}$ playing the role of $B$, the advantage*

$$\mathsf{Adv}^{zk}_{A,\Pi_+\rightleftharpoons\Pi_\times}(\mathcal{D}) = \left|\Pr[1 \leftarrow \mathcal{D}^A(pk, sk_B)] - \Pr[1 \leftarrow \mathcal{D}^{\mathsf{Sim}_A()}(pk', sk'_B)]\right|$$

*is negligible.*

*We define similarly an encryption switching protocols $\Pi_+ \rightleftharpoons \Pi_\times$ that is* zero-knowledge for $B$*. It is* zero-knowledge *if it is zero-knowledge for $A$ and $B$.*

## 4 Generic Construction of an ESP on a Ring

We describe in this section a generic construction of an encryption switching protocol in the semi-honest model. Even though an ESP could allow to switch between any encryption schemes, its main interest is when its implemented with homomorphic encryptions. Therefore, we start with an additively homomorphic

encryption $\Pi_+$ and a multiplicatively homomorphic encryption $\Pi_\times$ whose message space is respectively a ring $\mathcal{R}$ and a monoid $\mathcal{M}$. To fit most of the applications, we will make the assumption that $\mathcal{M} = \mathcal{R}^*$, the subgroup of invertible elements of $\mathcal{R}$, since in general the multiplicative homomorphic encryption will have a group as message space. In particular, this means that the intersection over which the switches are defined is $\mathcal{R} \cap \mathcal{M} = \mathcal{R}^*$.

As in [10], in the first place, we are going to describe how we can switch between $\Pi_+$-encryptions and $\Pi_\times$-encryptions over $\mathcal{R}^*$. Then we will show how to modify $\Pi_\times$ in order to extend its message space to $\mathcal{R}^* \cup \{0\}$.

**Definition 6 (Compatible encryption protocols).** *Let $(\mathcal{R}, +, \times)$ be a ring. Let $\Pi_+$ and $\Pi_\times$ be an additively and multiplicatively homomorphic encryption in the sense of Def. 1 and 2. They are said to be* compatible *if*

- *$\Pi_+$ and $\Pi_\times$ have respectively $\mathcal{R}$ and $\mathcal{R}^*$ as message space,*
- *both of them admit a one-round 2-party decryption as defined in Section 2.2,*
- *there exists a common setup algorithm* Setup *and common* KeyGen *which allows to set common parameters.*

*Remark 3.* To illustrate this, our instantiation (resp. Couteau et al.'s instantiation) switches between an additively homomorphic encryption whose message space is the field $(\mathbf{Z}/p\mathbf{Z}, +, \times)$ (resp. the ring $(\mathbf{Z}/N\mathbf{Z}, +, \times)$) and a multiplicative homomorphic encryption whose message space is the group $((\mathbf{Z}/p\mathbf{Z})^*, \times)$ (resp. $((\mathbf{Z}/N\mathbf{Z})^*, \times)$) and the former is modified so that its message space is the monoid $(\mathbf{Z}/p\mathbf{Z}, \times)$ (resp. $((\mathbf{Z}/N\mathbf{Z})^* \cup \{0\}, \times)$). In particular, Couteau et al.'s make the additional algorithmic assumption that $(\mathbf{Z}/N\mathbf{Z})^*$ is computationally equal to $\mathbf{Z}/N\mathbf{Z}$.

**Share Protocol of the ESP** The keys of $\Pi_+$ and $\Pi_\times$ are first shared by a trusted dealer, this corresponds to the Share algorithm from Def. 4. From public parameters generated using the common Setup algorithm and two pairs of keys $(pk_+, sk_+)$ and $(pk_\times, sk_\times)$ it outputs the secret share $sk_A = (sk_A^+, sk_A^\times)$ for Alice and $sk_B = (sk_B^+, sk_B^\times)$ for Bob using the Share procedures of the 2-party decryption of $\Pi_+$ and $\Pi_\times$.

### 4.1 Switching protocols over $\mathcal{R}^*$

We describe here the two 2-party switching protocols from an additive homomorphic encryption of $m$ to a multiplicative one and *vice versa*. Contrary to Couteau *et al.*'s protocol [10], the two protocols are actually the *same* since both the additive and the multiplicative scheme support a ScalMul operation and a *single-round* 2-party decryption. It is important to note that in our instantiation, the round complexity is only 2, since the first ciphertext $C^{(2)}_{R^{-1}}$ can be sent within the flow of the 2-party decryption which is only one round (cf. 2.2 or Fig. 9 and 11). We suppose that $m \neq 0$ here, and more precisely the message to be switched lies in $\mathcal{R} \cap \mathcal{M} = \mathcal{R}^*$.

**Switching protocols between $\Pi_1$.Encrypt($m$) and $\Pi_2$.Encrypt($m$)** In Fig. 3, as our switching protocols in the two directions are the same, the pair $(\Pi_1, \Pi_2)$ is either $(\Pi_+, \Pi_\times)$ or $(\Pi_\times, \Pi_+)$.

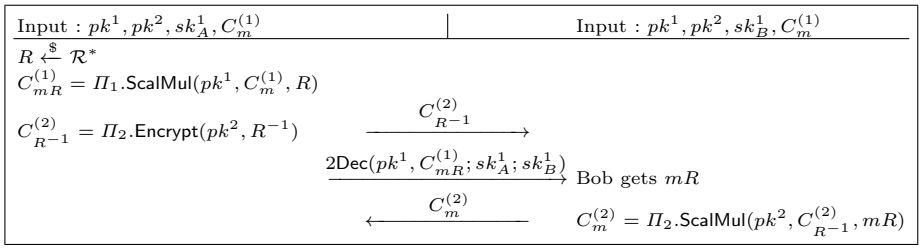


**Fig. 3.** 2-party $\mathsf{Switch}_{1\to 2}$ from $\Pi_1$ to $\Pi_2$ where $(\Pi_1, \Pi_2)$ is either $(\Pi_+, \Pi_\times)$ or $(\Pi_\times, \Pi_+)$.

The correctness of these two protocols is clear. They are generic and and the switch from $\Pi_\times$ to $\Pi_+$ is highly simpler than the one in [10] (ours is 2-round instead of 6-round) and our instantiation will keep this simplicity. We prove in the following theorem that they are zero-knowledge, and the security proof itself is also very simple. It only relies on the zero-knowledge property of the shared decryptions.

**Theorem 3.** *The ESP between $\Pi_+$ and $\Pi_\times$, whose switching routines are described in Figure 3, is zero-knowledge if $\Pi_+$ and $\Pi_\times$ are two compatible encryption schemes which have zero-knowledge one round 2-party decryptions.*

*Proof.* The proof consists in proving that after a share of the secret keys, both switching procedures are zero-knowledge for Alice and Bob. Let us start with the proof that the ESP is zero-knowledge for Alice. We are going to describe a simulator $\mathsf{Sim}$ whose behavior is indistinguishable from Alice's behavior in front of an adversarial Bob.

$\mathsf{Sim}_{\mathsf{Share}}$: The simulator receives as input the public key $(pk_+, pk_\times)$ and simulates the $\mathsf{Share}$ procedure as follows: it calls out the $\mathsf{Sim}_{\mathsf{Share}}^{2d}$ procedures of the zero-knowledge property of Alice for 2-party decryption of respectively $\Pi_+$ and $\Pi_\times$ with $pk_+$ and $pk_\times$ as input. In particular it obtains $sk'_B = (x_B^+, x_B^\times)$ it can feed the adversary with. When $\mathsf{Sim}$ is requested for a switch, it receives a pair of twin ciphertexts $(C, \bar{C})$.

**Game $G_0$.** This game is the real game. The simulator generates all the secret shares in an honest way and gives his share to Bob. It plays honestly any switching protocols on an input $(C, \bar{C})$ using Alice's secret key.

**Game $G_1$.** The first modification concerns the *additive to multiplicative* direction. The setup and key generation are the same as in the previous game.

When requested to participate to a $\mathsf{Switch}_{+\to\times}$, with $(C, \bar{C})$ as input, the simulator picks uniformally at random $x \in \mathcal{R}^*$ and sets $C^+_{mR} = \Pi_+.\mathsf{Encrypt}(x)$ and $C^\times_{R^{-1}} = \Pi_\times.\mathsf{ScalMul}(\bar{C}, x^{-1})$. The simulator then concludes the protocol honestly. This game is indistinguishable from the previous since, as $x$ is random, it is equivalent to a genuine protocol using $R = x/m$, where $m$ is the plaintext under $C$ and $\bar{C}$.

**Game $G_2$.** In this game, we modify the shared decryption for $\Pi_+$ using the simulator of 2-party decryption. First, the simulation gives the key $x^+_B$ obtained by the simulation of the shares to Bob. Then after $\mathsf{Sim}$ simulated the pair $(C^+_{mR}, C^\times_{R^{-1}})$ as above, it uses the simulator $\mathsf{Sim}^{2d}_A(C^+_{mR}, x, x^+_B, \cdot)$ for the 2-party decryption of $\Pi_+$ to interact with Bob, where $C^+_{mR}$ is an encryption of $x$. Thanks to the property of this simulator this game is indistinguishable from the previous one (note that the key $x^+_B$ is only used in that part of the protocol). Eventually, the last computation done by Bob, $\Pi_\times.\mathsf{ScalMul}(C^\times_{R^{-1}}, x)$ gives a multiplicative ciphertext of $m$ equivalent to $\bar{C}$.

**Game $G_3$.** In this game, we address the *multiplicative to additive* way. The setup and key generation are the same as in the previous games. As in Game $G_1$, when requested to participate to a $\mathsf{Switch}_{\times\to+}$, with $(C, \bar{C})$ as input, the simulator picks uniformally at random $x \in \mathcal{R}^*$ and sets $C^\times_{mR} = \Pi_\times.\mathsf{Encrypt}(x)$ and $C^+_{R^{-1}} = \Pi_+.\mathsf{ScalMul}(\bar{C}, x^{-1})$. Then, $\mathsf{Sim}$ continues honestly the protocol. This game is indistinguishable from the previous one.

**Game $G_4$.** The shared $\Pi_\times$ decryption is modified as in Game $G_2$. The simulation now gives the simulated key $x^\times_B$ to Bob and then uses the simulator for the 2-party decryption of $\Pi_\times$ with ciphertext $C^\times_{mR}$ and corresponding plaintext $x$. Thanks to the property of this simulator this game is indistinguishable from the previous one. Again, Bob's last computation $\Pi_+.\mathsf{ScalMul}(C^+_{R^{-1}}, x)$ gives a ciphertext equivalent to $\bar{C}$.
In conclusion, the advantage of the attacker is negligible.

We now prove that the ESP is zero-knowledge for Bob, by describing a simulator $\mathsf{Sim}$ whose behavior is indistinguishable from Bob's behavior in front of an adversarial Alice. The simulator receives as input the public key $pk = (pk_+, pk_\times)$ and simulates the $\mathsf{Share}$ procedure as above and feed the adversary (Alice) with the corresponding secret key. When $\mathsf{Sim}$ is requested for a switch, it receives a pair of twin ciphertexts $(C, \bar{C})$.

In both directions, the simulation is trivial, since Bob's only flow is the final forward of the twin ciphertext (we have suppose that the 2-party decryption has only one round from Alice to Bob), which is done by sending the $\bar{C}$ ciphertext. This is indistinguishable from a true execution since $\bar{C}$ is a random ciphertext which encrypts the same plaintext that $C$. □

### 4.2 Modification of $\Pi_\times$ to embed the zero message

One technical issue to design switching protocols between $\Pi_+$ and $\Pi_\times$ is to embed the zero message into $\Pi_\times$'s message space so that the message spaces

match. To do so, we need to modify the $\Pi_\times$ encryption. We will use a technique quite similar to those in [10]: During their encryption, if the message $m$ is equal to 0, a bit $b$ is set to 1. It is set to 0 for any other message. Then, the message $m + b$ (which is never 0) is encrypted using their Elgamal encryption. As this encryption scheme is no longer injective, to discriminate an encryption of 0, the ciphertext is accompanied by two encryptions under classical Elgamal of $T^b$ and $T'^b$ where $T, T'$ are random elements. We note that these two encryptions are in fact encryptions of $b$ which are homomorphic for the or gate : If $b = 0$, we get an Elgamal encryption of 1 and if $b = 1$, an Elgamal encryption of a random element (which is equal to 1 only with negligible probability). Thanks to the multiplicativity of Elgamal, if we multiply an encryption of $b$ and an encryption of $b'$, we get an Elgamal encryption of 1 only if $b = b' = 0$ and an Elgamal encryption of a random element otherwise. In [10] the second encryption of $b$ is actually an extractable commitment, the corresponding secret key is only known by the simulator in the security proof.

In our case, we use the additively homomorphic encryption $\Pi_+$ to discriminate the zero message: $\Pi_+$ is used to encrypt a random element $r$ if $m = 0$ and 0 otherwise. As a consequence, it will be possible to directly obtain an encryption of $\bar{b}$ (the complement of the bit $b$ used during encryption) under $\Pi_+$ using the zero-testing procedure during the switch from $\Pi_\times^0$ to $\Pi_+$ (see Fig. 7). This gives a real improvement compared to [10] when we instantiate our generic protocols. As in [10] we add a useless second encryption of $r$ to be used by the simulation in the security proof. Our modification is formally described in Fig. 4. The $\mathsf{Hom}_\times$ procedure is obtained by applying the $\mathsf{Hom}_\times$ procedures of $\Pi_\times$, and $\Pi_+$. For the $\mathsf{ScalMul}$ procedure, which corresponds to a multiplication by a plaintext $\alpha$, it applies the $\mathsf{ScalMul}$ procedure of $\Pi_\times$ if $\alpha \neq 0$ and add an encryption of 0 to the additive part. If $\alpha = 0$, it outputs an encryption of 0.

The protocol $\Pi_\times^0$ directly inherits the indistinguishability under a chosen message attack from those of $\Pi_\times$ and $\Pi_+$. By a standard hybrid argument, we can prove the following theorem, whose proof is omitted.

**Theorem 4.** *If $\Pi_+$ and $\Pi_\times$ are IND-CPA, then $\Pi_\times^0$ is also IND-CPA.*

### 4.3 Full Switching protocols

**Encrypted zero-test** In [10] an encrypted zero test ($\mathsf{EZT}$) to obliviously detect the zero messages during switches is presented. In our case, $\mathsf{EZT}$ takes as input a ciphertext $C_m^+$ from the additively homomorphic encryption $\Pi_+$ of a message $m$ and outputs a $\Pi_+$ ciphertext $C_b^+$ of a bit $b$ equals to 1 if $m = 0$ and equals to 0 otherwise. The $\mathsf{EZT}$ has to be zero-knowledge in the sense that there exists an efficient simulator for each player which, on input a pair of twin ciphertext $(C, \bar{C})$, is indistinguishable from these honest players. This simulator runs without the secret share of the the user it simulates.

During the security proof, the simulator will obtain the bit $b$ thanks to the knowledge of the secret key which decrypts the additional encryption of $r$ appended during encryption (see Fig. 4).

**Algo.** $(\Pi_+ \rightleftharpoons \Pi^0_\times).\mathsf{KeyGen}(1^\lambda)$

1. $\mathsf{params} \leftarrow (\Pi_+ \rightleftharpoons \Pi_\times).\mathsf{Setup}(1^\lambda)$
2. $((pk^\times, sk^\times), (pk^+, sk^+)) \leftarrow (\Pi_+ \rightleftharpoons \Pi_\times).\mathsf{KeyGen}(1^\lambda, \mathsf{params})$
3. $(pk', sk') \leftarrow \Pi_+.\mathsf{KeyGen}(1^\lambda, \mathsf{params})$
4. Set $pk \leftarrow (pk^\times, pk^+, pk')$ and $sk \leftarrow (sk^\times, sk^+, sk')$
5. Return $(pk, sk)$

**Algo.** $\Pi^0_\times.\mathsf{Encrypt}(pk, m)$

1. Parse $pk$ as $(pk^\times, pk^+, pk')$
2. If $m = 0$ set $b \leftarrow 1$ and $r \xleftarrow{\$} \mathcal{R}^*$ otherwise set $b \leftarrow 0$ and $r \leftarrow 0$
3. $C^\times_{m+b} \leftarrow \Pi_\times.\mathsf{Encrypt}(pk^\times, m + b)$
4. $C^+_r \leftarrow \Pi_+.\mathsf{Encrypt}(pk^+, r)$
5. $C'_r \leftarrow \Pi_+.\mathsf{Encrypt}(pk', r)$
6. Return $(C^\times_{m+b}, C^+_r, C'_r)$

**Algo.** $\Pi^0_\times.\mathsf{Decrypt}(sk, (C^\times_{m+b}, C^+_r, C'_r))$

1. Parse $sk$ as $(sk^\times, sk^+, sk')$
2. $B \leftarrow \Pi_+.\mathsf{Decrypt}(sk^+, C^+_r)$
3. If $B \neq 0$ return 0 else return $\Pi_\times.\mathsf{Decrypt}(sk^\times, C^\times_{m+b})$

**Fig. 4.** $\Pi_\times$ over $\mathcal{R}$ : $\Pi^0_\times$

This EZT protocol is done using garbled circuits techniques. An alternative would be to use techniques based on homomorphic encryption [32]. The resulting protocol is described in Fig. 5. The function $H : \mathcal{R}^* \longrightarrow \{0,1\}^\kappa$ (for a security parameter $\kappa$) belongs to a universal hash function family (in practice, this will be a reduction modulo $2^\kappa$ of the integer representation of an element of $\mathcal{R}$). We denote by $\mathsf{eq}$ the function that on input $(u, v) \in \{0,1\}^\kappa$ outputs 1 if $u = v$ and 0 otherwise and we denote by $\mathsf{Garble}(f)$ the computation of a garbled circuit evaluating the function $f$.

The correctness of the protocol comes from the fact that the last three lines of the protocol compute the encryption of $b_A \oplus b_B$ by homomorphically evaluating $b_A + b_B - 2b_Ab_B$ from the encryptions of $b_A$ and $b_B$. By construction, $b_A \oplus b_B = \mathsf{eq}(r', r'')$ which is equals to 1 if $m = 0$ and 0 otherwise, with probability $1 - 2^{-\kappa}$. This is exactly the encryption of the bit $b$. This protocol is zero-knowledge (see [10]). Using [28], the communication needed is $8\kappa^2$ bits of preprocessing for the garbled circuit and $\kappa^2$ bits and $\kappa$ oblivious transfers for the online phase (cf. [10, Fig. 4]).

### 4.4 2-party ESP between $\Pi_+.\mathsf{Encrypt}(m)$ and $\Pi^0_\times.\mathsf{Encrypt}(m)$

The protocol of Fig. 6 is quite similar to the one of [10]. First we use the EZT sub-protocol to get a $\Pi_+$ encryption of the bit $b$. A notable difference with the

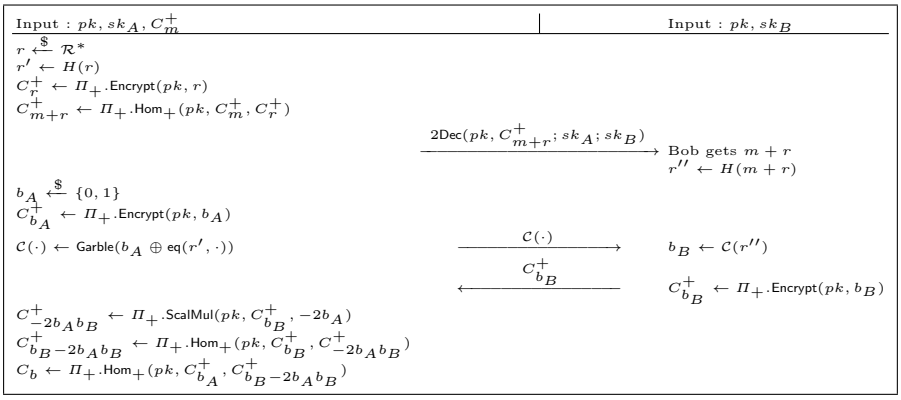


**Fig. 5.** EZT: 2-party protocol to compute $C^+_b$ from $C^+_m$

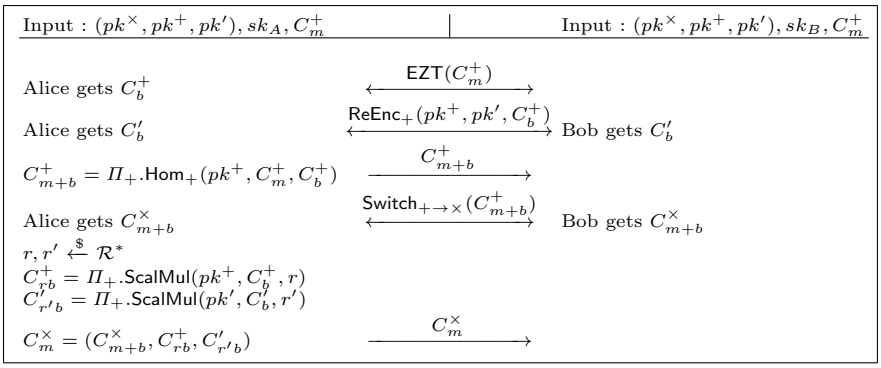


**Fig. 6.** 2-party $\mathsf{Switch}_{+\to\times}$ from $\Pi_+$ to $\Pi^0_\times$ over $\mathcal{R}^* \cup \{0\}$

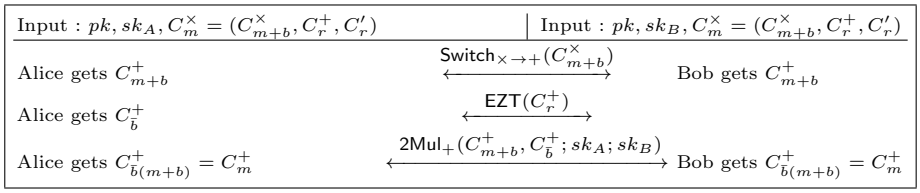


**Fig. 7.** 2-party $\mathsf{Switch}_{\times\to+}$ from $\Pi^0_\times$ to $\Pi_+$ over $\mathcal{R}^* \cup \{0\}$

protocol of [10] is that this encryption of $b$ can be used directly to set an element of the ciphertext for $\Pi^0_+$, which saves many rounds in the interaction. Since the bit $b$ is encrypted twice (this second encryption is only used during the security proof), the $\mathsf{ReEnc}_+$ protocols allows to re-encrypt the output of $\mathsf{EZT}$ to the right public key. Then, thanks to the homomorphic property of the $\Pi_+$ scheme, Alice can construct an additive encryption of $m + b$ and the $\mathsf{Switch}_{+\to\times}$ protocol of Fig. 3 is used to get the $\Pi_+$-encryption of $m + b$. The two ciphertexts of $b$ are randomized to get a proper multiplicative ciphertext.

In Fig. 7, starting from a multiplicative ciphertext of $m$, we run an $\mathsf{Switch}_{\times\to+}$ with the first component of $C^\times_m$, which is a $\Pi_\times$-encryption of $m + b$. Hence, we get $C^+_{m+b}$. Then, we run the $\mathsf{EZT}$ protocol on the second component $C^+_r$ and the output the encryption of a bit $b'$ whose value is 1 when $r = 0$, i.e., when $b = 0$ and 0 otherwise. Therefore $b' = \bar{b}$ and $\mathsf{EZT}$ actually outputs an encryption of $\bar{b}$. It is now possible to homomorphically remove the bit $b$ remaining in the $\Pi_+$-encryption of $m + b$, $C^+_{m+b}$. Inspired by the implicit technique used in [10], we use the $\mathsf{2Mul}_+$ protocol to obtain, from $C^+_{\bar{b}}$ and $C^+_{m+b}$, a $\Pi_+$-encryption of $(m + b)\bar{b}$ which is equal to a $\Pi_+$-encryption of $m$.

Note that we can not simply use the fact that $m + b + \bar{b} - 1 = m$ over $\mathbf{Z}$ to get $m$: The expression $m + b$ is really equal to the message $m$ plus the bit $b$ only for fresh multiplicative ciphertexts. After an homomorphic multiplication between a ciphertext of a non zero message with a ciphertext of zero it becomes something random. As a result, we have to multiply it by $\bar{b}$ to get 0.

The zero-knowledge property of our ESP essentially comes from the fact that each routine ($\mathsf{ReEnc}_+$ and $\mathsf{2Mul}_+$) is individually zero-knowledge, inherited from the zero-knowledge of the 2-party decryption of the encryption protocols. We also use the fact that the encryption schemes are IND-CPA, in order to be able to simulate intermediate ciphertexts. This means that the assumptions in our theorem are weak and very natural.

**Theorem 5.** *The ESP between $\Pi_+$ and $\Pi^0_\times$ whose routines are described in Fig. 6 and 7 is zero-knowledge if $\Pi_+$ and $\Pi_\times$ are two compatible encryptions that are IND-CPA and whose 2-party decryptions are zero-knowledge, and* $\mathsf{EZT}$ *is zero-knowledge.*

*Proof.* (sketch) The full proof of this theorem can be found in Appendix B. This proof can be sketched as follows: First we give the secret key $sk'$ to the simulation. From a pair of twin ciphertexts, it allows the simulation to know the bit that encode the fact that the plaintext is 0 or not. With that knowledge, the simulation can retrieve the ciphertexts that constitute the input and the output of each building block, and use their zero-knowledge simulator to emulate them. Then, we remove the knowledge of $sk'$ from the simulation which replaces each input and output by random ciphertexts. Thanks to the IND-CPA property of the encryption schemes this is indistinguishable from the previous step. As a result, the whole protocol is simulated without knowing any secret. □

## 5 Instantiation of our Generic Construction on $\mathbf{Z}/p\mathbf{Z}$

In this section we provide an instantiation of our generic construction on a field $\mathbf{Z}/p\mathbf{Z}$ for a prime $p$, by describing an additively homomorphic encryption and a multiplicatively homomorphic one. Both schemes enjoy an Elgamal structure. For the additively homomorphic encryption scheme, we will use as a basis the scheme introduced in [9] (denoted CL in the following). It uses the notion of a DDH *group with an easy* DL *subgroup*, which is instantiated using class groups of quadratic fields. For the multiplicatively homomorphic scheme, we devise a variant of the traditional Elgamal encryption over the whole group $(\mathbf{Z}/p\mathbf{Z})^*$. Both schemes are described in the next subsection. We also describe their 2-party decryption, since it is required by the generic construction.

### 5.1 Additively Homomorphic Scheme over $\mathbf{Z}/p\mathbf{Z}$

**Castagnos-Laguillaumie encryption** The encryption scheme from [9] is additively homomorphic modulo a prime $p$. The general protocol is well suited for relatively small $p$. For the ESP context, we need a large message space as $p$ must be at least of 2048 bits for the security of the Elgamal protocol. As a result, we use the first variant of CL described in [9, Section 4]. This variant is defined with subgroups of the class group of an order of a quadratic field of discriminant $\Delta_p = -p^3$. Thus all computations are done in this class group. Note that elements are classes of ideals, that can be represented by their unique reduced elements, i.e., by two integers of roughly the size of $\sqrt{|\Delta_p|}$. As a consequence, a group element can be represented with $3 \log p$ bits.

We provide some improvements detailed in the following. The CL scheme does exponentiations to some random powers in a cyclic group of unknown order. Let us denote by $\mathfrak{g}$ a generator of this group. Only an upper bound $B$ on this order is known. In order to make the result of these exponentiations look like uniform elements of the cyclic group, the authors of [9] choose to sample random exponents from a large enough uniform distribution, and more precisely over $\{0, \ldots, B'\}$ where $B' = 2^{\lambda-2}B$, so that the resulting distribution is as distance to uniform less than $2^{-\lambda}$.

However, it is more efficient to use a folded discrete Gaussian Distribution instead of a folded uniform distribution. Let $z \in \mathbf{Z}$ and $\sigma > 0$ a real number and let us denote by $\rho_\sigma(z) = \exp(-\pi z^2/\sigma^2)$ a Gaussian centered function and define the probability mass function $\mathcal{D}_\sigma$ over $\mathbf{Z}$ by $\mathcal{D}_\sigma(z) := \rho_\sigma(z)/\sum_{z\in\mathbf{Z}} \rho_\sigma(z)$.

If $z$ is sampled from $\mathcal{D}_\sigma$, we have $|z| > \tau\sigma$ with probability smaller than $\sqrt{2\pi e}\tau \exp(-\pi\tau^2)$ (cf. [35, Lemma 2.10]). We denote by $\tau(\lambda)$ the smallest $\tau$ such that this probability is smaller than $2^{-\lambda}$.

If we set $\sigma = \sqrt{\ln(2(1+2^{\lambda+1}))/\pi}B$, Lemma 1 of Appendix C shows that the distribution obtained by sampling $z$ from $\mathcal{D}_\sigma$ and computing $\mathfrak{g}^z$ is at distance less than $2^{-\lambda}$ to the uniform distribution in $\langle\mathfrak{g}\rangle$.

For instance for $\lambda = 128$, we only add, in the worst case, 6 iterations in the square and multiply algorithm to compute $\mathfrak{g}^z$, whereas one has to add 126 iterations with a folded uniform distribution.

*Description of the scheme.* We denote by $\mathsf{CL.Gen}$ a parameter generator for $\mathsf{CL}$. It takes as input $1^\lambda$ and outputs a tuple $(p, \mathfrak{g}, \mathfrak{f}, \sigma)$. This tuple is such that $p$ is a prime satisfying $p \equiv 3 \pmod 4$ so that computing discrete logarithms in $C(-p)$, the ideal class group of the quadratic order of discriminant $-p$, takes $2^\lambda$ times. Then $\mathfrak{g} \in C(-p^3)$ is a class of order $ps$ where $s$ is unknown and expected to be of the order of magnitude of the class number of $C(-p)$: a concrete implementation for $\mathfrak{g}$ is given in [9, Fig. 2]. It consists in generating a random ideal of the maximal order of discriminant $-p$, and lifting it in the order of discriminant $-p^3$. Eventually, $\mathfrak{f} \in C(-p^3)$ is the class of the ideal $p^2\mathbf{Z} + ((-p + \sqrt{-p^3})/2)\mathbf{Z}$ and $\sigma$ will be the standard deviation of the Gaussian Distribution discussed before: $\sigma = \sqrt{\ln(2(1+2^{\lambda+1}))/\pi}B$, with $B = \log(p)p^{3/2}/(4\pi)$.

The scheme relies on the notion of a $\mathsf{DDH}$ group with an easy $\mathsf{DL}$ subgroup. It is IND-CPA under the $\mathsf{DDH}$ assumption in the group generated by $\mathfrak{g}$. On the other hand, in the subgroup of order $p$ generated by $\mathfrak{f}$, there is a polynomial time algorithm, denoted $\mathsf{CL.Solve}$ which takes as input an element of $\langle \mathfrak{f} \rangle$ and which outputs its discrete logarithm in basis $\mathfrak{f}$. We refer the reader to [9] for concrete implementation of $\mathsf{CL.Gen}$ and $\mathsf{CL.Solve}$. The resulted scheme is given in Fig. 8.

$\underline{\mathsf{CL.Setup}(1^\lambda)}$

1. $(p, \mathfrak{g}, \mathfrak{f}, \sigma) \leftarrow \mathsf{CL.Gen}(1^\lambda)$
2. Return $\mathsf{params} := (p, \mathfrak{g}, \mathfrak{f}, \sigma)$

$\underline{\mathsf{CL.KeyGen}}$

1. Pick $x \xleftarrow{\$} \mathcal{D}_\sigma$ and set $\mathfrak{h} \leftarrow \mathfrak{g}^x$
2. Set $pk \leftarrow \mathfrak{h}$ and set $sk \leftarrow x$.
3. Return $(pk, sk)$

$\underline{\mathsf{CL.Decrypt}(sk, (\mathfrak{c}_1, \mathfrak{c}_2))}$

1. Set $\mathfrak{M} \leftarrow \mathfrak{c}_2/\mathfrak{c}_1^x$
2. $m \leftarrow \mathsf{CL.Solve}(\mathfrak{M})$
3. Return $m$

$\underline{\mathsf{CL.Encrypt}(pk, m)}$

1. Pick $r \xleftarrow{\$} \mathcal{D}_\sigma$
2. Compute $\mathfrak{c}_1 \leftarrow \mathfrak{g}^r$
3. Compute $\mathfrak{c}_2 \leftarrow \mathfrak{f}^m\mathfrak{h}^r$
4. Return $(\mathfrak{c}_1, \mathfrak{c}_2)$

$\underline{\mathsf{CL.Hom}_+(pk, (\mathfrak{c}_1, \mathfrak{c}_2), (\mathfrak{c}'_1, \mathfrak{c}'_2))}$

1. Pick $r \xleftarrow{\$} \mathcal{D}_\sigma$
2. Return $(\mathfrak{c}_1\mathfrak{c}'_1\mathfrak{g}^r, \mathfrak{c}_2\mathfrak{c}'_2\mathfrak{h}^r)$

$\underline{\mathsf{CL.ScalMul}(pk, (\mathfrak{c}_1, \mathfrak{c}_2), \alpha)}$

1. Pick $r \xleftarrow{\$} \mathcal{D}_\sigma$
2. Return $(\mathfrak{c}_1^\alpha\mathfrak{g}^r, \mathfrak{c}_2^\alpha\mathfrak{h}^r)$

**Fig. 8.** Castagnos-Laguillaumie over $\mathbf{Z}/p\mathbf{Z}$: $\mathsf{CL}$

**Theorem 6 ([9]).** *The* $\mathsf{CL}$ *scheme of Fig. 8 is an additively homomorphic encryption scheme over* $\mathbf{Z}/p\mathbf{Z}$*, IND-CPA under the* $\mathsf{DDH}$ *assumption in the ideal class group of the quadratic order of discriminant* $-p^3$*.*

**One round 2-party decryption for CL** We now devise in Fig. 9 a one round 2-party decryption for $\mathsf{CL}$ as defined in Section 2.2, i.e. subroutines to share

the secret key and the interactive protocol for decryption. As the scheme has an Elgamal structure, it can be readily adapted from the threshold variant of the original Elgamal scheme (cf. [39] for instance) with a simple additive secret sharing of the key $x = x_A + x_B$. However, as the group order is unknown, this secret sharing must be done over the integers. This kind of sharing has been addressed before (cf. [14, Section 4] for instance).

As $x$ is sampled from $\mathcal{D}_\sigma$, we saw before that $x \in [-\tau(\lambda)\sigma, \tau(\lambda)\sigma]$ for a small $\tau(\lambda)$ except with negligible probability. Then the integer $x_A$ is taken uniformly at random in the interval $[-\tau(\lambda)\sigma 2^\lambda, \tau(\lambda)\sigma 2^\lambda]$, and $x_B = x - x_A$. This choice makes the secret sharing private. Note that in that case, there is no gain in using a Gaussian Distribution to generate the shares. We refer the interested reader to Appendix D for details.

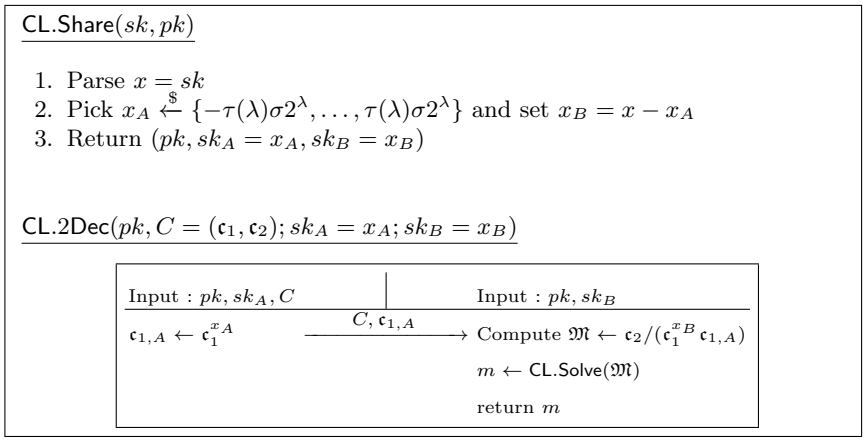


**Fig. 9.** 2-party Decryption for CL

**Theorem 7.** *The 2-party Decryption for* CL *described in Fig. 9 is correct and zero-knowledge.*

*Proof.* Correctness follows from the shared exponentiation. Let us prove first that the protocol is zero-knowledge for Alice (see Def. 7 in Appendix A).

For the secret key shares, the simulator $\mathsf{Sim}^{2d}_{\mathsf{Share}}$ picks $x'$ from $\mathcal{D}_\sigma$, $x'_A \xleftarrow{\$} \{-\tau(\lambda)\sigma 2^\lambda, \ldots, \tau(\lambda)\sigma 2^\lambda\}$ and set $x'_B = x' - x'_A$. As the secret sharing is private, the distribution of $x'_B$ is statistically indistinguishable from the distribution of the real $x_B$ (see Appendix D for the computation of the statistical distance).

Then we describe the simulator $\mathsf{Sim}^{2d}_A$ which emulates Alice. From a ciphertext $C$, a plaintext $m$, it computes $\mathfrak{M} = \mathfrak{f}^m$. Then it simulates $\mathfrak{c}_{1,A}$ by setting $\mathfrak{c}_{1,A} = \mathfrak{c}_2/(\mathfrak{M}\mathfrak{c}_1^{x'_B})$, so that Bob's computations leads to $\mathfrak{M}$. The value sent by the simulation is thus perfectly indistinguishable from the real one.

It is straightforward to see that the protocol is zero-knowledge for Bob: secret key shares are simulated as previously, and $x'_A$ is obviously indistinguishable from the real $x_A$, and then Bob sends nothing during the protocol. □

### 5.2 Multiplicatively Homomorphic Scheme over $\mathbf{Z}/p\mathbf{Z}$

**Elgamal over $(\mathbf{Z}/p\mathbf{Z})^*$** Let $q$ be an odd Sophie Germain prime, and let us denote by $p$ the associated prime, *i.e.*, $p = 2q + 1$. The DDH assumption is widely supposed to hold in the subgroup of order $q$ of $(\mathbf{Z}/p\mathbf{Z})^*$ which is the subgroup of quadratic residues modulo $p$, denoted $S_p$. The Elgamal cryptosystem defined in $S_p$ is multiplicatively homomorphic and semantically secure if the DDH assumption holds in that subgroup.

It is well-known that the DDH assumption does not hold in the whole group $(\mathbf{Z}/p\mathbf{Z})^*$. As a result, in order to extend the message space to $(\mathbf{Z}/p\mathbf{Z})^*$, we need to encode elements of $(\mathbf{Z}/p\mathbf{Z})^*$ as quadratic residues. The situation is quite similar to the Elgamal over $(\mathbf{Z}/n\mathbf{Z})^*$ of [10], but actually simpler to handle since we work modulo a prime $p$ and not modulo an RSA integer $n$ (in particular, we can publicly compute square roots or distinguish quadratic and non quadratic residues and we do not have to hide the factorization of $n$).

Since $p = 2q + 1$, we have $p \equiv 3 \pmod 4$, and $-1$ is not a quadratic residue modulo $p$. Let $m \in (\mathbf{Z}/p\mathbf{Z})^*$, let us denote by $(m/p)$ the Legendre symbol of $m$ modulo $p$. Then $(m/p) \times m$ is a quadratic residue mod $p$. Let $L$ be the group morphism from $((\mathbf{Z}/p\mathbf{Z})^*, \times)$ to $(\mathbf{Z}/2\mathbf{Z}, +)$ that maps $m$ to 0 (resp. to 1) if $m$ is a quadratic residue (resp. is a non quadratic residue). The map

$$\begin{array}{ccccc} ((\mathbf{Z}/p\mathbf{Z})^*, \times) & \longrightarrow & (S_p, \times) & \times & (\mathbf{Z}/2\mathbf{Z}, +) \\ m & \longmapsto & \big((m/p) \times m & , & L(m)\big) \end{array}$$

is a group isomorphism. As a consequence we can encode elements of $(\mathbf{Z}/p\mathbf{Z})^*$ as a square plus one bit. The square can be encrypted with the traditional Elgamal encryption, and the bit $L(m)$ has to be encrypted separately. In order to have a multiplicatively homomorphic encryption, $L(m)$ has to be encrypted with a scheme that is homomorphic for the addition in $\mathbf{Z}/2\mathbf{Z}$. We choose Goldwasser-Micali encryption [23] for that. The drawback is that we need an additional assumption, namely the Quadratic Residuosity assumption (QR) for the security of our protocol. To avoid that, an idea could have been to encrypt $L(m)$ as an integer in the exponent with another Elgamal scheme or with the additive scheme of the previous Subsection. However, after computing the product of $\ell$ messages $m_1, \ldots, m_\ell$ over encrypted data, the decryption would give more information than the Legendre symbol of the product of the $m_i$'s, namely $\sum_{i=1}^{n} L(m_i)$ in the integers, instead of modulo 2. Moreover, this extra information has to be taken into account to devise a zero-knowledge 2-party decryption. As this information can not be simulated, this gives a complex 2-party protocol, perhaps by using an extra homomorphic encryption scheme like in [10]. Note that a solution consisting in randomizing $L(m)$ by adding a (small) even integer, with a Gaussian Distribution, for instance, still leaks the number $\ell$ of multiplications

that have been made. As a result, it seems to be an interesting open problem to devise an encryption scheme that allows homomorphic addition in $\mathbf{Z}/2\mathbf{Z}$, or that simulates it without leaks, without relying on a factorization-based assumption (in [10], the same problem was handled more smoothly thanks to the fact that the authors worked with a composite modulus).

*Description of the scheme.* Let $\lambda$ be a security parameter. Let $\mathsf{GM.Gen}$ be a parameter generator for the Goldwasser-Micali encryption scheme. It takes as input $1^\lambda$ and outputs $(N, p', q')$ such that $p', q' \equiv 3 \pmod 4$ are primes and $N = p'q'$ is such that factoring $N$ takes $2^\lambda$ time. We use the threshold variant of Goldwasser-Micali described in [26] to define a suitable 2-party decryption.

We also define $\mathsf{Eg}^*.\mathsf{Gen}$ a parameter generator for Elgamal. It takes as input $1^\lambda$ and outputs $(p, q, g)$ such that $q$ is a prime, $p = 2q + 1$ is a prime such that computing discrete logarithms in $(\mathbf{Z}/p\mathbf{Z})^*$ takes $2^\lambda$ times, and $g$ a generator of $S_p$, *i.e.*, and element of $(\mathbf{Z}/p\mathbf{Z})^*$ of order $q$. We depict in Fig. 10, the adaptation of Elgamal over the whole multiplicative group $(\mathbf{Z}/p\mathbf{Z})^*$, denoted $\mathsf{Eg}^*$.

$\underline{\mathsf{Eg}^*.\mathsf{Setup}(1^\lambda)}$

1. Set $(p, q, g) \leftarrow \mathsf{Eg}^*.\mathsf{Gen}(1^\lambda)$
2. Return $\mathsf{params} := (p, q, g)$

$\underline{\mathsf{Eg}^*.\mathsf{KeyGen}}$

1. Set $(N, p', q') \leftarrow \mathsf{GM.Gen}(1^\lambda)$
2. Add $N$ to $\mathsf{params}$
3. Pick $x \xleftarrow{\$} \{0, \ldots, q-1\}$
4. Set $h \leftarrow g^x$
5. Set $pk \leftarrow h$
6. Set $sk \leftarrow (x, p', q')$
7. Return $(pk, sk)$

$\underline{\mathsf{Eg}^*.\mathsf{Decrypt}(sk, (c_1, c_2, c_3))}$

1. Set $M \leftarrow c_2/c_1^x \pmod p$
2. Set $L \leftarrow c_3^{(N-p'-q'+1)/4} \pmod N$
3. If $L = 1$ return $M$ else return $-M$

$\underline{\mathsf{Eg}^*.\mathsf{Encrypt}(pk, m)}$

1. Pick $r \xleftarrow{\$} \{1, \ldots, q-1\}$
2. Pick $r' \xleftarrow{\$} \{1, \ldots, N-1\}$
3. Set $c_1 \leftarrow g^r \pmod p$
4. Set $c_2 \leftarrow (m/p)mh^r \pmod p$
5. Set $c_3 \leftarrow (-1)^{L(m)}r'^2 \pmod N$
6. Return $(c_1, c_2, c_3)$

$\underline{\mathsf{Eg}^*.\mathsf{Hom}_\times(pk, (c_1, c_2, c_3), (c'_1, c'_2, c'_3))}$

1. Pick $r \xleftarrow{\$} \{0, \ldots, q-1\}$
2. Pick $r' \xleftarrow{\$} \{1, \ldots, N-1\}$
3. Return $(c_1c'_1g^r, c_2c'_2h^r, c_3c'_3r'^2)$

$\underline{\mathsf{Eg}^*.\mathsf{ScalMul}(pk, (c_1, c_2, c_3), \alpha)}$

1. Pick $r \xleftarrow{\$} \{0, \ldots, q-1\}$
2. Pick $r' \xleftarrow{\$} \{1, \ldots, N-1\}$
3. Set $c'_1 \leftarrow c_1g^r \pmod p$
4. Set $c'_2 \leftarrow (\alpha/p)\alpha c_2h^r \pmod p$
5. Set $c'_3 \leftarrow (-1)^{L(\alpha)}c_3r'^2 \pmod N$
6. Return $(c'_1, c'_2, c'_3)$

**Fig. 10.** Elgamal over $(\mathbf{Z}/p\mathbf{Z})^*$: $\mathsf{Eg}^*$

The following theorem is a consequence of the previous discussion and the properties of the Goldwasser-Micali variant. Note that modulo $N$, $c_3^{(N-p'-q'+1)/4}$ equals 1 if $c_3$ is a quadratic residue, and $-1$ if $c_3$ has Jacobi symbol 1 and is not a quadratic residue.

**Theorem 8.** *The* $\mathsf{Eg}^*$ *scheme of Fig. 10 is multiplicatively homomorphic over* $(\mathbf{Z}/p\mathbf{Z})^*$*, and it is IND-CPA under the* DDH *assumption in the subgroup of quadratic residues of* $(\mathbf{Z}/p\mathbf{Z})^*$ *and the* QR *assumption in* $(\mathbf{Z}/N\mathbf{Z})^\times$*.*

**One round 2-party decryption for $\mathsf{Eg}^*$** We describe in Fig. 11 a one round 2-party decryption for $\mathsf{Eg}^*$. This protocol is adapted from the threshold variant of the original Elgamal scheme and the basic threshold Goldwasser-Micali of [26, Subsection 3.1].

This simple protocol gives a huge performance improvement compared to the Elgamal over $(\mathbf{Z}/n\mathbf{Z})^*$ of [10]: in that work, after the exponentiations, a CRT reconstruction is needed to recover $m$, and a quantity that leads to the factorization of $n$ must be shared. To make this 2-party reconstruction zero-knowledge, the authors use an additional additively homomorphic encryption, and have to do the reconstruction over encrypted data. As a result, the protocol is very complex (implicitly described in [10, Fig. 2]) with 5 rounds instead of 1.

$\mathsf{Eg}^*.\mathsf{Share}(sk, pk)$

1. Parse $(x, p', q') = sk$
2. Pick $x_A \xleftarrow{\$} \{0, \ldots, q-1\}$ and set $x_B \equiv x - x_A \pmod q$
3. Pick $p_A, p_B, q_A, q_B \xleftarrow{\$} \{0, \ldots, N\}$ such that $p_A \equiv p_B \equiv q_A \equiv q_B \equiv 0 \pmod 4$
4. Set $p_0 = p' - p_A - p_B$ and $q_0 = q' - q_A - q_B$
5. Set $pk \leftarrow (pk, N_0 = (N - p_0 - q_0)/4)$
6. Return $(pk, sk_A = (x_A, x'_A), sk_B = (x_B, x'_B))$

$\mathsf{Eg}^*.\mathsf{2Dec}(pk, C = (c_1, c_2, c_3); sk_A = (x_A, p_A, q_A); sk_B = (x_B, p_B, q_B))$

| Input : $pk, sk_A, C$ | | Input : $pk, sk_B$ |
|---|---|---|
| $c_{1,A} \leftarrow c_1^{x_A} \pmod p$ | | |
| $c_{3,A} \leftarrow c_3^{(-p_A-q_A)/4} \pmod N$ | | |
| | $\xrightarrow{C, c_{1,A}, c_{3,A}}$ | Set $M \leftarrow c_2/(c_1^{x_B} c_{1,A}) \pmod p$ |
| | | Set $L \leftarrow c_{3,A} c_3^{(-p_B-q_B)/4} C_3^{N_0} \pmod N$ |
| | | If $L = 1$ return $M$ else return $-M$ |

**Fig. 11.** 2-party Decryption for $\mathsf{Eg}^*$

**Theorem 9.** *The 2-party Decryption for* $\mathsf{Eg}^*$ *described in Fig. 11 is correct and zero-knowledge.*

*Proof.* The proof is similar to the proof of Theorem 7. For the Elgamal part of the protocol, secret key shares are simply taken uniformly at random in $\{0, \ldots, q-1\}$, and the value sent by Alice is computed as $c_{1,A} = c_2/(Mc_1^{x_B})$, where $M = (m/p)m$. The Goldwasser-Micali part of the protocol is also simulated in a similar fashion from $c_3$ and $L(m)$ and the key share from a fake factorization of $N$ just as in [26, Subsection 3.1] □

**Extension of the message space from $(\mathbf{Z}/p\mathbf{Z})^*$ to $\mathbf{Z}/p\mathbf{Z}$** We use the generic construction depicted in Fig. 4 with the additively homomorphic scheme described in the previous subsection. We denote by $\mathsf{Eg}_p.\mathsf{Gen}$, a group generator which combines the generators for $\mathsf{Eg}^*$ and $\mathsf{CL}$ : on input $1^\lambda$, it first runs $\mathsf{Eg}^*.\mathsf{Gen}$, which outputs $(p, q, g)$. The prime $p$ equals 3 mod 4 and is such that computing discrete logarithms in $(\mathbf{Z}/p\mathbf{Z})^*$ takes time $2^\lambda$. As the best algorithms for computing such discrete logarithms are faster than the algorithms for computing discrete logarithms in the class group $C(-p)$ (the sub-exponential complexity is respectively $L_p[1/3, (64/9)^{1/3} + o(1)]$ and $L_p[1/2, 1 + o(1)]$, see [43,24]), this prime $p$ is compatible with the prime generated by $\mathsf{CL}.\mathsf{Gen}$. As a result $\mathsf{Eg}_p.\mathsf{Gen}$ executes $\mathsf{CL}.\mathsf{Gen}$ by setting this prime $p$ and adapts the others quantities accordingly. The resulting scheme is described in Fig. 12 for completeness.

### 5.3 ESP over $\mathbf{Z}/p\mathbf{Z}$: Efficiency and Comparisons

In Table 1 we give the round complexity ($rc$) and bit complexity ($bc$) of our algorithms and we compare our full ESP protocols with that of Couteau et al. [10]. For sake of clarity, and because it is identical to that of [10], we omit the complexities resultant from the garbled circuit-based $\mathsf{EZT}$ protocols. Our 2-party decryption algorithms (both for $\mathsf{CL}$ and $\mathsf{Eg}^*$) only require 1 round. Note that we carefully analyzed the interactive algorithms so as to gather consecutive flows when possible within a single round. For example our $\mathsf{2Mul}_+$ and $\mathsf{ReEnc}_+$ protocols (see Fig. 1 and 2) require only 2 rounds since Alice can send $C^+_{rX}$ (resp. $C'_{-r}$) and her $\mathsf{2Dec}$ data simultaneously. For the same reason, our generic switches in $\mathcal{R}^*$ also require 2 rounds. Therefore, our $(\Pi_+ \rightleftharpoons \Pi^0_\times).\mathsf{Switch}_{+\to\times}$ needs 7 rounds: 2 for the initial $\mathsf{EZT}$, 2 for $\mathsf{ReEnc}_+$, 2 for sending $C^+_{m+b}$ and the $\mathsf{Switch}_{+\to\times}$, and 1 for sending the final result to Bob. In the other direction, the initial $\mathsf{Switch}_{\times\to+}$ and the $\mathsf{EZT}$ are independent and can thus be processed simultaneously in 2 rounds. Adding 2 rounds for the $\mathsf{2Mul}_+$, the round complexity for $(\Pi_+ \rightleftharpoons \Pi^0_\times).\mathsf{Switch}_{\times\to+}$ adds up to 4 rounds only. In comparison, and using the same optimizations, the ESP switches from Couteau et al. requires 7 and 11 rounds respectively.

We express the communication cost in terms of the number of bits exchanged between the parties. The bit complexity ($bc$) is given as a function of the ring/field size. Observe that although, the best (conjectured) asymptotic complexity to compute a discrete logarithm in the ideal class group used

$\mathsf{Eg}_p.\mathsf{Setup}(1^\lambda)$

1. Set $(p, q, g, \mathfrak{g}, \mathfrak{f}, \sigma) \leftarrow \mathsf{Eg}^*.\mathsf{Gen}(1^\lambda)$
2. Return $\mathsf{params} := (p, q, g, \mathfrak{g}, \mathfrak{f}, \sigma)$

$\mathsf{Eg}_p.\mathsf{KeyGen}$

1. Set $(N, p', q') \leftarrow \mathsf{GM.Gen}(1^\lambda)$
2. Add $N$ to $\mathsf{params}$
3. Pick $x^\times \xleftarrow{\$} \{0, \ldots, q-1\}$
4. Set $h^\times \leftarrow g^{x^\times}$
5. Pick $x^+ \xleftarrow{\$} \mathcal{D}_\sigma$ and set $\mathfrak{h}^+ \leftarrow \mathfrak{g}^{x^+}$
6. Pick $x' \xleftarrow{\$} \mathcal{D}_\sigma$ and set $\mathfrak{h}' \leftarrow \mathfrak{g}^{x'}$
7. Set $pk \leftarrow (h^\times, \mathfrak{h}^+, \mathfrak{h}')$
8. Set $sk \leftarrow (x^\times, p', q', x^+, x')$
9. Return $(pk, sk)$

$\mathsf{Eg}_p.\mathsf{Decrypt}(sk, (c_1, c_2, c_3))$

1. Set $\mathfrak{M} \leftarrow \mathfrak{c}_2/\mathfrak{c}_1^{x^+}$
2. Set $B \leftarrow \mathsf{CL.Solve}(\mathfrak{M})$
3. If $B \neq 0$ return 0
4. Set $M \leftarrow c_2/c_1^{x^\times} \pmod p$
5. Set $L \leftarrow c_3^{(N-p'-q'+1)/4} \pmod N$
6. If $L = 1$ return $M$ else return $-M$

$\mathsf{Eg}_p.\mathsf{Encrypt}(pk, m)$

1. If $m = 0$ set $b \leftarrow 1$ and $r \xleftarrow{\$} (\mathbf{Z}/p\mathbf{Z})^*$ otherwise set $b \leftarrow 0$ and $r \leftarrow 0$
2. Pick $r^\times \xleftarrow{\$} \{1, \ldots, q-1\}$
3. Pick $r^{\times\prime} \xleftarrow{\$} \{1, \ldots, N-1\}$
4. Set $c_1 \leftarrow g^{r^\times} \pmod p$
5. Set $c_2 \leftarrow ((m+b)/p)(m+b)h^{\times r^\times} \pmod p$
6. Set $c_3 \leftarrow (-1)^{L(m+b)} r^{\times\prime 2} \pmod N$
7. Pick $r^+ \xleftarrow{\$} \mathcal{D}_\sigma$
8. Compute $\mathfrak{c}_1 \leftarrow \mathfrak{g}^{r^+}, \mathfrak{c}_2 \leftarrow \mathfrak{f}^r \mathfrak{h}^{+r^+}$
9. Pick $r' \xleftarrow{\$} \mathcal{D}_\sigma$
10. Compute $\mathfrak{c}_1' \leftarrow \mathfrak{g}^{r'}, \mathfrak{c}_2' \leftarrow \mathfrak{f}^r \mathfrak{h}'^{r'}$
11. Return $(c_1, c_2, c_3, \mathfrak{c}_1, \mathfrak{c}_2, \mathfrak{c}_1', \mathfrak{c}_2')$

**Fig. 12.** Elgamal over $\mathbf{Z}/p\mathbf{Z}$ : $\mathsf{Eg}_p$

in $\mathsf{CL}$ is in $L_p[1/2, 1 + o(1)]$ (see. [24]), one must consider a prime $p$ that is large enough to guarantee that the DLP over $(\mathbf{Z}/p\mathbf{Z})^*$ is hard, i.e such that $L_p[1/3, (64/9)^{1/3} + o(1)] > 2^\lambda$ (see e.g. [43]). In Table 1, $\ell$, represents the bit length of $p$ for our protocol over $\mathbf{Z}/p\mathbf{Z}$ and of $n$ for Couteau et al.'s protocol over $\mathbf{Z}/n\mathbf{Z}$.

For our protocols, we give the bit complexities for two variants: for the version of $\mathsf{CL}$ used in this paper $bc$ is the cost deduced from Fig. 8. The drawback of this scheme is that ciphertexts are represented with 2 elements of $C(-p^3)$ which gives $2 \times 2 \times \frac{3}{2} \times \ell = 6\ell$ against $2\ell$ for Paillier. Therefore, we include a column with the cost $bc$' that correspond to the so-called "faster variant" of $\mathsf{CL}$ from [9, Section 4]. This variant defines ciphertexts in $C(-p) \times C(-p^3)$, represented with $\ell + 3\ell = 4\ell$ elements. Moreover, for 2-party decryption we only have to share an exponentiation in $C(-p)$ instead of $C(-p^3)$ so the cost drops from $6\ell + 3\ell = 9\ell$ to $4\ell + \ell = 5\ell$.

For the former variant, the security depends upon DDH in $C(-p^3)$ whereas for the faster variant it is based upon the following indistinguishability argument: Let $\mathfrak{g}$ be a generator of a subgroup of $C(-p)$. After having chosen $m$, the adversary is asked to distinguished the following distributions : $\{(\mathfrak{g}^x, \mathfrak{g}^y, \psi(\mathfrak{g}^{xy})), x, y \leftarrow \mathcal{D}_{\sigma/p}\}$ and $\{(\mathfrak{g}^x, \mathfrak{g}^y, \psi(\mathfrak{g}^{xy})\mathfrak{f}^m), x, y \leftarrow \mathcal{D}_{\sigma/p}\}$, where $\mathcal{D}_\sigma$ is the Gaussian Discrete distribution defined in Subsection 5.1 and $\psi$ is a lifting map from $C(-p)$ to $C(-p^3)$, defined in [9, Lemma 3]. We denote LDDH by the corresponding assumption. The algorithmic assumptions required for each protocol are presented in Table 2.

**Table 1.** Comparisons of the round complexities and bit complexities of our protocols (v1 and v2) with that of Couteau et al. [10]. $(*)$ For the $\mathsf{EZT}$ protocol the communication cost for the garbled circuit is omitted as it is the same for v1, v2 and [10] (cf. Subsection 4.3 for the cost).

| | round complexity | | bit complexity | | |
|---|---|---|---|---|---|
| Algorithms | this work | [10] | v1 | v2 | [10] |
| $\mathsf{Eg}^*.\mathsf{2Dec}$ | 1 | n/a | $5\ell$ | $5\ell$ | n/a |
| $\mathsf{CL.2Dec}$ | 1 | n/a | $9\ell$ | $5\ell$ | n/a |
| $\mathsf{CL.EZT}(*)$ | 2 | n/a | $15\ell$ | $9\ell$ | n/a |
| $\mathsf{CL.2Mul}_+$ | 2 | n/a | $21\ell$ | $13\ell$ | n/a |
| $\mathsf{CL.ReEnc}_+$ | 2 | n/a | $21\ell$ | $13\ell$ | n/a |
| $(\Pi_+ \rightleftharpoons \Pi_\times).\mathsf{Switch}_{+\to\times}$ | 2 | 2 | $15\ell$ | $11\ell$ | $10\ell$ |
| $(\Pi_+ \rightleftharpoons \Pi_\times).\mathsf{Switch}_{\times\to+}$ | 2 | 6 | $17\ell$ | $13\ell$ | $36\ell$ |

| | round complexity | | bit complexity | | |
|---|---|---|---|---|---|
| ESP protocols | this work | [10] | v1 | v2 | [10] |
| $(\Pi_+ \rightleftharpoons \Pi^0_\times).\mathsf{Switch}_{+\to\times}$ | 7 | 7 | $69\ell$ | $45\ell$ | $37\ell$ |
| $(\Pi_+ \rightleftharpoons \Pi^0_\times).\mathsf{Switch}_{\times\to+}$ | 4 | 11 | $53\ell$ | $35\ell$ | $61\ell$ |

**Table 2.** Algorithmic assumptions

| This work (v1) | This work (v2) | [10] |
|---|---|---|
| DDH in $C(-p^3)$ | LDDH in $C(-p^3)$ | DCR |
| DDH in $S_p$ | DDH in $S_p$ | DDH in $S_n$ |
| QR | QR | QR |

## 6 ESP secure against malicious adversaries

To reach the security against malicious adversaries, it is necessary to add zero-knowledge proofs by all parties that every computation is done correctly with the knowledge of every plaintext. In [10], the zero-knowledge proofs are classical Schnorr-like proofs and range proofs, but they need also to design a new strong primitive called *twin ciphertext proof* (TCP) to prove that a pair of ciphertexts from two different encryption schemes is actually a pair of twin ciphertexts. This allows to avoid generic circuit-based zero-knowledge proofs, but still requires a costly cut-and-choose technique (which can be amortized). This proof consists first in gathering a large pool of random genuine twin ciphertexts (proved thanks to the knowledge of the plaintext and the randomness, and of the homomorphic property of the encryption schemes). This part is done once for all. During an ESP, each time a twin ciphertext proof is needed, a fresh twin ciphertext pair is taken from the pool to perform a simple co-linearity proof.

To enhance our generic construction against malicious adversaries, we use the same method. In fact, the additional properties needed for the homomorphic encryption schemes are the same as in [11]: the $\Pi_+$ and the $\Pi_\times$ encryption schemes must support zero-knowledge proof of plaintext knowledge, proof that the ScalMul operation has been performed correctly and also support a 2-party decryption in the malicious setting. Then we use the TCP technique as in [10] for twin ciphertext proofs.

As a result, we modify our generic construction by adding such proofs in each step of the switching protocols. This ensures honest behavior and thus make the ESP secure in the malicious settings. In particular this brings soundness in the sense of [10]: no malicious player can force the output of an ESP not to be a twin ciphertext.

The protocols $\Pi_+$ and $\Pi_\times$ described in the instantiation from the previous section support the required features. For the $\Pi_\times$ encryption scheme, we need zero knowledge proofs and 2-party decryption secure against malicious adversary for the classical Elgamal and for the Goldwasser-Micali encryption scheme. This can be done with classical methods: zero-knowledge proof *à la* Schnorr, adding verification keys to the public keys for 2-party decryption and proof of exponentiations to the same power. Note that for Goldwasser-Micali, we need to modify key generation to use strong primes $p'$ and $q'$ as in [26].

For the $\Pi_+$ encryption scheme which is based on the Castagnos-Laguillaumie encryption scheme, we need proofs for an Elgamal variant in a group of unknown

order, namely a class group of a quadratic order. Then 2-party decryption secure against malicious adversary is obtained as for the $\Pi_\times$ scheme.

Generalizations of Schnorr proofs in group of unknown orders have been addressed extensively in [7]. In this framework, a generalized Schnorr proof can be used if the cyclic group considered is what is called a *safeguard group*, which is roughly a group whose set of small orders elements is small and known, and for which it is hard to find roots of elements. The case of class groups has been explicitly taken in account for example in [13,12], where it is argue in particular that class groups of discriminant $-p$, $C(-p)$, can be considered to have the properties of safeguard groups. As a result, we can apply directly the framework of [7] for the faster variant of $\mathsf{CL}$ mentioned in Subsection 5.3 as exponentiations are defined in $C(-p)$ for this variant.

## 7 Conclusion

The encryption switching protocol is a promising cryptographic primitive formalized by Couteau et al. in [10]. We propose in this article a generic framework to build such an ESP. Our approach makes the design of an ESP simple and efficient. In particular, we propose an instantiation whose round complexity is dramatically improved compared to Couteau et al., since we reduce by a factor 3 the number of round in the multiplicative to additive direction (while we have the same number of rounds in the other way). Again, in terms of bit complexity, our switching protocol in the multiplicative to additive direction gains a factor almost 1.7, while in the other direction Couteau et al.'s switch is smaller by a factor 1.2. This is essentially because in our case, the additively homomorphic encryption has large ciphertexts. In particular, any additively homomorphic encryption satisfying the conditions of our construction with smaller elements will allow to gain in terms of bit complexity. Our instantiation, which is secure in the semi-honest model under classical assumptions can be extended to the malicious case. We believe that it is possible to improve our instantiation by deviating a bit more from the generic construction. Moreover, an interesting open problem is to design an encryption scheme which is homomorphic for the $+$ in $\mathbf{F}_2$ without the factorization assumption. A consequence could be to have an ESP whose security relies only on a discrete logarithm related assumption. Designing a more efficient encrypted zero-test is also a direction which will allow a significant improvement in the protocol.

**Acknowledgments:** The authors would like to thank Geoffroy Couteau for fruitful discussions, careful reading and constructive comments on the preliminary version of this work. We also express our thanks to Bruno Grenet, Romain Lebreton, Benoît Libert and Damien Stehlé for their feedbacks.

The authors are supported in part by the French ANR ALAMBIC project (ANR-16-CE39-0006), by the "Investments for the future" Programme IdEx Bordeaux - CPU (ANR-10-IDEX-03-02), and by ERC Starting Grant ERC-2013-StG-335086-LATTAC.

## References


1. Allender, E., Jiao, J., Mahajan, M., Vinay, V.: Non-commutative arithmetic circuits: depth reduction and size lower bounds. Theoretical Computer Science 209(1), 47 – 86 (1998)
2. Asharov, G., Jain, A., López-Alt, A., Tromer, E., Vaikuntanathan, V., Wichs, D.: Multiparty computation with low communication, computation and interaction via threshold FHE. In: EUROCRYPT 2012. LNCS, vol. 7237, pp. 483–501. Springer (2012)
3. Asharov, G., Lindell, Y., Schneider, T., Zohner, M.: More efficient oblivious transfer extensions with security for malicious adversaries. In: EUROCRYPT 2015, Part I. LNCS, vol. 9056, pp. 673–701. Springer (2015)
4. Beaver, D.: Foundations of secure interactive computing. In: CRYPTO'91. LNCS, vol. 576, pp. 377–391. Springer (1992)
5. Bendlin, R., Damgård, I., Orlandi, C., Zakarias, S.: Semi-homomorphic encryption and multiparty computation. In: EUROCRYPT 2011. LNCS, vol. 6632, pp. 169–188. Springer (2011)
6. Bogetoft, P., Damgård, I., Jakobsen, T., Nielsen, K., Pagter, J., Toft, T.: A practical implementation of secure auctions based on multiparty integer computation. In: FC 2006. LNCS, vol. 4107, pp. 142–147. Springer (2006)
7. Camenisch, J., Kiayias, A., Yung, M.: On the portability of generalized Schnorr proofs. In: EUROCRYPT 2009. LNCS, vol. 5479, pp. 425–442. Springer (2009)
8. Canetti, R.: Security and composition of multiparty cryptographic protocols. Journal of Cryptology 13(1), 143–202 (2000)
9. Castagnos, G., Laguillaumie, F.: Linearly homomorphic encryption from DDH. In: CT-RSA 2015. LNCS, vol. 9048, pp. 487–505. Springer (2015)
10. Couteau, G., Peters, T., Pointcheval, D.: Encryption switching protocols. In: CRYPTO 2016, Part I. LNCS, vol. 9814, pp. 308–338. Springer (2016)
11. Cramer, R., Damgård, I., Nielsen, J.B.: Multiparty computation from threshold homomorphic encryption. In: EUROCRYPT 2001. LNCS, vol. 2045, pp. 280–299. Springer (2001)
12. Damgård, I., Fujisaki, E.: A statistically-hiding integer commitment scheme based on groups with hidden order. In: ASIACRYPT 2002. LNCS, vol. 2501, pp. 125–142. Springer (2002)
13. Damgård, I., Koprowski, M.: Generic lower bounds for root extraction and signature schemes in general groups. In: EUROCRYPT 2002. LNCS, vol. 2332, pp. 256–271. Springer (2002)
14. Damgård, I., Mikkelsen, G.L.: Efficient, robust and constant-round distributed RSA key generation. In: TCC 2010. LNCS, vol. 5978, pp. 183–200. Springer (2010)
15. Damgård, I., Nielsen, J.B.: Universally composable efficient multiparty computation from threshold homomorphic encryption. In: CRYPTO 2003. LNCS, vol. 2729, pp. 247–264. Springer (2003)
16. Damgård, I., Zakarias, S.: Constant-overhead secure computation of Boolean circuits using preprocessing. In: TCC 2013. LNCS, vol. 7785, pp. 621–641. Springer (2013)
17. Desmedt, Y., Frankel, Y.: Threshold cryptosystems. In: CRYPTO'89. LNCS, vol. 435, pp. 307–315. Springer (1990)
18. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Transactions on Information Theory 31, 469–472 (1985)

19. Fouque, P.A., Poupard, G., Stern, J.: Sharing decryption in the context of voting or lotteries. In: FC 2000. LNCS, vol. 1962, pp. 90–104. Springer (2001)
20. Gavin, G., Minier, M.: Oblivious multi-variate polynomial evaluation. In: INDOCRYPT 2009. LNCS, vol. 5922, pp. 430–442. Springer (2009)
21. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: 40th ACM STOC. pp. 197–206. ACM Press (2008)
22. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or A completeness theorem for protocols with honest majority. In: 19th ACM STOC. pp. 218–229. ACM Press (1987)
23. Goldwasser, S., Micali, S.: Probabilistic encryption. Journal of Computer and System Sciences 28(2), 270–299 (1984)
24. Jacobson Jr., M.J.: Computing discrete logarithms in quadratic orders. Journal of Cryptology 13(4), 473–492 (2000)
25. Katz, J., Lindell, Y.: Introduction to Modern Cryptography, Second Edition. Chapman & Hall/CRC, 2nd edn. (2014)
26. Katz, J., Yung, M.: Threshold cryptosystems based on factoring. In: ASIACRYPT 2002. LNCS, vol. 2501, pp. 192–205. Springer (2002)
27. Kiayias, A., Yung, M.: Secure games with polynomial expressions. In: Automata, Languages and Programming, 28th International Colloquium, ICALP 2001, Proceedings. LNCS, vol. 2076, pp. 939–950. Springer (2001)
28. Kolesnikov, V., Schneider, T.: Improved garbled circuit: Free XOR gates and applications. In: ICALP 2008, Part II. LNCS, vol. 5126, pp. 486–498. Springer (2008)
29. Lim, H.W., Tople, S., Saxena, P., Chang, E.C.: Faster secure arithmetic computation using switchable homomorphic encryption. Cryptology ePrint Archive, Report 2014/539 (2014)
30. Lindell, Y., Pinkas, B.: Secure two-party computation via cut-and-choose oblivious transfer. In: TCC 2011. LNCS, vol. 6597, pp. 329–346. Springer (2011)
31. Lindell, Y., Pinkas, B., Smart, N.P.: Implementing two-party computation efficiently with security against malicious adversaries. In: SCN 08. LNCS, vol. 5229, pp. 2–20. Springer (2008)
32. Lipmaa, H., Toft, T.: Secure equality and greater-than tests with sublinear online complexity. In: ICALP 2013, Part II. LNCS, vol. 7966, pp. 645–656. Springer (2013)
33. Malkhi, D., Nisan, N., Pinkas, B., Sella, Y.: Fairplay - secure two-party computation system. In: Proceedings of the 13th USENIX Security Symposium, August 9-13, 2004, San Diego, CA, USA. pp. 287–302. USENIX (2004)
34. Micali, S., Rogaway, P.: Secure computation (abstract). In: CRYPTO'91. LNCS, vol. 576, pp. 392–404. Springer (1992)
35. Micciancio, D., Regev, O.: Worst-case to average-case reductions based on gaussian measures. SIAM J. Comput. 37(1), 267–302 (2007)
36. Naor, M., Pinkas, B.: Oblivious polynomial evaluation. SIAM J. Comput. 35(5), 1254–1281 (2006)
37. Nielsen, J.B., Nordholt, P.S., Orlandi, C., Burra, S.S.: A new approach to practical active-secure two-party computation. In: CRYPTO 2012. LNCS, vol. 7417, pp. 681–700. Springer (2012)
38. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: EUROCRYPT'99. LNCS, vol. 1592, pp. 223–238. Springer (1999)
39. Pedersen, T.P.: A threshold cryptosystem without a trusted party (extended abstract) (rump session). In: EUROCRYPT'91. LNCS, vol. 547, pp. 522–526. Springer (1991)

40. Pinkas, B., Schneider, T., Smart, N.P., Williams, S.C.: Secure two-party computation is practical. In: ASIACRYPT 2009. LNCS, vol. 5912, pp. 250–267. Springer (2009)
41. Shoup, V., Gennaro, R.: Securing threshold cryptosystems against chosen ciphertext attack. Journal of Cryptology 15(2), 75–96 (2002)
42. Tassa, T., Jarrous, A., Ben-Ya'akov, Y.: Oblivious evaluation of multivariate polynomials. J. Mathematical Cryptology 7(1), 1–29 (2013)
43. Thomé, E.: Algorithmic Number Theory and Applications to the Cryptanalysis of Cryptographical Primitives. Habilitation à diriger des recherches, Université de Lorraine (2012)
44. Tople, S., Shinde, S., Chen, Z., Saxena, P.: AUTOCRYPT: enabling homomorphic computation on servers to protect sensitive web content. In: ACM CCS 13. pp. 1297–1310. ACM Press (2013)
45. Valiant, L.G., Skyum, S., Berkowitz, S., Rackoff, C.: Fast parallel computation of polynomials using few processors. SIAM Journal on Computing 12(4), 641–644 (1983)
46. Yao, A.C.C.: Protocols for secure computations (extended abstract). In: 23rd FOCS. pp. 160–164. IEEE Computer Society Press (Nov 1982)
47. Ye, Q., Wang, H., Pieprzyk, J., Zhang, X.M.: Efficient disjointness tests for private datasets. In: ACISP 08. LNCS, vol. 5107, pp. 155–169. Springer (2008)

## A 2-Party Decryption : zero-knowledge

**Definition 7.** *An encryption scheme $\Pi$ supporting 2-party decryption is* zero-knowledge for $A$ *if there exists an efficient simulator* $\mathsf{Sim}^{\mathsf{2d}} = (\mathsf{Sim}^{2d}_{\mathsf{Share}}, \mathsf{Sim}^{2d}_{A})$ *which simulates the sharing phase and the player $A$.*

*The subroutine* $\mathsf{Sim}^{2d}_{\mathsf{Share}}$ *takes as input a public key $pk$ and outputs $(pk', sk'_B)$ that simulates the public key obtained from the* Share *algorithm and Bob's share of the secret key.*

*The subroutine* $\mathsf{Sim}^{2d}_{A}$ *takes as input a public key $pk$ a ciphertext $c$, a plaintext $m$, possibly $sk_B$ and a flow* flow*. It emulates honest player $A$'s answer upon receiving the flow* flow *when running the protocol* $2\mathsf{Dec}(pk, c; sk_A; sk_B)$ *without $sk_A$, and forcing the output to be $m$.*

*Then, for all $\lambda \in \mathbf{N}$, for any* $(\mathsf{params} \leftarrow \mathsf{Setup}(1^\lambda)$*, for any pair of keys* $(pk, sk) \leftarrow \Pi.\mathsf{KeyGen}(1^\lambda, \mathsf{params})$*, for any shares* $(pk, sk_A, sk_B) \leftarrow \mathsf{Share}(pk, sk)$ *or for any* simulated *share* $(pk', sk'_B) \leftarrow \mathsf{Sim}^{2d}_{\mathsf{Share}}(pk)$*, and for any adversary $\mathcal{D}$ playing the role of $B$, the advantage*

$$\mathsf{Adv}^{zk}_{A,\Pi}(\mathcal{D}) = \left|\Pr[1 \leftarrow \mathcal{D}^{A}(pk, sk_B)] - \Pr[1 \leftarrow \mathcal{D}^{\mathsf{Sim}^{2d}_{A}()}(pk', sk'_B)]\right|$$

*is negligible.*

*We define similarly that $\Pi$ is* zero-knowledge for $B$*. It is* zero-knowledge *if it is zero-knowledge for $A$ and $B$.*

## B Proof of Theorem 5

**Theorem 5.** *The ESP between $\Pi_+$ and $\Pi^0_\times$ whose routines are described in Fig. 6 and 7 is zero-knowledge if $\Pi_+$ and $\Pi_\times$ are two compatible encryptions*

*that are IND-CPA and whose 2-party decryption are zero-knowledge, and* $\mathsf{EZT}$ *is zero-knowledge.*

*Proof.* Once again, the proof consists in proving that after a share of the secret keys, both switching procedures are zero-knowledge for Alice and Bob. As both switches consist in a sequence of protocols that have been independently proved secure, the main issue in the proof consists in showing that their sequential combination is still secure. The reduction will get a pair $(C, \bar{C})$ of input and output of the whole switches, and the main idea is to construct such intermediate pairs for each independent subroutines using random ciphertexts.

**ZK for Alice.** Let us start with the proof that the ESP is zero-knowledge for Alice. We describe a simulator $\mathsf{Sim}$ whose behavior is indistinguishable from Alice's behavior in front of an adversarial Bob.
$\mathsf{Sim}_{\mathsf{Share}}$: The simulator receives the public key $(pk_+, pk_\times)$ and sets $\mathsf{Sim}_{\mathsf{Share}}$ as follows: it calls out the $\mathsf{Sim}^{2d}_{\mathsf{Share}}$ procedures of the zero-knowledge property of Alice for 2-party decryption of respectively $\Pi_+$ and $\Pi_\times$ with $pk_+$ and $pk_\times$ as input. In particular it gets $sk'_B = (x^+_B, x^\times_B)$ it feeds the adversary with. When $\mathsf{Sim}$ is requested for a switch, it receives a pair of twin ciphertexts $(C, \bar{C})$.

**Game** $G_0$**.** This game is the real game. The simulator simulates all the secrets in an honest way and gives his share to Bob. It plays honestly any switching protocols on an input $(C, \bar{C})$ using Alice's secret key.

**Game** $G_1$**.** Each time $\mathsf{Sim}$ is requested for a switch ($\mathsf{Switch}_{\times\to+}$ or $\mathsf{Switch}_{+\to\times}$) it is given as input $(C, \bar{C})$ and one of the two is an encryption of $m$ under $\Pi^0_\times$, which contains an $\Pi_+$-encryption under $pk'$ of the bit $b$. The simulation uses its knowledge of the secret key $sk'$ to decrypt the bit $b$. This game is indistinguishable from the previous one.

**Game** $G_2$**.** A modification is done for the *additive to multiplicative* case. The setup and key generation are the same as in the previous game. When requested to participate to a $\mathsf{Switch}_{+\to\times}$, with $(C, \bar{C})$ as input, the simulator uses its knowledge of $b$ to query the EZT's simulator for Alice with $(C, \Pi_+.\mathsf{Encrypt}(pk^+, b))$ as input. By definition of the simulator for the EZT, this game is indistinguishable from the previous one.

**Game** $G_3$**.** After the simulation of the EZT procedure, Alice and Bob gets $C^+_b$. The simulation now uses the $\mathsf{ReEnc}_+$'s simulator for Alice with this $C^+_b$ and $\Pi_+.\mathsf{Encrypt}(pk', b)$ as input, once again thanks to the knowledge of $b$. Thanks to the zero-knowledge property of $\mathsf{ReEnc}_+$, this game is indistinguishable from the previous one.

**Game** $G_4$**.** Now the simulation uses the simulator for the $\mathsf{Switch}_{+\to\times}$. As the simulation knows $\bar{C}$, it can extract its first component which is a $\Pi_\times$-encryption of $m + b$. Therefore, it calls $\mathsf{Switch}_{+\to\times}$'s simulator for Alice with $C^+_{m+b}$ (obtained by genuinely computing the $\mathsf{Hom}_+$ after the re-encryption) and $\bar{C}$'s first component. Because we proved that the $\mathsf{Switch}_{+\to\times}$ procedure is zero-knowledge in Theorem 3, this game is indistinguishable from the previous one.

**Game** $G_5$**.** The final flow from the switching protocols is simply the forward of $\bar{C}$ since it is a twin ciphertext of $C$: this game is indistinguishable from the previous one.

**Game** $G_6$**.** The modification now concerns the *multiplicative to additive* case. The simulation has as input $(C, \bar{C})$ where $C$ is an encryption of a message $m$ under $\Pi^0_\times$ and $\bar{C}$ is a twin ciphertext. $\mathsf{Sim}$ still knows the bit $b$. To simulate the switch, it uses the corresponding simulator for Alice with, as input the first component of $C$ which is an encryption using $\Pi_\times$ of $m + b$ and $\Pi_+.\mathsf{Hom}_+(pk^+, \bar{C}, \Pi_+.\mathsf{Encrypt}(pk^+, b))$ which is an encryption $m + b$ under $\Pi_+$. Because of the zero-knowledge property of this switch proved in Theorem 3, this game is indistinguishable from the previous one.

**Game** $G_7$**.** The simulation now simulates the EZT procedure: it feeds the corresponding simulator with the second component of $C$ (which is an encryption of a random element under $\Pi_+$) and $\Pi_+.\mathsf{Encrypt}(pk^+, \bar{b})$, which is a valid input. The EZT being zero-knowledge, this game is indistinguishable from the previous.

**Game** $G_8$**.** The last step of the switch in the multiplicative to additive direction is the computation of the $\Pi_+$ encryption of a product. The simulation makes a call to the simulator of the $\mathsf{2Mul}_+$ protocol with as input: the output of the first switch, the output of the EZT and $\bar{C}$. As this is a genuine input, this game is indistinguishable from the previous.

**Game** $G_9$**.** From now on, the simulation will not use its knowledge of $b$ and of the secret key $sk'$. To do so, in the additive to multiplicative direction, the simulation will feed the EZT simulator with $(C, C')$, where $C'$ is a ciphertext of a random element in $\mathcal{R}$ under $pk$, instead of an encryption of $b$ (see Game $G_2$.). Thanks to the IND-CPA property of $\Pi_+$, this game is indistinguishable from the previous one.

**Game** $G_{10}$**.** The simulation runs the simulator for the re-encryption process with $C^+_b$ and a ciphertext of random element in $\mathcal{R}$ under $pk'$, instead of an encryption of $b$, and again, because $\Pi_+$ is IND-CPA, this game is indistinguishable from the previous one.

**Game** $G_{11}$**.** In the multiplicative to additive direction, the simulator of the first ESP is run with the first component of $C$ and a ciphertext of a random element in $\mathcal{R}^*$ under $pk^\times$. Since $\Pi_\times$ is IND-CPA, this game is indistinguishable from the previous one.

**Game** $G_{12}$**.** The simulation now runs the EZT simulator with the second component of $C$ and a ciphertext of a random element of $\mathcal{R}$ instead of an encryption of $\bar{b}$. Because $\Pi_+$ is IND-CPA, this game is indistinguishable from the previous.

**Game** $G_{13}$**.** The simulation now uses the procedure $\mathsf{Sim}_{\mathsf{Share}}$ to simulates Bob's keys. By the zero-knowledge property of the 2-party decryption, this game is indistinguishable from the previous one and the adversary is in an environment completely simulated by $\mathsf{Sim}$.

**ZK for Bob.** The proof that the protocols are zero-knowledge for Bob follows the same lines. It is a bit simpler since Bob has less contribution in the additive to multiplicative direction and the switch the other way around is essentially symmetric. □

## C Uniform Sampling in a Cyclic Group of unknown order With a Folded Discrete Gaussian Distribution

Let $\sigma > 0$ a real number and let us denote by $\rho_\sigma$ the Gaussian centered function: $\rho_\sigma(x) = \exp(-\pi||x||^2/\sigma^2)$ for $x \in \mathbf{R}^n$. For a lattice $\Lambda$, we note $\rho_\sigma(\Lambda) = \sum_{x\in\Lambda} \rho_\sigma(x)$ and define the probability mass function $\mathcal{D}_{\Lambda,\sigma}$ over $\Lambda$ by

$$\forall x \in \Lambda, \mathcal{D}_{\Lambda,\sigma}(x) = \rho_\sigma(x)/\rho_\sigma(\Lambda).$$

The smoothing parameter was defined by Micciancio and Regev [35]. For a lattice $\Lambda$, and a real $\epsilon > 0$, the smoothing parameter $\eta_\epsilon(\Lambda)$ is the smallest $s$ such that $\rho_{1/s}(\Lambda^\star \setminus \{0\}) \leqslant \epsilon$.

From [35, Lemma 3.3], we have :

**Fact 1**

$$\eta_\epsilon(\Lambda) \leqslant \sqrt{\frac{\ln(2n(1+1/\epsilon))}{\pi}}\lambda_n(\Lambda).$$

The following result is implicit in [35] and made explicit in [21, Corollary 2.8].

**Fact 2** Let $\Lambda, \Lambda'$ be $n$-dimensional lattices of same rank, with $\Lambda' \subseteq \Lambda$. Then for any $\epsilon \in \mathbf{R}, 0 < \epsilon < 1/2$, any $\sigma \geqslant \eta_\epsilon(\Lambda')$, $(\mathcal{D}_{\Lambda,\sigma} \mod \Lambda')$ is within statistical distance at most $2\epsilon$ of the uniform distribution over $(\Lambda \mod \Lambda')$.

**Lemma 1.** *Let $G$ be a cyclic group of order $q$, generated by $g$. Consider the random variable $X$ with values in $G$ with uniform distribution:* $\Pr[X = h] = \frac{1}{q}$ *for all $h$ in $G$, and $Y$ the random variable with values in $G$ defined as follows. Draw $y$ from $\mathcal{D}_{\mathbf{Z},\sigma}$, with $\sigma \geqslant q\sqrt{\frac{\ln(2(1+1/\epsilon))}{\pi}}$, and define $Y = g^y$. Then, $\Delta(X, Y) \leqslant 2\epsilon$.*

*Proof.* Let $X'$ the random variable with values in $\{0, \ldots, q-1\}$ with uniform distribution and $Y'$ defined by $Y' = (y \mod q)$ where $y$ is drawn from $\mathcal{D}_{\mathbf{Z},\sigma}$. Clearly, $\Delta(X, Y) = \Delta(X', Y')$.

We apply the Fact 1 with $n = 1$ and $\Lambda' = q\mathbf{Z}$. As $\lambda_n(q\mathbf{Z}) = q$, we get

$$\eta_\epsilon(q\mathbf{Z}) \leqslant q\sqrt{\frac{\ln(2(1+1/\epsilon))}{\pi}}.$$

Then we apply Fact 2 with $\Lambda = \mathbf{Z}$ and $\Lambda' = q\mathbf{Z}$. For any $\sigma \geqslant q\sqrt{\frac{\ln(2(1+1/\epsilon))}{\pi}}$, $\Delta(X', Y') \leqslant 2\epsilon$.

## D Sharing Secret over the Integers

As one of the protocols involved in our construction needs a group of unknown order, sampling the secret key exponents and their shares is an issue. We have to use secret sharing of integers in a public interval. This problem has already been addressed, for example in [14, Section 4].

We adapt here this solution to fit our special case of sharing between two parties. Let $s \in \{-N, \ldots, N\}$ be an integer to be shared. Let $\lambda$ be a security parameter. Let $s_A$ (resp. $s_B$) be the share of Alice (resp. of Bob). The idea is to take $s_A$ uniform in a very large interval in order to make the distribution $s_B = s - s_A$ almost independent of $s$. More precisely, the share $s_A$ is drawn uniformly in $\{-2^\lambda N, \ldots, 2^\lambda N\}$ and $s_B := s - s_A$.

Let us show that the scheme is private. Let $s' \in \{-N, \ldots, N\}$ be another secret shared as $(s'_A, s'_B)$ with $s'_A$ uniform in $I := \{-2^\lambda N, \ldots, 2^\lambda N\}$ and $s'_B := s' - s'_A$. The scheme is private if $s_A$ (resp. $s_B$) is indistinguishable from $s'_A$ (resp. $s'_B$). For $s_A$ and $s'_A$ it is clear. The share $s_B$ and $s'_B$ follow the uniform distribution on $I$ respectively translated by $-s$ and $-s'$. We next show that the statistical distance $\Delta(s_B, s'_B)$ between the random variables $s_B$ and $s'_B$ is negligible. Denote $p = 1/(2^{\lambda+1}N + 1) = \Pr[s_A = z]$ for $z \in I$. The statistical distance $\Delta(s_B, s'_B)$ will be maximal if $|s - s'|$ is maximal, for example, wlog, if $s = N$ and $s' = -N$. Let $J = \{z \in \mathbf{Z}, \Pr[s_B = z] > \Pr[s'_B = z]\}$. It is clear that $J = \{-2^\lambda N - N, \ldots, -2^\lambda N + N - 1\}$ and that for $z \in J$, $\Pr[s_B = z] = p$ and $\Pr[s'_B = z] = 0$. As a result as $\Delta(s_B, s'_B) = \sum_{z \in J} \Pr[s_b = z] - \Pr[s'_B = z]$, one has $\Delta(s_B, s'_B) = 2Np < 2N/(2^{\lambda+1}N) = 2^{-\lambda}$ which is negligible.

In Appendix C, we saw that Discrete Gaussian distribution can improve uniform sampling in certain cases. However for sharing over the integer, there is no gain: suppose that $s_A$ is taken from $\mathcal{D}_{\mathbf{Z},\sigma}$ for some $\sigma$ and $s_B = s - s_A$. As before, we compute the statistical distance between two shares of different secrets $s$ and $s'$, and this still reduces to computing the distance between $\mathcal{D}_{\mathbf{Z},\sigma} - s$ and $\mathcal{D}_{\mathbf{Z},\sigma} - s'$. Let $t = |s' - s|$. This statistical distance is the same than between $\mathcal{D}_1 := \mathcal{D}_{\mathbf{Z},\sigma}$ and $\mathcal{D}_2 := \mathcal{D}_{\mathbf{Z},\sigma} + t$. Let $J \subset \mathbf{Z}$ be the subset of integers $z$ such that $\mathcal{D}_1(z) > \mathcal{D}_2(z)$. Equality occurs when $z^2 = (z-t)^2$, i.e., when $z = t/2$. So $J = \{z < t/2\}$. As a result $\Delta(D_1, D_2) = \sum_{z < t/2} \mathcal{D}_1(z) - \mathcal{D}_2(z) = \sum_{-t/2 \leqslant z < t/2} \mathcal{D}_1(z) \leqslant t/\rho_\sigma(\mathbf{Z})$, where the equality comes with the cancellation of the terms, and the inequality by upper bounded all the terms by the value in 0. Then, as $\rho_\sigma(\mathbf{Z}) > \sigma$, we eventually find that $\Delta(D_1, D_2) < t/\sigma = |s' - s|/\sigma < 2N/\sigma$.

As a result the standard deviation must be chosen as $\sigma = 2^{\lambda+1}N$ in order to make that distance negligible, so we obtain something similar to what we saw with the uniform distance.