Encryption Switching Protocols

Geoffroy Couteau[1(✉)], Thomas Peters[2], and David Pointcheval[1]

[1] ENS, CNRS, INRIA, PSL Research University, Paris, France
geoffroy.couteau@ens.fr
[2] UCLouvain, ICTEAM, Louvain-la-Neuve, Belgium
thomas.peters@uclouvain.be

Abstract. We formally define the primitive of *encryption switching protocol* (ESP), allowing to switch between two encryption schemes. Intuitively, this two-party protocol converts given ciphertexts from one scheme into ciphertexts of the same messages under the other scheme, for any polynomial number of *switches*, in any direction. Although ESP is a special kind of two-party computation protocol, it turns out that ESP implies general two-party computation (2-PC) under natural conditions. In particular, our new paradigm is tailored to the evaluation of functions over rings. Indeed, assuming the compatibility of two additively and multiplicatively homomorphic encryption schemes, switching ciphertexts makes it possible to efficiently reconcile the two internal laws. Since no such pair of public-key encryption schemes appeared in the literature, except for the non-interactive case of fully homomorphic encryption which still remains prohibitive in practice, we build the first multiplicatively homomorphic ElGamal-like encryption scheme over $(\mathbb{Z}_n, \times)$ as a complement to the Paillier encryption scheme over $(\mathbb{Z}_n, +)$, where n is a strong RSA modulus. Eventually, we also instantiate secure ESPs between the two schemes, in front of malicious adversaries. This enhancement relies on a new technique called *refreshable twin ciphertext pool*, which we show being of independent interest. We additionally prove this is enough to argue the security of our general 2-PC protocol against malicious adversaries.

1 Introduction

The development of the Internet witnessed the explosive growth of the amount of available data. We now live in an era of big data in which there is an always increasing need for efficient tools to store and manipulate huge quantities of information. While most companies now outsource their data to get an arbitrarily large storage capacity with efficient access, manipulating data in the Cloud raises many security issues. Secure multi-party computation (MPC) has thus gained tremendous importance by providing privacy-preserving tools allowing manipulations of sensitive inputs.

T. Peters—Work done while being at ENS under the ERC CryptoCloud Project.

M. Robshaw and J. Katz (Eds.): CRYPTO 2016, Part I, LNCS 9814, pp. 308–338, 2016.
DOI: 10.1007/978-3-662-53018-4_12

Secure Two-Party and Multiparty Computation. Secure two-party computation (2-PC) targets the following problem: Alice and Bob, modeled as probabilistic polynomial-time algorithms, wish to jointly compute a public function f of their respective inputs x and y, while keeping them private. We will focus on the case where Alice only gets the final result $f(x, y)$, while Bob should learn nothing, but this is not really a loss of generality. To this end, they perform an *interactive protocol*, that is expected to be *correct* (*i.e.*, the final output of the protocol is indeed $f(x, y)$) and *private* (*i.e.*, no one can learn from his own view any information that he could not have deduced from his input, and the outcome $f(x, y)$ for Alice). Secure multiparty computation is the natural extension of this problem to more than two players. Two kinds of adversarial behaviors are usually considered: *semi-honest* adversaries (*a.k.a.* honest-but-curious) follow the specifications of the protocol and try to get as much information as possible from the transcript, while *malicious* adversaries might deviate from these specifications in any way to gain more information.

Starting with the seminal work of Yao [41], there have been a vast amount of publications targeting secure two-party and multiparty computation. Today's most efficient schemes are based on various paradigms, such as secret sharing with preprocessing (e.g. TinyOT [32], SPDZ [11], MiniMac [12]), oblivious transfers [1], garbled circuits [28], or homomorphic encryption [10]. In addition, there are several hybrid constructions which combine various approaches (e.g. garbled circuit and homomorphic encryption in [21], secret sharing and garbled circuits in [13]). Most of those schemes are very efficient when the circuit to be computed is of low depth. However, when high-depth circuits are involved, the efficiency drops down: protocols based on secret sharing, oblivious transfers, partially homomorphic encryption, or garbled circuits have a communication proportional to the depth of the circuit. At the exception of the latter one, they also have a round complexity proportional to the depth of the circuit. This can be avoided with somewhat homomorphic encryption, but as soon as the circuit has a high depth, the players will have to rely on prohibitively expensive bootstrapping procedures. In the honest-but-curious setting, hybrid protocols might provide efficient solutions in some particular cases (although they will still suffer from comparable downsides in general, as they combine approaches which do all have such downsides). However, enhancing hybrid protocols *efficiently* to security against malicious adversaries is highly non-trivial, due to the lack of a common structure between the various elements manipulated in those protocols; in fact, [13,21] do only consider the honest-but curious setting.

Switching Between Homomorphic Schemes. The existence of very efficient MPC protocols for circuits containing a large number of additions, and few multiplications, suggests that multiplications might be way more expensive than additions. However, there exist encryption schemes which are *multiplicatively homomorphic*, the most famous one being the ElGamal encryption scheme [14]. In such cryptosystems, multiplications come essentially for free, but additions cannot be performed (unless a fully homomorphic scheme is used). Therefore, a natural way to design a MPC protocol in which multiplications would not incur a significant

overhead compared to additions would be to *combine* a multiplicative cryptosystem with an additive cryptosystem: multiplications would be performed homomorphically on multiplicative ciphertexts, and additions on additive ciphertexts. The missing ingredient in such a protocol is a procedure to *convert* a multiplicative (resp. additive) ciphertext into an additive (resp. multiplicative) ciphertext encrypting the same plaintext: an *encryption switching protocol*.

To our knowledge, three papers have considered switching between ciphertexts under different homomorphic schemes in the past. The concept was initially introduced in [17], where the authors propose a variant of the ElGamal encryption scheme to work over $\mathbb{Z}_n^*$, together with a protocol to switch between this scheme and the Paillier scheme. In [40], a trusted software is used to switch between various homomorphic schemes. In a recent unpublished paper [27], the authors propose methods to switch from the ElGamal scheme to the Paillier scheme, to evaluate DNF formulae.

As [40] relies on a trusted software, it cannot be compared to our work, which does not make this assumption. Moreover, we found both [17,27] to be flawed: in [17], a variant of the ElGamal encryption scheme is proposed; however, the public key of the scheme contains a square root β of unity with Jacobi symbol -1. But then, computing $\gcd(\beta - 1, n)$ gives a non-trivial factor of n. Hence, the scheme leaks the factorization of the modulus. In [27], the following variant of the ElGamal scheme is proposed: to encrypt $m \in \mathbb{Z}_n^*$, pick a random scalar r in $\mathbb{Z}_n^*$ and output $(g^r \bmod n, mh^r \bmod n)$, where g is a square ($g = 16$ in the article) and h is g^x for some secret key x. Given a ciphertext (c_0, c_1), any player can compute the Jacobi symbol of c_0 and c_1, and check whether they are equal or different. The former case corresponds to the Jacobi symbol of m being 1, while the latter case corresponds to the Jacobi symbol of m being -1: the scheme leaks the Jacobi symbol of the plaintext, which contradicts the semantic security, at least in $\mathbb{Z}_n^*$.

Indeed, constructing a multiplicatively homomorphic variant of the ElGamal encryption scheme that is still semantically secure over $\mathbb{Z}_n^*$ (and *a fortiori* over $\mathbb{Z}_n$) turns out to be a non-trivial task.

Our Contribution. In this work, we formally define *encryption switching protocol* (ESP), which allows two players to interactively and obliviously convert an encryption of a message m with a cryptosystem Π_1 to an encryption of the same message with a cryptosystem Π_2, provided that m lies in the intersection of the plaintext spaces of the cryptosystems. To instantiate this primitive, we introduce (and formally prove the security of) a new multiplicatively homomorphic variant of the ElGamal encryption scheme whose plaintext space is $\mathbb{Z}_n^*$. To our knowledge, our scheme is the first secure construction of a multiplicatively homomorphic IND-CPA encryption scheme over $\mathbb{Z}_n^*$ and might be of independent interest. We extend our variant of the ElGamal cryptosystem to a space which is "nearly" equal to $\mathbb{Z}_n$, in a sense that we formally define. We then construct *encryption switching protocols* between our new scheme and the Paillier encryption scheme. Our ESPs (between the two encryption schemes, in both directions) have a

constant communication (counted as a number of group elements), and their security relies on standard assumptions (the decisional composite residuosity, the decisional Diffie-Hellman, and the quadratic residuosity assumptions). In addition to its application to two-party computation, which will be outlined afterward, we believe that the primitive of ESP is of theoretical interest on its own.

To demonstrate the generality of our approach, we construct a generic two-party computation protocol over a ring $(\mathscr{R}, \oplus, \otimes)$ assuming the existence of homomorphic cryptosystems for each law, $\oplus$ and $\otimes$, and encryption switching protocols. We formally prove that our generic protocol achieves the standard security notions for two-party computation. Our new paradigm is particularly suited for high depth circuits.

We then turn our attention to the malicious setting. The natural way to provide security against malicious adversaries is to ask each player to prove, using a zero-knowledge proof, that he behaved honestly. However, ESPs can be seen as hybrid protocols, as they combine primitives with very different structures (in our case, the ElGamal scheme and the Paillier scheme). As is often the case in hybrid schemes, the lack of a common algebraic structure between the schemes prevents us from using standard zero-knowledge proofs. We tackle this issue by introducing a new technique for zero-knowledge, which we call a *refreshable twin-ciphertext pool*. In addition to providing an efficient way to enhance the security of ESPs to the malicious setting, we show that our new technique allows us to improve over several classical zero-knowledge proofs, such as proofs of knowledge of a double logarithm, or proof of primality of a committed value, which is of independent interest.

A nice feature of our two-party computation paradigm is that it is in fact *sufficient* to instantiate it with an ESP secure against malicious adversaries for the full generic two-party computation protocol to be secure against malicious adversaries.

Related Work. We already mentioned (and argued the insecurity of) [17,27] which design methods for switching between homomorphic schemes, and [40], which relies on a trusted software to achieve a comparable goal. Fully homomorphic encryption (FHE), gathering both additive and multiplicative homomorphic properties in a single encryption scheme, has been a long standing open problem until the seminal work of Gentry [18]. It relies on a somewhat homomorphic encryption scheme, that allows to perform a bounded number of operations, and a technique called bootstrapping to remove this bound. Our work can be seen as a similar line of work, using homomorphic encryption schemes (HEs) to perform an unlimited number of *specific* operations, and then relying on a *switching* technique to replace one HE by another one to get access to other specific operations. However, a fundamental difference is that the bootstrapping is a *non-interactive* technique, while our encryption switching protocols are interactive.

We stress that our ESP primitive makes use of shared decryption keys to obliviously decrypt and re-encrypt under the other encryption scheme, with a similar public key. This is totally different from proxy re-encryption, where the

proxy knows a key to convert a ciphertext under one key into a ciphertext under another independent key. For instance, disclosure of secret key of one encryption scheme in our realization breaks the semantic security of the other one too.

Preliminaries. Because of lack of space, basics on classical tools are postponed to the full version [6] (as well as the optimizations and detailed proofs), and the reader is recommended to refer to it for more details. But in short, a public-key encryption scheme Π is defined by the four algorithms $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$, where the two first generate the global parameters and the keys, and the two others encrypt and decrypt. If nothing else is specified we assume that a correctly encrypted message is always returned back by the decryption algorithm. We denote $\mathscr{M}$ the message space.

Throughout this paper, κ denotes the security parameter. The notation $x \xleftarrow{\$} S$ indicates that x is sampled uniformly at random from the finite set S. We write $a = b \bmod n$ to specify that $a = b$ in $\mathbb{Z}_n$ and we write $a \leftarrow [b \bmod n]$ to affect the smallest non-negative integer to a so that $a = b \bmod n$.

2 Two-Party Computation from ESPs

We introduce a theoretical framework for alternating between different encryption schemes: the new primitive of *encryption switching protocol* (ESP) allows to switch a ciphertext under an encryption scheme into a ciphertext of the same message under the other encryption scheme without damaging their semantic security. We define this primitive as a 2-party protocol and we show that secure ESP implies secure general 2-party computation under natural conditions. This is the first main contribution of the paper.

2.1 Definitions

Definition 1 (Twin-Ciphertext Pair). *For $i = 1, 2$, let Π_i be an encryption scheme $(\mathsf{Setup}_i, \mathsf{KeyGen}_i, \mathsf{Enc}_i, \mathsf{Dec}_i)$ with plaintext space $\mathscr{M}_i$. A* twin-ciphertext pair *(c_1, c_2) is a pair of ciphertexts so that:*

1. *c_1 is an encryption of $m_1 \in \mathscr{M}_1$ under Π_1;*
2. *c_2 is an encryption of $m_2 \in \mathscr{M}_2$ under Π_2;*
3. *$m_1 = m_2$ (which in turn belongs to $\mathscr{M}_1 \cap \mathscr{M}_2$).*

Given an encryption c of a message $m \in \mathscr{M}_1 \cap \mathscr{M}_2$, under one of the two above encryption schemes, we will say that any ciphertext c' which does encrypt m under the other encryption scheme is a twin ciphertext *of c.*

On the other hand, if c and c' encrypt the same m under the same encryption scheme, they are said *equivalent*. Informally, given a ciphertext c of a plaintext m under one of the two above encryption schemes, an *encryption switching protocol* (ESP) describes how users A and B, sharing the decryption key, can interact to

construct a twin ciphertext of c. This is of course under the restriction that the plaintext m lies in the intersection of the two message spaces. We focus on two encryption schemes that use common Setup and KeyGen algorithms for generating the global parameters and the keys[1].

Definition 2 (Encryption Switching Protocol). *For $i = 1, 2$, let Π_i be an encryption scheme $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}_i, \mathsf{Dec}_i)$. An* encryption switching protocol *(ESP) between Π_1 and Π_2, noted $\Pi_1 \rightleftharpoons \Pi_2$, is a tuple $(\mathsf{Share}, \mathsf{Switch})$:*

$\mathsf{Share}(\mathsf{pk}, \mathsf{sk})$ *given the common keys* sk *and* pk *of both schemes, it outputs a secret sharing $(\mathsf{sk}_A, \mathsf{sk}_B)$ of sk and updates pk if necessary. The party A (resp. B) is intended to be given sk_A (resp. sk_B);*

$\mathsf{Switch}_{\mathsf{par}}((\mathsf{pk}, \mathsf{sk}_A, c), (\mathsf{pk}, \mathsf{sk}_B, c))$ *is an interactive protocol in the direction $\mathsf{par} \in \{1{\to}2, 2{\to}1\}$ which, from a ciphertext c under the source encryption scheme, jointly computes a twin ciphertext c' of c under the target encryption scheme or outputs $\perp$ (in case of problems during the protocol execution).*

Correctness. An ESP $\Pi_1 \rightleftharpoons \Pi_2 = (\mathsf{Share}, \mathsf{Switch})$ is *correct* if both Π_1 and Π_2 are correct encryption schemes, and for any $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\kappa)$, any keys $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$, any key shares $(\mathsf{pk}, \mathsf{sk}_A, \mathsf{sk}_B) \leftarrow \mathsf{Share}(\mathsf{pk}, \mathsf{sk})$, any message $m \in \mathscr{M}_1 \cap \mathscr{M}_2$, and any $c_i \leftarrow \mathsf{Enc}_i(\mathsf{pk}_i, m)$ for $i = 1, 2$,

$$\mathsf{Dec}_2(\mathsf{sk}, \mathsf{Switch}_{1\to 2}((\mathsf{pk}, \mathsf{sk}_A, c_1), (\mathsf{pk}, \mathsf{sk}_B, c_1))) = m,$$
$$\mathsf{Dec}_1(\mathsf{sk}, \mathsf{Switch}_{2\to 1}((\mathsf{pk}, \mathsf{sk}_A, c_2), (\mathsf{pk}, \mathsf{sk}_B, c_2))) = m,$$

always hold. Ciphertexts on messages in the intersection of the two plaintext spaces are called *switchable*.

2.2 Security Notions

We expect ESP not to break the $\mathsf{IND\text{-}CPA}$ security of the encryption schemes, even in front of malicious adversaries: the adversary $\mathscr{A}$ is given pk, but since it plays against Alice or Bob it can choose either sk_B or sk_A, respectively. Then, even interacting with an oracle that emulates the other party as an honest player, $\mathscr{A}$ should not be able to break $\mathsf{IND\text{-}CPA}$ security of neither Π_1 nor Π_2. Let us more formally define this security notion.

Definition 3 ($\mathscr{O}_A$ and $\mathscr{O}_B$ Oracles). *For appropriate keys $(\mathsf{pk}, \mathsf{sk}_A, \mathsf{sk}_B)$, we denote the stateful oracle $\mathscr{O}_A(i{\to}j, c, \mathsf{Flow})$ that emulates the honest player A: it provides the answers A would send back upon receiving the flow Flow when running the protocol $\mathsf{Switch}_{i\to j}((\mathsf{pk}, \mathsf{sk}_A, c), (\mathsf{pk}, \mathsf{sk}_B, c))$. We similarly define the oracle $\mathscr{O}_B$ that emulates the honest player B. A special flow 'Start' is used to initialize the protocol.*

[1] In any case, we could just take the concatenation of the outputs of the algorithms of the two schemes.

In our target application of 2-PC, these oracles will not be available on any input, but on controlled ciphertexts only. Hence our following security notion.

Definition 4 (ESP Security). *An encryption switching protocol* $\Pi_1 \rightleftharpoons \Pi_2$ *is* ***secure*** *if it is* strongly sound *and* zero-knowledge *(see below).*

The soundness property guarantees that no malicious player can successfully force the outcome of Switch not to be a twin ciphertext of the input, when the input is indeed a switchable ciphertext. The *strong* requirement means that the soundness holds even if the adversary is also given the whole secret key sk (or both sk_A and sk_B), instead of just one of the two shares.

Definition 5 (Strong Soundness). *An encryption switching protocol* $\Pi_1 \rightleftharpoons \Pi_2$ *is* ***strongly sound****, if it is strongly sound for A and strongly sound for B. The scheme is strongly sound for B, if for any* $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\kappa)$*, any keys* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$*, any secret key shares* $(\mathsf{pk}, \mathsf{sk}_A, \mathsf{sk}_B) \leftarrow \mathsf{Share}(\mathsf{pk}, \mathsf{sk})$*, for all PPT adversary* $\mathscr{A}$ *playing the role of A, the success*

$$\mathsf{Succ}_B^{\textit{esp-sound}}(\mathscr{A}) = \Pr[\mathit{BadSwitch} | \, \mathscr{A}^{\mathscr{O}_B(\cdot,\cdot,\cdot)}(\mathsf{pk}, \mathsf{sk}_A, \mathsf{sk})]$$

is negligible, where the event BadSwitch *is raised when a full protocol execution of* Switch *with* $\mathscr{O}_B$ *on a switchable input ciphertext c successfully outputs* $c^\star$ *which is not a twin ciphertext of c. (In a non-strong version of soundness the adversary is only given* $(\mathsf{pk}, \mathsf{sk}_A)$*.) We denote* $\mathsf{Succ}^{\mathsf{esp\text{-}sound}}(\kappa, t)$ *the maximal success an adversary can get against A or B within time t.*

The zero-knowledge property guarantees that no information leaks about the secret key shares to a malicious player when switches are performed on switchable ciphertexts: its view can be simulated without any additional information than its own secret share.

Definition 6 (Zero-Knowledge). *An encryption switching protocol* $\Pi_1 \rightleftharpoons \Pi_2$ *is* ***zero-knowledge****, if it is zero-knowledge for A and zero-knowledge for B. The scheme is zero-knowledge for B if there exist two efficient simulators,* $\mathscr{S}im_B^{\mathsf{share}}$ *and* $\mathscr{S}im_B^{\mathsf{ESP}}$ *of* Share *and the oracle* $\mathscr{O}_B$ *respectively, with the following property: for any* $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\kappa)$*, any keys* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$*, any secret key shares* $(\mathsf{pk}, \mathsf{sk}_A, \mathsf{sk}_B) \leftarrow \mathsf{Share}(\mathsf{pk}, \mathsf{sk})$ *or simulated shares* $(\mathsf{pk}', \mathsf{sk}'_A) \leftarrow \mathscr{S}im_B^{\mathsf{share}}(\mathsf{pk})$*, and for any PPT adversary* $\mathscr{A}$ *playing the role of A, the advantage*

$$\mathsf{Adv}_B^{\mathsf{esp\text{-}zk}}(\mathscr{A}) = \left| \Pr[1 \leftarrow \mathscr{A}^{\mathscr{O}'_B(\cdot,\cdot,\cdot,\cdot)}(\mathsf{pk}, \mathsf{sk}_A)] - \Pr[1 \leftarrow \mathscr{A}^{\mathscr{S}im_B(\cdot,\cdot,\cdot,\cdot)}(\mathsf{pk}', \mathsf{sk}'_A)] \right|$$

is negligible, where the adversary $\mathscr{A}$ *is given unbounded access to either the simulator* $\mathscr{S}im_B$ *or the stateful oracle* $\mathscr{O}'_B$ *described below, with the restriction that input ciphertexts* $(c, \bar{c})$ *to* $\mathscr{S}im_B$ *or* $\mathscr{O}'_B$ *are twin ciphertexts:*

Oracle $\mathscr{O}'_B(_{i\to j}, c, \bar{c}, \mathit{Flow})$*: on input a direction* $_{i\to j}$*, a ciphertext c under the encryption scheme* Π_i*, a ciphertext* $\bar{c}$ *under the encryption scheme* Π_j*, and a message flow* Flow*, ignores* $\bar{c}$ *and runs* $\mathscr{O}_B(_{i\to j}, c, \mathit{Flow})$*;*

Simulator $\mathscr{S}im_B(i{\to}j, c, \bar{c}, \mathsf{Flow})$*: on the same inputs as above, emulates the output an honest player* B *would answer upon receiving the flow* Flow *when running the protocol* $\mathsf{Switch}_{i\to j}((\mathsf{pk}, \mathsf{sk}_A, c), (\mathsf{pk}, \mathsf{sk}_B, c))$*, without* sk_B *but possibly with* sk_A*, and forcing the output to be a ciphertext* $\bar{c}'$ *equivalent to* $\bar{c}$ *(*i.e.*, a ciphertext* $\bar{c}'$ *such that* $\mathsf{Dec}(\mathsf{sk}, \bar{c}) = \mathsf{Dec}(\mathsf{sk}, \bar{c}')$*).*

If the adversary $\mathscr{A}$ *can be unbounded,* $\Pi_1 \rightleftharpoons \Pi_2$ *is statistically zero-knowledge. We denote* $\mathsf{Adv}^{\mathsf{esp\text{-}zk}}(\kappa, t)$ *the maximal advantage an adversary can get against* A *or* B *within time* t*.*

At a high level, Definition 4 says that (misbehaving) players A and B separately gain no information on the plaintexts even if they can switch the ciphertexts between Π_1 and Π_2. In that sense, switching ciphertexts is a special kind of two-party computation. It is pretty clear that a secure ESP on appropriate encryption schemes allows to build two-party protocols in $\mathscr{M}_1 \cap \mathscr{M}_2$.

2.3 Computational Equality

Let us consider an adversary $\mathscr{A}$ which can efficiently sample messages in both the intersection of the message spaces $\mathscr{M}_1 \cap \mathscr{M}_2$ and their symmetric difference $\mathscr{M}_1 \oplus \mathscr{M}_2 = (\mathscr{M}_1 \cup \mathscr{M}_2) \backslash (\mathscr{M}_1 \cap \mathscr{M}_2)$. A simple observation shows that a secure ESP could not be safe to use inside a larger protocol, even in front of a passive adversary, since the switching protocol does not provide any guarantee on non-switchable ciphertexts, that encrypt messages outside $\mathscr{M}_1 \cap \mathscr{M}_2$. They could help to distinguish ciphertexts. More generally, we would like Switch not to help for distinguishing *switchable* ciphertexts from *non-switchable* ciphertexts, which would break the $\mathsf{IND\text{-}CPA}$ security with the Switch oracle.

A solution could be a restriction on the choice of the ciphertexts asked to the Switch oracles, so that the plaintexts lie in $\mathscr{M}_1 \cap \mathscr{M}_2$. But this would not be strong enough for practical purpose, since there is no reason that it cannot happen during a complex evaluation. We thus define the following additional property, to be satisfied by the message spaces, with the common public key pk as auxiliary input:

Definition 7 (Computational Equality). *Let* $(\mathscr{M}_1, \mathscr{M}_2, \mathsf{aux})$ *be two sets and some additional information.* $\mathscr{M}_1$ *and* $\mathscr{M}_2$ *are computationally equal given auxiliary input* aux *if, for any adversary* $\mathscr{A}$*, its success probability for outputting a message in the symmetric difference* $\mathscr{M}_1 \oplus \mathscr{M}_2$*, denoted* $\mathsf{Succ}^{\mathsf{comp\text{-}eq}}(\mathscr{A}) = \Pr[m \leftarrow \mathscr{A}(\mathscr{M}_1, \mathscr{M}_2, \mathsf{aux}) : m \in \mathscr{M}_1 \oplus \mathscr{M}_2]$*, is negligible.*

We have defined the security of ESP for switchable inputs and, informally, the computational equality will guarantee that non-switchable inputs are quite unlikely during the execution of a protocol involving ESPs.

2.4 Ring-Homomorphic Encryption Schemes

Toward our aim of getting two-party computation protocols from ESP, our goal is to design two encryption schemes on a ring structure $(\mathscr{R}, \oplus, \otimes)$, where the

encryption algorithms are homomorphic on the plaintexts (under either $\oplus$ or $\otimes$) and on the random coins (with an appropriate group law $\odot$ over the randomness space R which may differ in every case), using the combinations $\boxplus$ and $\boxtimes$ of the ciphertexts:

$$\begin{aligned} \mathscr{E}_{\oplus}(m_1; r_1) \boxplus \mathscr{E}_{\oplus}(m_2; r_2) &= \mathscr{E}_{\oplus}(m_1 \oplus m_2; r_1 \odot r_2) \\ \mathscr{E}_{\otimes}(m_1; r_1) \boxtimes \mathscr{E}_{\otimes}(m_2; r_2) &= \mathscr{E}_{\otimes}(m_1 \otimes m_2; r_1 \odot r_2) \end{aligned} \tag{1}$$

In particular, this implies that we can maul any ciphertext of m into a ciphertext of $R \otimes m$, for a *known* R, with an appropriate operation $\bullet$ in each case (and the appropriate operation $\cdot$ on the random coins) on the ciphertexts:

$$R \bullet \mathscr{E}_{\oplus}(m; r) = \mathscr{E}_{\oplus}(R \otimes m; R \cdot r) \quad R \bullet \mathscr{E}_{\otimes}(m; r) = \mathscr{E}_{\otimes}(R \otimes m; R \cdot r). \tag{2}$$

Note that we explicitly choose $\boxplus$, $\boxtimes$ and $\bullet$ to be deterministic functions, so that any local homomorphic evaluation on ciphertexts leads to the same ciphertext result. Note also that the existence of $\boxplus$ and $\boxtimes$ implies the stability of the plaintexts spaces of $\mathscr{E}_{\oplus}()$ and $\mathscr{E}_{\otimes}()$, under $\oplus$ and $\otimes$ respectively.

2.5 General Secure Two-Party Computation

The reason of designing ESP is to take advantage of the nice (homomorphic) properties of the two schemes which may not be available in a single *efficient* encryption scheme. When additions $\oplus$ are required, we use ciphertexts under the additively homomorphic encryption scheme *w.r.t.* $\boxplus$, and when multiplications $\otimes$ and exponentiations are needed, we convert the operands into the other multiplicatively homomorphic encryption scheme *w.r.t.* $\boxtimes$. In other words, ESP aims at reconciling additively and multiplicatively homomorphic schemes, to jointly compute the encryption of $f(x, y)$, for any public function f over $(\mathscr{R}, \oplus, \otimes)$, on encryptions of x and y. Below, we consider two-party computation which reveals the result to a single party only (Alice).

Secure 2-PC. More formally, assuming only Alice gets the outcome, the security game of such a privacy-preserving evaluation is the following one: The adversary against Bob chooses its input x and the possible inputs y_0, y_1 for Bob, with the additional restriction that $f(x, y_0) = f(x, y_1)$ (otherwise the outcome would reveal Bob's actual input value); It gets the encryption of x and the encryption of y_b for a random bit $b \xleftarrow{\$} \{0, 1\}$; At the end of the joint evaluation with Bob, it should try to guess b, and thus Bob's actual input value. If the adversary plays the role of Bob against Alice, then it chooses its input y and the possible inputs x_0, x_1 for Alice but without any additional restriction. When no adversary can guess b in any of the two games (against Alice or Bob), with non-negligible advantage, we say that the $\mathsf{2\text{-}PC}$ protocol is *input-indistinguishable*. This is formally defined in the full version [6].

Since we assume that Alice receives the outcome of the $\mathsf{2\text{-}PC}$ in our design we also assume that Alice and Bob are able to decrypt ciphertexts *from their shares*. Without loss of generality, we assume that Π_2 admits a 2-party decryption (as

detailed in the full version [6]) so that only Alice gets the plaintexts. A rigorous construction $\Pi_{2\mathrm{PC}}$ is proposed in the full version [6], using a secure ESP between homomorphic encryption schemes over computationally-equal message spaces, following the above intuition, leads to the next result.

Theorem 8. *Let Π_1 and Π_2 be IND-CPA (complementary) homomorphic encryption schemes over a ring $(\mathscr{R}, \oplus, \otimes)$, whose message spaces are computationally equal, equipped with a secure ESP, $\Pi_1 \rightleftharpoons \Pi_2 = (\mathsf{Share}, \mathsf{Switch})$, so that Π_2 admits a 2-party decryption for A from the same key shares output by* Share *and which is statistically sound and zero-knowledge, then the $\Pi_{2\mathrm{PC}}$ protocol is an input-indistinguishable 2-PC for any function f over $(\mathscr{R}, \oplus, \otimes)$.*

We stress that this theorem is for the malicious setting: if the ESP protocols (and the 2-party decryption) are secure against malicious adversaries, the $\Pi_{2\mathrm{PC}}$ protocol is secure against malicious adversaries, without any additional zero-knowledge proofs.

Intuition. Our approach for $\Pi_{2\mathrm{PC}}$ consists in starting from ciphertexts of x and y, and to switch to the appropriate encryption scheme in order to be able to make operations through the homomorphic property, until the encryption of the result is reached. The rationale of the *computational-equality property* for the message spaces, with the public key as auxiliary input, is the following one: on encryptions of valid inputs x and y_b, the evaluation of the encryption of $f(x, y_b)$ follows a deterministic path of switches and public homomorphic operations on the ciphertexts. In the honest-but-curious setting, the sequences of involved plaintexts is indeed determined by x and y_b, and in the malicious setting, the soundness property ensures that the same happens. Then, if all the ciphertexts are switchable, using the simulators from the zero-knowledge property of the ESP leads to the privacy of the computation: no information leaks on b. If a ciphertext happens to be non-switchable with non-negligible probability during the computation, simply generating the sequences of plaintexts from (x, y_0) and from (x, y_1) would efficiently generate an element in the symmetric difference: we need this to be intractable. Eventually, the outcome of the protocol is recovered by performing 2-party decryption.

Sketch of the Proof. The structure of the proof follows a sequence of indistinguishable games from the real game with (x, y_0), between the adversary and a simulator emulating the challenger using $b = 0$ with all the secret information to the real game with (x, y_1), and so using $b = 1$. We consider the output guess b', which should remain the same. The first games consist of a preparation for replacing y_0 by y_1. We indeed cannot apply the semantic security of the encryption schemes yet since the decryption keys are known to the simulator. But first, with the computational-equality property, we can guarantee that all the input ciphertexts of the ESPs are switchable. Then, with the soundness of the ESPs, we know that the outputs of the ESPs are twin ciphertexts. Actually, we need here the *strong* flavor of soundness since the secret key is still known. Again we apply

the soundness of the final 2-party decryption to guarantee the correct decryption (since the decryption key is still known, we require the *statistical* soundness, but a *strong* flavor would be enough too). Now that we know all the input-output pairs of the internal primitives (ESPs and decryption) are correct, we can safely replace the honest emulation using the secret key by the simulators without the secret key, thanks to the zero-knowledge property. So, the secret key is not required anymore, and we can replace y_0 by y_1, applying the IND-CPA security game to the first encryption scheme. We also have to propagate to the outputs of the ESPs, using again the IND-CPA security game of the other encryption scheme. This is done sequentially, with hybrid games, to end with a game where the input is (x, y_1) and all the intermediate ciphertexts are consistent. We can then move back to the honest emulation (and not the simulators for the ESPs and the decryption) using the secret key. The full construction is described and formally proven secure in the full version [6].

Our Next Goal. Three properties must be satisfied to securely evaluate functions over a ring: the *homomorphism* of the encryption schemes, the *security* of the ESPs and the *computational equality* of the messages spaces. Instantiating these building blocks would allow us to achieve our second objective: building an *efficient* two-party computation over a ring as a realistic alternative to standard methods, particularly for arithmetic functions with a high multiplicative depth. After discussing some applications of ESPs, we provide a first step toward our goal by designing a secure ESP to switch between two homomorphic encryption schemes over $\mathbb{Z}_n^*$.

3 Applications

In this section, we motivate our paradigm for two-party computation with some concrete examples involving high-depth circuits.

Private Disjointness Testing (PDT). Two players, Alice and Bob, holding respective databases $A = (a_i)_{i \leq a}$ and $B = (b_i)_{i \leq b}$, wish to know whether their databases have at least one common element or not, and nothing more. The state-of-the-art solution to PDT is [42], which solves the problem with complexity $O\left((a+b)^2\right)$ (counting group elements).

A natural way to solve the PDT is to view the items of A as the roots of a polynomial $P(X) = \sum_{i=0}^{a} \alpha_i X^i$. Alice and Bob perform an interactive protocol which outputs $u = r \prod_{i=1}^{b} P(b_i)$ to Alice, where r is a uniformly random value picked by Bob. If this value is 0, then one of the $P(b_i)$'s is zero, which means that one of the b_i's is in A. However, the circuit computing u is of depth $O(\log b)$, hence most 2-PC protocols computing this circuit are not constant round. Using carefully constructed circuits such as the sort-compare-shuffle circuit of [22] (adapted to the case of PDT), the (constant-round) garbled circuit approach transmits $O(\kappa \ell (a+b) \log(a+b) + \kappa b M(\kappa))$ bits, where ℓ is the size of

the items in A and B and $M(\kappa)$ the circuit size of modular multiplication (multiplications are performed modulo a κ-bit value to avoid integer multiplication while maintaining statistical correctness).

Our framework allows us to design a linear-communication constant-round protocol for the private disjointness test:

1. Alice builds the polynomial $P = \sum \alpha_i X^i$ so that $P(a_i) = 0$ for $i \leq a$, and sends $(C_i = \mathscr{E}_\oplus(\alpha_i))_i$;
2. Bob computes and sends $D_i \leftarrow \boxplus_j b_i^j \bullet C_i = \mathscr{E}_\oplus(P(b_i))$ for $i \leq b$;
3. They perform b ESPs in parallel to get $(D'_i = \mathscr{E}_\otimes(P(b_i)))_{i \leq b}$;
4. Bob picks $r \xleftarrow{\$} \mathbb{Z}_n$ and computes $E \leftarrow r \bullet \boxtimes_i D'_i = \mathscr{E}_\otimes(r \times \prod P(b_i))$.
5. Alice and Bob jointly decrypt the ciphertext, Bob gets the result and checks whether the plaintext is zero or not.

The total communication complexity of this protocol is $a+b+2$ ciphertexts and b parallel ESPs. With constant size ESPs (as we will construct in the following), this gives a total communication of $O(a+b)$ in constant round. We want to stress that this does not mean that, for concrete parameters, this approach will necessarily beat the best super-linear garbled circuits for PDT; however, garbled circuits have enjoyed decades of optimizations, and given its asymptomatic complexity, our new approach seems worth considering for further investigations and could benefit from numerous optimizations. Note also that hybrid frameworks (such as [21]) can also provide linear-communication constant-round solutions, but unlike these protocols, our approach is easily enhanced to the malicious setting: in a high level, items 1 and 2 are secure from [7] and the next items are secure against malicious adversaries if so are the ESPs performing the switches (and Sect. 6 provides an efficient technique to achieve this security).

Oblivious Multivariate Polynomial Evaluation (OMPE). This is the natural extension of oblivious polynomial evaluation [31] over multivariate polynomials [39]. Once an ESP is available, constructing an OMPE protocol is straightforward (we use the notations of [39]). Unlike previous solutions, it keeps the degree d of P hidden.

- Alice holds an N-variate polynomial P of degree d with M monomials;
- Bob holds $(x_1, \cdots, x_N)$ and sends $(\mathscr{E}_\otimes(x_i))_{i \leq N}$;
- Alice computes all the M monomials of $P(x_1, \cdots, x_N)$ encrypted under $\mathbb{Z}_n^*$-EG, due to the multiplicativity;
- Alice and Bob perform M parallel ESPs on the encrypted monomials to get the M additively encrypted monomials, and then get $\mathscr{E}_\oplus(P(x_1, \cdots, x_N))$;
- Alice and Bob jointly decrypt it, so that Bob (or both) gets $P(x_1, \cdots, x_N)$.

Our OMPE protocol transmits $O((N+M)\log n)$ bits, to be compared with $O(Nd\kappa^2)$ for [39]. In addition, our protocol can be adapted to the case of multivariate polynomials whose most compact representation is not their canonical form; for example, if the polynomial is of the form $\prod_i \sum_j X_j^{\delta_{ij}}$, extending it to its canonical form would result in an expression with exponentially many terms.

Instead, the polynomial can be directly evaluated from this compact form: first using the multiplicative homomorphism to evaluate the $X_i^{\delta_{ij}}$'s, they switch to perform the sums, and then switch again to perform the final product. Several applications of OMPE are discussed in [39], such as testing whether the union of two sets of vectors are of full rank which has applications in linear secret sharing schemes, where the secret can be recovered when a full rank set of vectors is known; the players can determine whether they could recover the secret together without revealing their set. We get a more efficient *Full-Rank Test* protocol.

4 An Encryption Switching Protocol over $\mathbb{Z}_n^*$

For the internal laws on the plaintexts in $\mathbb{Z}_n$ we keep the usual notations $+$ and $\times$ (or $\cdot$ and even nothing), but we still use the notations of the Sect. 2.4 for the external operations on the ciphertexts and the relations on the random coins.

In order to complete the Paillier encryption scheme, that is additively homomorphic in $\mathbb{Z}_n$, we build an ElGamal variant that is multiplicatively homomorphic in $\mathbb{Z}_n^*$, both for the same RSA modulus n. The security of our new variant relies on the DDH assumption in $\mathbb{J}_n$, the (maximal) cyclic subgroup of $\mathbb{Z}_n^*$ whose elements have a Jacobi symbol equal to $+1$, and the QR assumption in $\mathbb{Z}_n^*$ (see the full version [6] for more details about the structure of the ring $\mathbb{Z}_n$). In order to build a secure encryption switching protocol, we need an additional property from the two encryption schemes: they can be *randomized*. An encryption scheme $\mathscr{E}$ is randomizable if there exists an efficient algorithm Rand such that for every message m and every random coins $r \in \mathsf{R}$:

$$\{\mathscr{E}(m;r') \mid r' \xleftarrow{\$} \mathsf{R}\} \equiv \{\mathsf{Rand}(\mathscr{E}(m;r),r') \mid r' \xleftarrow{\$} \mathsf{R}\} \tag{3}$$

where $\equiv$ denotes the computational/statistical/perfect indistinguishability of the two distributions. For the sake of simplicity, we will denote $\mathsf{Rand}(C)$ the probabilistic algorithm which picks r uniformly at random and returns $\mathsf{Rand}(C;r)$.

We now recall basic computational assumptions and an implication to $\mathbb{J}_n$, then we review the Paillier encryption which also admits a verifiable 2-party decryption algorithm (where either the two players, or one player only, get the result) and we introduce our new ElGamal encryption schemes. Finally, we show how to switch between these schemes from encryptions over $\mathbb{Z}_n^*$.

4.1 Computational Assumptions

The security of our protocols will rely on the following standard assumptions:

- The DDH (Decisional Diffie-Hellman) assumption in a cyclic group $\mathbb{G} = \langle g \rangle$ of order q states that, given (g^a, g^b) for $a, b \xleftarrow{\$} \mathbb{Z}_q$, g^{ab} is indistinguishable from a random element in $\mathbb{G}$.
- The QR (Quadratic Residuosity) assumption in $\mathbb{Z}_n^*$, for an RSA modulus n, states that a random element in QR_n (square in $\mathbb{Z}_n^*$) is indistinguishable from a random element in $\mathbb{J}_n$ (element of $\mathbb{Z}_n^*$ with Jacobi symbol $+1$).

– The DCR (Decisional Composite Residuosity) assumption in $\mathbb{Z}_{n^2}^*$, for an RSA modulus n, states that a random n-th power in $\mathbb{Z}_{n^2}^*$ is indistinguishable from a random element in $\mathbb{Z}_{n^2}^*$.

The DDH assumption is usually assumed to hold in large prime-order subgroups of $\mathbb{Z}_p^*$. In the following, $n = pq$ is a **strong RSA modulus** if $p = 2p' + 1$ and $q = 2q' + 1$ are safe primes (with both p' and q' also prime). With such a modulus n, DDH is also a reasonable assumption in QR_n, since the order is $p'q'$ (see the full version [6] for more details). Adding the QR assumption in $\mathbb{Z}_n^*$, this makes the DDH assumption in $\mathbb{J}_n$ (of order $2p'q'$) reasonable too:

Theorem 9. *When $n = pq$ is a strong RSA modulus, the DDH assumption in $\mathbb{J}_n$ is implied by the DDH assumption in both the large prime-order subgroups of $\mathbb{Z}_p^*$ and $\mathbb{Z}_q^*$ and the QR assumption in $\mathbb{Z}_n^*$. (The proof is in the full version [6].)*

However, given $m \in \mathbb{Z}_n^*$, computing Jacobi symbol $J_n(m)$ is easy and then the DDH assumption does not hold in $\mathbb{Z}_n^*$ which, in addition, is non cyclic.

4.2 $\mathbb{Z}_n$-P: The Paillier Encryption Scheme on $\mathbb{Z}_n$

For the Paillier encryption scheme (denoted $\mathbb{Z}_n$-P), we will use the notation $\mathscr{E}_{\oplus}(\cdot)$ since this will be our additively homomorphic encryption scheme. It implicitly uses the strong RSA modulus $n = pq$, and we denote $\lambda = \lambda(n) = (p-1)(q-1)/2$, the maximal order of an element of $\mathbb{Z}_n^*$. One can note that $\lambda = (n-1)/2 + (2 - (p+q))/2$ is statistically close to $(n-1)/2$ or $n/2$ if we consider the Euclidean division (we will abuse this notation $n/2$ in the following).

The Paillier Cryptosystem. In [33], Paillier proposed an encryption scheme $\mathbb{Z}_n$-P for a modulus $\mathsf{pk} = n$ as public key, and $\mathsf{sk} = d \leftarrow [\lambda^{-1} \mod n] \times \lambda \bmod n\lambda$ as secret key: $\mathbb{Z}_n\text{-P.Enc}(\mathsf{pk}, m; r)$, for a message $m \in \mathbb{Z}_n$ and random coins r in $\mathbb{Z}_n^*$, outputs $c = (1+n)^m \cdot r^n \bmod n^2$; $\mathbb{Z}_n\text{-P.Dec}(\mathsf{sk}, c)$ returns $m = ([c^d \bmod n^2] - 1)/n$. (See details in the full version [6]). This scheme is IND-CPA under the DCR assumption over $\mathbb{Z}_{n^2}^*$, and it is additively homomorphic in $\mathbb{Z}_n$. It satisfies Eq. (1), $\boxplus$ being the multiplication in $\mathbb{Z}_{n^2}^*$. The randomization algorithm Rand is given by $\mathbb{Z}_n\text{-P.Rand}(c; r) = c \cdot r^n \bmod n^2$, for any random coins r in $\mathbb{Z}_n^*$.

2-Party Paillier Decryption. In this section, we briefly recall the semi-honest case where players are honest-but-curious. The reader can refer to the full version [6] for more details and a description in the malicious case which makes use of classical zero-knowledge proofs.

We assume that a trusted dealer generates the key shares for the two parties, Alice and Bob (distributed key generation can be found in [20]). The dealer generates random $d_A, d_B \in \mathbb{Z}_{n\lambda}$ subject to $d_A + d_B = d \bmod n\lambda$ defined above. Then, Alice gets d_A and Bob gets d_B.

In order to allow Bob to decrypt the ciphertext C, Alice computes and sends $C_A \leftarrow C^{d_A} \bmod n^2$, which allows Bob to get the plaintext $m \leftarrow ([C_A \times C^{d_B} \bmod$

$n^2] - 1)/n$. Note that we do intentionally not disclose m to Alice in general. But this is perfectly symmetric if one wants Alice to get the result instead of Bob.

The **correctness** of this protocol is straightforward. Let us show that it is statistically **zero-knowledge**: To emulate Alice in front of a curious Bob, we first pick d_B in $\mathbb{Z}_{n^2/2}$ instead of $\mathbb{Z}_{n\lambda}$ (since $n/2$ is statistically close to λ) and we give it to Bob. The simulator with input (m, d_B) sends $C_A \leftarrow (1 + n \cdot m) \times C^{-d_B}$, which enforces the decryption to m for Bob. This simulation is statistically indistinguishable from a real execution when C does indeed encrypt m. No emulation of Bob is needed as he does not send any message.

4.3 $\mathbb{Z}_n^*$-EG: An ElGamal Variant in $\mathbb{Z}_n^*$

The ElGamal Cryptosystem. In [14], ElGamal proposed the famous encryption scheme that applies in any cyclic group $\mathbb{G} = \langle g \rangle$ of order q, in which the DDH assumption holds: for a secret scalar $\mathsf{sk} = x \xleftarrow{\$} \mathbb{Z}_q$, the public key is $\mathsf{pk} = h \leftarrow g^x$: $\mathsf{Enc}(\mathsf{pk}, m; r)$, for a message $m \in \mathbb{G}$ and random coins r in $\mathbb{Z}_q$, outputs $c = (c_0 = g^r, c_1 = h^r \cdot m)$; $\mathsf{Dec}(\mathsf{sk}, c)$ returns $m = c_1/c_0^x$.

This scheme is IND-CPA under the DDH assumption over $\mathbb{G}$, and it is multiplicatively homomorphic in $\mathbb{G}$. ElGamal encryption scheme satisfies Eq. (1), $\boxtimes$ being the component-wise multiplication in $\mathbb{G}^2$. The randomization algorithm Rand is given by $\mathsf{Rand}(c; r) = (c_0 \cdot g^r, c_1 \cdot h^r)$, for any random coins r in $\mathbb{Z}_q$. The 2-party decryption protocol is quite similar to the above Paillier one.

In the following, we will essentially use QR_n-EG and $\mathbb{J}_n$-EG, the ElGamal encryption schemes in QR_n and $\mathbb{J}_n$ respectively.

Extension to $\mathbb{Z}_n^*$. However, our main goal is to extend the ElGamal encryption scheme to $\mathbb{Z}_n^*$. The global parameters contain the strong RSA modulus n, with a generator g of $\mathbb{J}_n$. The global setup and the algorithms are described on Fig. 1.

Description. Since the larger space that ElGamal can securely encrypt is $\mathbb{J}_n$, in order to encrypt a message $m \in \mathbb{Z}_n^*$, we have to split m into two parts, $m_1, m_2 \in \mathbb{J}_n$: given $\chi \in \mathbb{Z}_n^* \setminus \mathbb{J}_n$, a natural encoding is $m_1 = J_n(m) = (-1)^a$ and $m_2 = \chi^a m$, with an appropriate integer a. But, even if $\{\pm 1\}$ could be seen as a subgroup of $\mathbb{J}_n$, $\psi : \mathbb{Z}_2 \times \mathbb{J}_n \mapsto \mathbb{Z}_n^*$, $\psi(a, m) = \chi^{-a} m$ is not an homomorphism when the order of χ is not 2. But we cannot leave in the clear[2] a square root of 1 lying in $\mathbb{Z}_n^* \setminus \mathbb{J}_n$ (as done in [17]). However, for a generator g of $\mathbb{J}_n$, we can instead encode m with $m_1 = g^a$ and $m_2 = \chi^{-a} m$ for any integer a such that $J_n(m) = (-1)^a$, and encrypt m_2 into (C_0, C_1) using $\mathbb{J}_n$-EG, and appending m_1 in clear. The intricate point in the decryption phase will be to reconstruct χ^a from $m_1 = g^a$: if one defines $v = [p^{-1} \bmod q] \cdot p \bmod n$ and $\chi \leftarrow (1 - v) \cdot g^{t_p} + v \cdot g^{t_q} \bmod n$, for even t_p and odd t_q randomly drawn in $\mathbb{Z}_\lambda$, then $\chi \in \mathbb{Z}_n^* \setminus \mathbb{J}_n$. In addition, from $m_1 = g^a$, one gets χ^a as $(1 - v) m_1^{t_p} + v m_1^{t_q} \bmod n$. The complete description of the scheme is described on Fig. 1.

[2] Given two square roots of the same element with distinct Jacobi symbols allows efficiently factoring n.

Setup and Key Generation

- The main strong RSA modulus n:
  - p, q two safe primes, $n \leftarrow pq$;
  - $g_0 \xleftarrow{\$} \mathbb{Z}_n^*$, $g \leftarrow -g_0^2$ (a generator of $\mathbb{J}_n$, of order λ);
  - $d \leftarrow [\lambda^{-1} \mod n] \cdot \lambda \bmod n\lambda$: $d = 0 \bmod \lambda$ and $d = 1 \bmod n$;
  - $v \leftarrow [p^{-1} \mod q] \cdot p \bmod n$: $v = 0 \bmod p$ and $v = 1 \bmod q$;
  - an even $t_p \xleftarrow{\$} \mathbb{Z}_\lambda$ and an odd $t_q \xleftarrow{\$} \mathbb{Z}_\lambda$: $\chi \leftarrow (1-v) \cdot g^{t_p} + v \cdot g^{t_q} \bmod n$;
  - $s \xleftarrow{\$} \mathbb{Z}_\lambda$, and set $g_1 \leftarrow g^s \bmod n$ (for $\mathbb{J}_n$-EG).
- The additional modulus N:
  - P, Q two strong primes, $N \leftarrow PQ$ (such that $N > (2 + 2^{\kappa+1})n^2$);
  - $D \leftarrow [\Lambda^{-1} \mod N] \cdot \Lambda \bmod N\Lambda$, where Λ is the order of $\mathbb{J}_N$.
- Keys: $\mathsf{pk} \leftarrow (n, g, \chi, g_1, N)$ and $\mathsf{sk} \leftarrow (d, v, t_p, t_q, s, D)$.
- Partial keys: $(d_A, v_A, t_{pA}, t_{qA}, s_A, D_A) \xleftarrow{\$} \mathbb{Z}_{n\lambda} \times \mathbb{Z}_n \times \mathbb{Z}_\lambda^3 \times \mathbb{Z}_{N\Lambda}$, and $d_B \leftarrow d - d_A \bmod n\lambda$, $v_B \leftarrow v - v_A \bmod n$, $t_{pB} \leftarrow t_p - t_{pA} \bmod \lambda$, $t_{qB} \leftarrow t_q - t_{qA} \bmod \lambda$, $s_B \leftarrow s - s_A \bmod \lambda$, and $D_B \leftarrow D - D_A \bmod N\Lambda$.

$\mathscr{E}_\otimes(\cdot) = \mathbb{Z}_n^*$**-EG: ElGamal Encryption Scheme in** $\mathbb{Z}_n^*$

$\mathsf{Enc}(\mathsf{pk}, m)$: On input $m \in \mathbb{Z}_n^*$, compute $(m_1, m_2) \leftarrow (g^a, \chi^{-a} m) \in {\mathbb{J}_n}^2$ for $a \xleftarrow{\$} \mathbb{Z}_{n/2}$, so that $J_n(m) = (-1)^a$. Then, choose $r \xleftarrow{\$} \mathbb{Z}_{n/2}$ and compute $C \leftarrow \mathbb{J}_n\text{-}\mathsf{EG.Enc}(m_2; r) = (c_0 = g^r, c_1 = m_2 g_1^r)$.
Return the ciphertext $c \leftarrow \mathscr{E}_\otimes(m; r) = (C = (c_0, c_1), m_1)$.

$\mathsf{Rand}(\mathsf{pk}, c)$: Parse $c = (C = (c_0, c_1), m_1)$, choose $r_1 \xleftarrow{\$} \mathbb{Z}_{n/2}$ and $r_2 \xleftarrow{\$} \mathbb{Z}_{n/4}$, output $c' \leftarrow (C' = (g^{r_1} \cdot c_0, \chi^{-2r_2} g_1^{r_1} \cdot c_1), g^{2r_2} \cdot m_1)$.

$\mathsf{Dec}(\mathsf{sk}, c)$: Parse $c = (C = (c_0, c_1), m_1)$ and check whether $J_n(c_1) = 1$. If not, return $\perp$, otherwise compute $m_2 \leftarrow \mathbb{J}_n\text{-}\mathsf{EG.Dec}(C) = c_1 / c_0^s$ in $\mathbb{Z}_n^*$ and then $m_0 \leftarrow (1-v) \cdot m_1^{t_p} + v \cdot m_1^{t_q} \bmod n$.
Return $m \leftarrow m_0 m_2 \bmod n$.

$\mathscr{E}_\oplus(\cdot) = \mathbb{Z}_n$**-P: Paillier Encryption Scheme on** $\mathbb{Z}_n$

$\mathsf{Enc}(\mathsf{pk}, m)$: given $m \in \mathbb{Z}_n$, for a random $r \xleftarrow{\$} \mathbb{Z}_n^*$, output $c \leftarrow (1+n)^m \cdot r^n \bmod n^2$.

$\mathsf{Rand}(\mathsf{pk}, c)$: choose $r \xleftarrow{\$} \mathbb{Z}_n^*$, output $c' \leftarrow r^n \cdot c \bmod n^2$.

$\mathsf{Dec}(\mathsf{sk}, c)$: return $m \leftarrow ([c^d \bmod n^2] - 1)/n$.

Fig. 1. Setup and encryption schemes in $\mathbb{Z}_n^*$

Properties. The **correctness** follows from the Chinese Remainder Theorem: by construction, $\chi \leftarrow (1-v) \cdot g^{t_p} + v \cdot g^{t_q} \bmod n$, with v such that $v = 0 \bmod p$ and $v = 1 \bmod q$, then, $\chi = g^{t_p} \bmod p$ (so that $\chi \in \mathsf{QR}_p$) and $\chi = g^{t_q} \bmod q$ (so that $\chi \notin \mathsf{QR}_q$). Then, from $m_0 \leftarrow (1-v) m_1^{t_p} + v m_1^{t_q} \bmod n$, we also have $m_0 = g^{a t_p} = \chi^a \bmod p$ and $m_0 = g^{a t_q} = \chi^a \bmod q$, and so $m_0 = \chi^a \bmod n$. Hence, $m_0 \cdot m_2 \bmod n$ is indeed the plaintext m in $\mathbb{Z}_n^*$.

The **multiplicative homomorphism** comes from the fact that a does not need to be in $\mathbb{Z}_2$, but just has to satisfy $(-1)^a = J_n(m)$ to make both m_1 and m_2 in $\mathbb{J}_n$. If one multiplies two ciphertexts c and c', of m and m' respectively, one gets $(g^{r+r'}, \chi^{-a-a'} mm' \cdot g_1^{r+r'}, g^{a+a'}) = (g^{r''}, \chi^{-a''} mm' \cdot g_1^{r''}, g^{a''})$, which is statistically indistinguishable from a direct encryption of mm' since $\mathbb{Z}_{n/2}$ is statistically close to $\mathbb{Z}_\lambda$.

As usual, the **randomization** just consists in multiplying by a ciphertext of $m = 1$, and so with any random encoding of 1: $(m_1 = g^{2a}, m_2 = \chi^{-2a})$. Hence, on input a ciphertext $C = (C_0, C_1, \alpha)$ and two random integers (r_1, r_2), $\mathsf{Rand}(C; r_1, r_2)$ outputs $C' \leftarrow (g^{r_1} \cdot C_0, \chi^{-2r_2} \cdot g_1^{r_1} \cdot C_1, g^{2r_2} \cdot \alpha)$. Note that this algorithm returns a ciphertext in which both the random coins and the encoding of the plaintext are uniform, hence this is a perfect randomization algorithm.

Security. A ciphertext $c = (C = (c_0, c_1), m_1)$ contains m_1 in clear but m_2 is encrypted using $\mathbb{J}_n$-EG. While m_1 encodes the Jacobi symbol of the plaintext m (if m_1 is a square, $m \in \mathbb{J}_n$ and if m_1 is not a square, $m \in \mathbb{Z}_n^* \setminus \mathbb{J}_n$), under the QR assumption in $\mathbb{Z}_n^*$, it is infeasible to distinguish squares from non-squares in $\mathbb{J}_n$: m_1 does not leak anything. The choice of χ is completely independent from the $\mathbb{J}_n$-EG decryption key. This means that the IND-CPA security of the scheme just relies on the DDH assumption in $\mathbb{J}_n$ (Theorem 9) and the QR assumption in $\mathbb{Z}_n^*$.

4.4 $\mathbb{Z}_n^*$-ESP: Encryption Switching Protocols on $\mathbb{Z}_n^*$

For an ESP, the general approach consists of four steps: Alice first randomizes the ciphertext, Bob gets the decryption and then re-encrypts it under the second encryption scheme, and Alice eventually de-randomizes it. Figure 2 contains the full description of the two protocols, from $\mathbb{Z}_n$-P to $\mathbb{Z}_n^*$-EG and from $\mathbb{Z}_n^*$-EG to $\mathbb{Z}_n$-P. The former is easy because of the simple 2-party $\mathbb{Z}_n$-P decryption. The latter requires a more intricate 2-party $\mathbb{Z}_n^*$-EG decryption, that needs to interactively compute χ^a from g^a. It requires a second Paillier encryption scheme in $\mathbb{Z}_{N^2}^*$ for a larger modulus $N > (2 + 2^{\kappa+1})n^2$ to make the computations in $\mathbb{Z}$ but masking the number of loops in the reduction modulo n.

Proof of Security of $\mathbb{Z}_n^*$-ESP. About the **correctness**, C encrypts m, C'_A encrypts R_A^{-1}, and C_A encrypts $x = R_A \cdot m$, in both directions. Then $x \bullet C'_A$ is a ciphertext of m under the second encryption scheme. In the multiplicative to additive direction, this is a bit more intricate, but $A_3 = -B^{t_{pA}} B_1 + B^{t_{qA}} B_2 = -B^{t_p} + B^{t_q}$ and $A_4 = A_1 B_1 + A_2 B_2 = (1 - v_A) B^{t_p} + v_A B^{t_q}$, hence E_5 and E_6 contain encryption of $B' \times (v_B (B^{t_q} - B^{t_p}) + ((1 - v_A) B^{t_p} + v_A B^{t_q} + kn)) = B' \times ((1 - v) B^{t_p} + v B^{t_p})$. But as already remarked, $(1 - v) B^{t_p} + v B^{t_p} = \chi^{a+r_1} \bmod n$ if $\alpha = g^a$. Hence, the plaintext $m_6 = \chi^a$, and x is the expected value. (The blinding factor kn added in E_6, which masks the number of reductions modulo n, disappears in the end.)

About the **zero-knowledge**, the full and detailed proof in the honest-but-curious setting of Theorem 10 can be found in the full version [6]. But in short, the proof is done in two steps, for Alice and for Bob. For each player, we exhibit a simulator which, essentially, generates the key share of its opponent from the public key without having any information on the key of the player it emulates, and is given for each switch a target output of the protocol. The simulator forces the output of the switch to be a re-randomization of its target output. He does so by sending random ciphertexts instead of correct ciphertexts and computing some

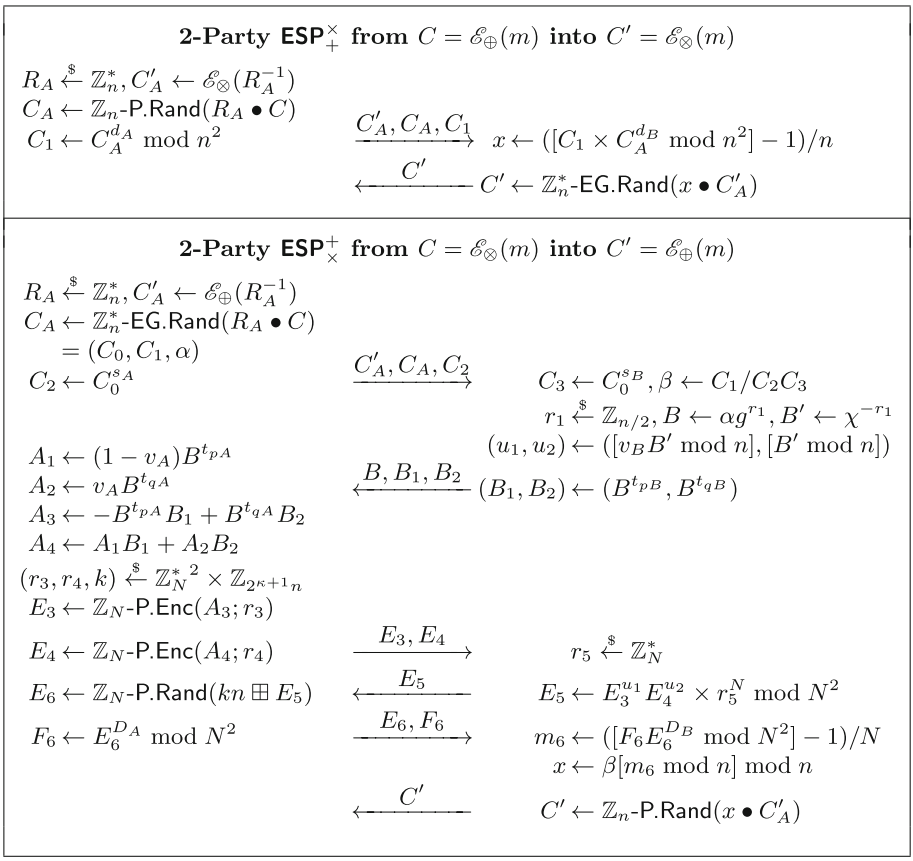

Fig. 2. Interactive protocols for encryption switching in $\mathbb{Z}_n^*$

intermediate values using either its input or its target output (both being a twin-ciphertext pair). The Paillier scheme with a second larger modulus N is necessary to hide some redundancy in the flows sent by the player that a simulator could not have sampled without the knowledge of the keys. The full proof involves several subtleties (typically, ensuring that indistinguishability between two situations involving values over $\mathbb{J}_n$ is implied by the DDH assumption over QR_n).

Theorem 10. *When instantiated with the Paillier encryption scheme and the $\mathbb{Z}_n^*$-EG encryption scheme, both over $\mathbb{Z}_n^*$, the $\mathbb{Z}_n^*$ − ESP are zero-knowledge under the DDH assumption in QR_n, the QR assumption in $\mathbb{Z}_n^*$, the DCR assumption over $\mathbb{Z}_n^*$, and the DCR assumption over $\mathbb{Z}_N^*$.*

Using our two complementary homomorphic schemes and $\mathbb{Z}_n^* - \mathsf{ESP}$ allows to evaluate functions over $\mathbb{Z}_n^*$, but no information leaks only if no intermediate computation will evaluate to 0 during the protocol. This is the goal of the next section to extend the message space of our ElGamal variant to $\mathbb{Z}_n^* \cup \{0\}$, which can be shown to be computationally equal to $\mathbb{Z}_n$.

5 An Encryption Switching Protocol over the Ring $\mathbb{Z}_n$

In order to allow computations over encrypted data in the full ring $(\mathbb{Z}_n, +, \times)$, we need to extend $\mathbb{Z}_n^*$-EG to a message space that is computationally equal to $\mathbb{Z}_n$. To this aim, we just have to handle zero. This will indeed make the two sets $\mathscr{M}_1 = \mathbb{Z}_n$ and $\mathscr{M}_2 = \mathbb{Z}_n^* \cup \{0\}$ computationally equal: finding an element in the symmetric difference provides a non-trivial non-invertible element, which breaks the factorization of n.

In the following, we use the notation $\mathscr{E}_\otimes(\cdot)$ for our above $\mathbb{Z}_n^*$-EG, and still $\mathscr{E}_\oplus(\cdot)$ for the Paillier encryption scheme $\mathbb{Z}_n$-P, both homomorphic on $(\mathbb{Z}_n^*, \times)$ and $(\mathbb{Z}_n, +)$ respectively, with the same strong RSA modulus n. We will also denote QR_n-EG and QR_n-EG$'$, two ElGamal encryption schemes over QR_n, and so with additional secret keys s_2, s_3, and $g_2 = g^{2s_2}$, $g_3 = g^{2s_3}$. QR_n-EG and QR_n-EG$'$ are clearly homomorphic in $(\mathsf{QR}_n, \times)$, and the IND-CPA security just relies on the DDH assumption in QR_n, which is independent of the factorization of n. Note however that QR_n-EG$'$ will be used as an extractable commitment and not an encryption scheme: the secret key s_3 is not kept by anybody (excepted the simulator in the security proof).

5.1 $\mathbb{Z}_n$-EG: Zero-Handling ElGamal Encryption Scheme in $\mathbb{Z}_n$

The global setup and the algorithms are represented in Fig. 3, but our $\mathbb{Z}_n$-EG encryption scheme essentially uses $\mathbb{Z}_n^*$-EG to encrypt $m+b$, where $b = 0$ if $m \neq 0$ and $b = 1$ otherwise, in $C_1 \leftarrow \mathscr{E}_\otimes(m+b)$, and is completed with two ciphertexts of b: $C_2 \leftarrow \mathsf{QR}_n\text{-EG.Enc}(T^b)$ and $C_3 \leftarrow \mathsf{QR}_n\text{-EG}'\text{.Enc}(T'^b)$, with two random squares T and T'.

The decryption algorithm is in two steps: one first decrypts C_2 to check whether the plaintext is 1, in which case $b = 0$ and so C_1 can be decrypted to get m, otherwise $b = 1$ and so one does not need to decrypt C_1 since $m = 0$. The purpose of C_3 will be for the simulation of the ESP (and namely of the *encrypted zero-test*, see below, in which the simulator is given a twin-ciphertext pair). This is reason why the decryption key s_3 will just be known to the simulator.

Properties. This scheme is correct, although the decryption is only statistically correct since the random square T can be equal to 1 with negligible probability. Since this is a combination of ElGamal encryption schemes, the resulting scheme is also IND-CPA. The 2-party decryption algorithms of $\mathbb{Z}_n^*$-EG and QR_n-EG immediately give rise to a 2-party decryption algorithm for $\mathbb{Z}_n$-EG: this is in two steps, as above, since the decryption of C_2 leads to either 1 or a random value.

Homomorphism. The multiplicativity of $\mathbb{Z}_n^*$-EG makes this scheme homomorphic until a zero is involved. And thanks to the absorbing property of random values T, it also captures the absorbing property of the zero value in the ring $\mathbb{Z}_n$: the multiplication is thus performed component-wise. In Fig. 3, we propose a randomization algorithm. One could note that C_1 will keep track of the operations performed on the ciphertexts when the global ciphertext encrypts zero, even

Setup and Key Generation

- The main strong RSA modulus n:
  - p, q two safe primes, $n \leftarrow pq$;
  - $g_0 \xleftarrow{\$} \mathbb{Z}_n^*$, $g \leftarrow -g_0^2$ (a generator of $\mathbb{J}_n$, of order λ);
  - $d \leftarrow [\lambda^{-1} \mod n] \cdot \lambda \mod n\lambda$: $d = 0 \mod \lambda$ and $d = 1 \mod n$;
  - $v \leftarrow [p^{-1} \mod q] \cdot p \mod n$: $v = 0 \mod p$ and $v = 1 \mod q$;
  - an even $t_p \xleftarrow{\$} \mathbb{Z}_\lambda$ and an odd $t_q \xleftarrow{\$} \mathbb{Z}_\lambda$: $\chi \leftarrow (1-v) \cdot g^{t_p} + v \cdot g^{t_q} \mod n$;
  - $s \xleftarrow{\$} \mathbb{Z}_\lambda$, and set $g_1 \leftarrow g^s \mod n$ (for $\mathbb{J}_n$-EG).
  - $s_2, s_3 \xleftarrow{\$} \mathbb{Z}^2_{\lambda/2}$, and set $g_2 \leftarrow g^{2s_2} \mod n$ (for QR_n-EG) and $g_3 \leftarrow g^{2s_3} \mod n$ (for QR_n-EG′).
- The additional modulus N:
  - P, Q two strong primes, $N \leftarrow PQ$ (such that $N > (2 + 2^{\kappa+1})n^2$);
  - $D \leftarrow [\Lambda^{-1} \mod N] \cdot \Lambda \mod N\Lambda$, where Λ is the order of $\mathbb{J}_N$.
- Keys: $\mathsf{pk} \leftarrow (n, g, \chi, g_1, g_2, g_3, N)$ and $\mathsf{sk} \leftarrow (d, v, t_p, t_q, s, s_2, D)$.
- Partial keys: $(d_A, v_A, t_{pA}, t_{qA}, s_A, s_{2A}, D_A) \xleftarrow{\$} \mathbb{Z}_{n\lambda} \times \mathbb{Z}_n \times \mathbb{Z}^3_\lambda \times \mathbb{Z}_{\lambda/2} \times \mathbb{Z}_{N\Lambda}$, and $d_B \leftarrow d - d_A \mod n\lambda$, $v_B \leftarrow v - v_A \mod n$, $t_{pB} \leftarrow t_p - t_{pA} \mod \lambda$, $t_{qB} \leftarrow t_q - t_{qA} \mod \lambda$, $s_B \leftarrow s - s_A \mod \lambda$, $s_{2B} \leftarrow s_2 - s_{2A} \mod \lambda/2$, and $D_B \leftarrow D - D_A \mod N\Lambda$.

$\mathscr{E}^0_\otimes(\cdot) = \mathbb{Z}_n$-**EG: ElGamal Encryption Scheme in** $\mathbb{Z}_n$

$\mathsf{Enc}(\mathsf{pk}, m)$: On input $m \in \mathbb{Z}_n$, if $m = 0$, then set $b = 1$ else set $b = 0$. Then, choose $T, T' \xleftarrow{\$} \mathsf{QR}_n$ and compute $C_1 \leftarrow \mathscr{E}_\otimes(m + b)$, $C_2 \leftarrow \mathsf{QR}_n\text{-EG.Enc}(T^b)$, $C_3 \leftarrow \mathsf{QR}_n\text{-EG'.Enc}(T'^b)$.
Return the ciphertext $C = \mathscr{E}^0_\otimes(m) = (C_1, C_2, C_3)$.

$\mathsf{Rand}(\mathsf{pk}, C = (C_1, C_2, C_3))$: Choose random $r_2, r_3 \xleftarrow{\$} \mathbb{Z}_{n/4}$, and compute $C'_1 \leftarrow \mathbb{Z}_n^*\text{-EG.Rand}(C_1)$, $C'_2 \leftarrow \mathsf{QR}_n\text{-EG.Rand}(C_2^{r_2})$, and $C'_3 \leftarrow \mathsf{QR}_n\text{-EG'.Rand}(C_3^{r_3})$. Output $C' \leftarrow (C'_1, C'_2, C'_3)$.

$\mathsf{Dec}(\mathsf{sk}, C)$: Parse $C = (C_1, C_2, C_3)$ and first decrypt $T'' \leftarrow \mathsf{QR}_n\text{-EG.Dec}(C_2)$. If $T'' = \perp$, return $\perp$; if $T'' = 1$, return 0; otherwise compute $m \leftarrow \mathscr{D}_\otimes(C_1)$ and return m.

$\mathscr{E}_\oplus(\cdot) = \mathbb{Z}_n$-**P: Paillier Encryption Scheme on** $\mathbb{Z}_n$

$\mathsf{Enc}(\mathsf{pk}, m)$: given $m \in \mathbb{Z}_n$, for a random $r \xleftarrow{\$} \mathbb{Z}_n^*$, compute $c \leftarrow (1+n)^m \cdot r^n \mod n^2$. Output $c \in \mathbb{Z}_{n^2}^*$;

$\mathsf{Rand}(\mathsf{pk}, c)$: choose $r \xleftarrow{\$} \mathbb{Z}_n^*$, output $c' \leftarrow r^n \cdot c \mod n^2$.

$\mathsf{Dec}(\mathsf{sk}, c)$: return $m \leftarrow ([c^d \mod n^2] - 1)/n$.

Fig. 3. Setup and encryption schemes in $\mathbb{Z}_n$

after randomization. We will limit the decryption of C_1 only if C_2 contains 1, and then C_1 contains the plaintext, independent of the previous steps.

Computational Equality of Message Spaces. The message space of $\mathbb{Z}_n$-EG is now $\mathbb{Z}_n^* \cup \{0\}$, which is computationally equal to $\mathbb{Z}_n$, the message space of the Paillier encryption scheme: elements in the symmetric difference are non-trivial multiples of p or q, which lead to the factorization of the modulus n.

5.2 Encrypted Zero Test

To switch between encryption schemes over $\mathbb{Z}_n$, we have to obliviously detect the zeroes during the switch; this will be done by a sub-protocol, the *encrypted zero-test* (EZT). An EZT is a protocol in which two players share a decryption key, with an encryption C of a message m as input, and wish to get an encryption C' of a bit b as output, where $b = 1$ if $m = 0$, and $b = 0$ otherwise. An EZT is *zero-knowledge* if there is an efficient simulator for each player which is indistinguishable from an honest player, and runs on input (C, C'), where C' is a twin ciphertext of C, without the knowledge of the share of the secret key of the player it emulates, but just the share of the other player.

We stress that the EZT takes as input a Paillier ciphertext C of a message m and outputs a Paillier ciphertext of b, that is 1 if $m = 0$ and 0 otherwise. However, for our ESP protocols, the simulators of the ESP are given twin-ciphertext pairs (the input C of the ESP and an expected output C'), the simulator of the EZT can also take advantage of such a pair: thanks to C_3 in C' and the trapdoor s_3, the simulator can learn the value of b.

Various protocols have been proposed for this functionality (or closely related functionalities), such as [19,30,43]. Garbled circuits for testing the equality of strings, as proposed in [25], can also be used to construct an EZT with a better communication: given a ciphertext C encrypting a plaintext m,

- Alice picks $x \xleftarrow{\$} \mathbb{Z}_n$ and sends $C_A \xleftarrow{\$} \mathsf{Rand}(C \boxplus x) = \mathsf{Enc}(mx)$ to Bob. Both players jointly decrypt C_A; Bob gets the result y. Let $x' \leftarrow x \bmod 2^\kappa$ (that Alice computes) and $y' \leftarrow y \bmod 2^\kappa$ (that Bob computes).
- Let $f(u, v)$ be the function which returns 1 if $u = v$, and 0 else. Alice picks $b_A \xleftarrow{\$} \{0, 1\}$ and builds a garbled circuit computing $b_A \mathsf{\ xor\ } f(x', y')$. Using [25], the resulting circuit has 2κ gates.
- Bob gets a bit b_B from evaluating the garbled circuit with Yao's protocol. He sends an encryption C_B of b_B to Alice.
- Alice outputs $C' \leftarrow \mathsf{Rand}(b_A \boxplus C_B \boxminus (2b_A) \bullet C_B) = \mathsf{Enc}(b_A + b_B - 2b_Ab_B) = \mathsf{Enc}(b_A \oplus b_B) = \mathsf{Enc}(f(x', y'))$.

The correctness follows from the fact that $x' = y'$ implies, with overwhelming probability, that $x - y = m = 0$, which is the plaintext of C.

Figure 4 sums up the efficiency the protocol of [25], and of the protocol of [30], which is the most efficient solution based on homomorphic encryption. Both protocols involve three rounds of on-line communication.

Authors	Preprocessing	Communication	Assumptions
[30]	2κ ciphertexts	3 ciphertexts	DCR
[25]	$8\kappa^2$ bits	κ^2 bits + κ oblivious transfers	Oblivious transfers

Fig. 4. Comparison of two EZT protocols

5.3 Encryption Switching Protocols on $\mathbb{Z}_n$

Our ESP on $\mathbb{Z}_n$ is described on Fig. 5, where the double arrows indicate an execution of an interactive sub-protocol, either a $\mathbb{Z}_n^*$-ESP or an EZT, and com is any extractable commitment (for the simulation). In the full version [6], we prove:

Theorem 11. *When instantiated with the $\mathbb{Z}_n$-P and $\mathbb{Z}_n$-EG encryption schemes, both over $\mathbb{Z}_n$, if both $\mathbb{Z}_n^*$ − ESP and EZT are zero-knowledge and if com is hiding, then the $\mathbb{Z}_n$ − ESP given in Fig. 5 is zero-knowledge under the DDH assumption in QR_n, the QR assumption in $\mathbb{Z}_n^*$ and the DCR assumption over $\mathbb{Z}_n^*$.*

Instantiating the $\mathbb{Z}_n^*$ − ESP with our construction of Sect. 4, we additionally require the DCR assumption over $\mathbb{Z}_N^*$.

6 Security Against Malicious Adversaries

In the previous sections, we built two homomorphic encryption schemes with zero-knowledge ESPs that achieve our goal of secure two-party computation from ESP: namely, at the end of the ESP executions, the *semi-honest* users do not know more than before about the input plaintexts. To prevent malicious behaviors, and move from the semi-honest setting to the malicious setting, additional validity checks are required. They are performed with zero-knowledge proofs.

Indeed, the last step toward secure ESPs is to ensure its soundness, so that malicious adversaries will not gain more information than an honest adversary would do. Moreover, the use of the simulators of the additional proofs will preserve the zero-knowledge of our ESPs. In this section, we provide this final building block.

Unfortunately, ESPs are essentially *non-arithmetic* protocols, and namely the internal decryptions and re-encryptions. Hence, ensuring honest behavior might require garbled circuits-based zero-knowledge proofs such as [15,23,35], or cut-and-choose techniques, both at a very high computational cost, but also from the communication point of view, which we cannot afford.

In this section, we present a more efficient technique for such zero-knowledge proofs, based on a particular pre-processing phase. We first explain how a *pool of random twin-ciphertext pairs* allows designing efficient (amortized) proofs of honest behavior in our ESPs.

6.1 Refreshable Twin-Ciphertext Pool

First, we set up a perfectly hiding commitment scheme $\mathsf{com}_\oplus$ over a group of order n: let k be a small integer such that $t \leftarrow 2kn + 1$ is prime. Let (g_t, h_t) be two generators of the subgroup of $\mathbb{Z}_t^*$ of order n. On input $m \in \mathbb{Z}_n$ and a randomness $r \in \mathbb{Z}_n$, the scheme outputs $\mathsf{com}_\oplus(m; r) = g_t^m h_t^r$.

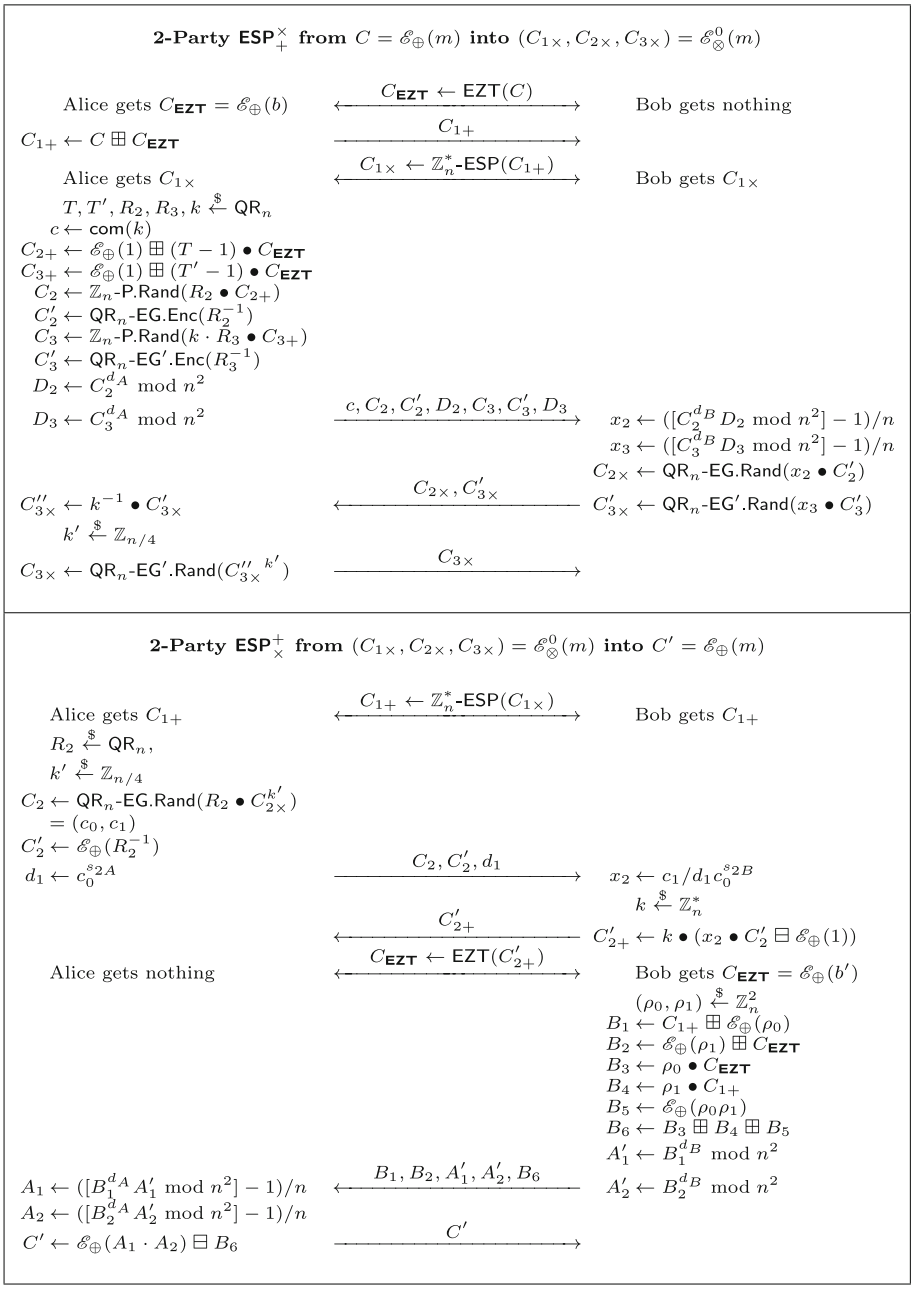

Fig. 5. Interactive Encryption Switching in $\mathbb{Z}_n$

Pre-processing Random Twin-Ciphertext Pairs. Our starting point is a protocol that allows a prover to convince a verifier that two ciphertexts, from two different cryptosystems, do indeed encrypt the same value. This means they form a *twin-ciphertext pair*. Such a proof will be denoted TCP, for *Twin-Ciphertext Proof*. It comes at a cost of the cut-and-choose technique and thus requires $O(\kappa)$ communication. However, we show in the full version [6] how to amortize ℓ TCPs using only a *single* cut-and-choose protocol, for any arbitrarily large ℓ. It relies on the techniques developed by Groth and Bayer on generalized Pedersen commitments [3,34]. But we use a new zero-knowledge proof on multi-exponentiation with committed base, of independent interest: we can create a pool of ℓ proven twin-ciphertext pairs in $O(\ell+\kappa)$. We then show several applications to speed-up various zero-knowledge arguments.

In order to generate an arbitrary number of twin-ciphertext pairs $(C_i, C'_i)_i = (\mathscr{E}_\oplus(m_i), \mathscr{E}_\otimes(m_i))_i$ of random plaintexts m_i, under two homomorphic encryption schemes, we first show how to generate a first pair: Alice has a pair of ciphertexts $(C, C') = (\mathscr{E}_\oplus(m, r), \mathscr{E}_\otimes(m', s))$ for which she knows both the plaintexts and the random coins. She wants to prove that $m = m'$ to Bob:

- Alice generates κ twin-ciphertext pairs $(C_i, C'_i)_i = (\mathscr{E}_\oplus(\mu_i; r_i), \mathscr{E}_\otimes(\mu_i; s_i))_i$, for values μ_i picked at random over $\mathbb{Z}_n$, and commits to those pairs (using any commitment scheme);
- Bob sends a challenge $c = c_1 \cdots c_\kappa \xleftarrow{\$} \{0,1\}^\kappa$;
- Alice opens the κ commitments on the twin-ciphertext pairs, and for each $i \le \kappa$, she sends
  - the plaintext μ_i and the random coins (r_i, s_i), if $c_i = 0$;
  - the ratio $R_i = m/\mu_i$ and the random coins $\rho_i \leftarrow (R_i \cdot r_i) \odot ((-1) \cdot r)$ according to the additive case, and $\sigma_i \leftarrow (R_i \cdot s_i) \odot ((-1) \cdot s)$ according to the multiplicative case — using the notations from Sect. 2.4;
- Bob checks the openings of commitments and
  - either checks the validity of (C_i, C'_i) with μ_i and the random coins;
  - or computes $D_i = R_i \bullet C_i \boxplus (-1) \bullet C$ and $D'_i = R_i \bullet C'_i \boxtimes (-1) \bullet C'$, according to the relations (1) and (2). Bod then checks whether both $D_i = \mathscr{E}_\oplus(0; \rho_i)$ and $D'_i = \mathscr{E}_\otimes(1; \sigma_i)$ hold.

We prove the security of TCP in the full version [6].

Using $\mathsf{com}_\oplus$ Instead of Paillier. The Paillier encryption scheme $\mathscr{E}_\oplus()$ in the twin-ciphertext proof can be replaced by the above perfectly hiding commitment scheme $\mathsf{com}_\oplus : (m; r) \mapsto g_t^m h_t^r$, that is also additively homomorphic. But then the proofs become arguments. Alice generates κ pairs, each pair consisting of an additive commitment and a $\mathbb{Z}_n^*$-EG ciphertext, and the rest of the proof is exactly the same. We keep using the notation $\mathscr{E}_\oplus()$ below.

Efficient Online TCP. Let us assume that we have already proven that a *random* twin-ciphertext pair $P_i = (\mathscr{E}_\oplus(m_i; r_i), \mathscr{E}_\otimes(m_i; s_i))$ is correct. When one wants to perform a TCP during a protocol on a new twin-ciphertext pair $P = (\mathscr{E}_\oplus(m; r), \mathscr{E}_\otimes(m; s))$, it is enough to reveal some relations between the random

coins of the pairs P and P_i, in order to show that the plaintexts are co-linear: if one of them is correct, so is the other. And this can be done without disclosing m (as m_i is random, disclosing m/m_i will not reveal m). Thereby, all our protocols are described in the following model: first, in a pre-processing phase, a large pool of random twin-ciphertext pairs are generated and proven correct with a batch argument. Then, in the on-line phase, each time a TCP is required, a twin-ciphertext pair from the pool is used and the player performs a cheap co-linearity proof. This proof *consumes* P_i and ensures the correctness of the switch.

Refreshing the Twin-Ciphertext Pool. The expected number of TCPs might not be known to the players; however, once a pool of twin-ciphertext pairs has been set up, the same batch technique that we describe in the full version [6] can be used to generate ℓ new random twin-ciphertext pairs, while consuming a single pair of the pool. The batch argument transmits $O(\ell + \kappa)$ group elements but does not rely on cut-and-choose, hence cut-and-choose in only needed *once*, when generating the very first element of the pool.

6.2 Zero-Knowledge Proofs

The pool of twin-ciphertext pairs allows the players to perform TCPs efficiently. Apart from TCPs, the zero-knowledge proofs needed to enhance ESPs to the malicious setting are classical protocols. For zero-knowledge proofs involving the decryption keys, we have to add the corresponding *verification keys* in the public key: first, we pick $h_0, r_0 \xleftarrow{\$} \mathbb{Z}_n^*$, $R_0 \xleftarrow{\$} \mathbb{Z}_N^*$ and set $h \leftarrow -h_0^2$, then we add $(h_p, h_q) = (h^{t_p}, h^{t_q})$ and $(u, U) \leftarrow ((1+n)\cdot r_0^{2n} \bmod n^2, (1+N)\cdot R_0^{2N} \bmod N^2) \in \mathsf{QR}_{n^2} \times \mathsf{QR}_{N^2}$ to the public key. The latter pair satisfies $u^d = 1 + n \bmod n^2$ and $U^D = 1 + N \bmod N^2$. Second, we set up the commitment scheme $\mathsf{com}_\oplus$ previously described. Each time a player performs computations, he commits to the operands if they are not already encrypted, and proves his honest behavior, with a zero-knowledge proof, using the above elements in the public key. Note that in our generic 2-PC from ESP, the switches run either sequentially or in parallel, but they are never intertwined; hence, we do not need to use zero-knowledge proofs secure in the concurrent setting (which would be less efficient).

Range Proofs. In the multiplicative to additive direction, a second Paillier encryption scheme is used, with a different modulus N. The plaintext space of this scheme is large enough to ensure that no modular reduction occurs during computations over input ciphertexts encrypting values in $\{0, \cdots, n-1\}$. Thereby, it is necessary to prove that these values are indeed in that range, which is handled by range proofs. The method of [4] provides an efficient (constant-communication) proof. Hence, we first have to commit to the encrypted value, using a generator of a space with a different modulus $n' > n$ whose factorization is unknown as in [8,16]: the plaintext is thus committed over $\mathbb{Z}_{\lambda(n')}$, a space of unknown order. Then, equality between the encrypted value (over $\mathbb{Z}_N$) and the committed value (over $\mathbb{Z}_{\lambda(n')}$, whose order is unknown) can be proven using [9],

and the range proof is performed on the committed value. The soundness of this proof relies on the *strong RSA assumption* [2,16] modulo $n' > n$. We stress that it is necessary that the factorization of n' is not known by anyone nor shared between the parties, as the strong soundness requirement of our proof of 2-PC states that the adversary is given the full secret key; hence, unlike [9], we *cannot* use $n' = N$. Note also that n' can be taken way smaller than N (which has to be large enough to allow for some multiplications without overflow).

We stress that the need of the strong RSA assumption in the security of our constant-size on-line ESP comes from the range proof, only. To date, there is no constant-size range proof over $\mathbb{Z}_n$ in the literature whose soundness does not rely on this assumption.

Classical Zero-Knowledge Proofs. Appart from TCP and range proofs, all the proofs are classical zero-knowledge proofs *á la* Schnorr [36]: Proof of re-randomization of ciphertexts and proof of correct computation of $R \bullet C$ given $\mathsf{com}_{\oplus}(R)$ are just proofs of exponentiations to the same power either in the same groups or in two groups which one order ($\mathbb{J}_n$) is unknown. It is also easy to generate $\mathscr{E}_{\otimes}(m^{-1})$ from $\mathscr{E}_{\otimes}(m)$, just inverting all the components in $\mathbb{J}_n$.

6.3 Ensuring Honest Behavior in ESP Protocols

Let us illustrate how TCP are used on the $\mathbb{Z}_n^*$-ESP from Paillier to $\mathbb{Z}_n^*$-EG (see Fig. 2): Alice sends (C_A, C'_A) and commits to R_A: $c \leftarrow \mathsf{com}_{\oplus}(R_A)$. She makes a TCP on the pair $(c, {C'_A}^{-1})$, and a classical proof of product to show that C_A encrypts the product of the value committed in c and the value encrypted in C; combined together, those proofs are enough to ensure Alice's honest behavior. Additional proofs (including range proofs) are needed in the other direction as it is a more complex case.

Sketch of the Proof of Security. In our full proof of security of the semi-honest protocols, we have already included the generation of the verification keys by the simulator. Hence, the enhanced protocol only adds zero-knowledge proofs and perfectly hiding commitments to the semi-honest proof. The zero-knowledge property of these proofs states that a simulator can fake them, *i.e.* convince a verifier of the truth of the associated statement, even if the statement is not true. Thereby, we add the two following games to our game-based proof of security in the semi-honest model, right after the very first game (in which the simulator plays honestly, using the secret keys):

1. from this game, each time the simulator is asked to perform a zero-knowledge proof, it fakes it instead. This game is indistinguishable from the honest game due to the zero-knowledge property of the proofs;
2. from this game, each time the simulator has to commit, the simulator sends a uniformly random commitment. As $\mathsf{com}_{\oplus}$ is perfectly hiding, this game is perfectly indistinguishable from the previous one.

The rest of the game-based proof is exactly the same as the semi-honest proof.

6.4 From Secure ESP to Secure 2-PC

We stress that our 2-PC protocol is made fully secure as soon as ESPs is secure against malicious adversaries (as well as the 2-party decryption procedure at the end of the protocol). This comes from the fact, that apart from ESPs and the final decryption, all the operations are *local* homomorphic evaluations of *public* functions on ciphertexts known by both players. The homomorphic operations themselves are deterministic and performed by both players.

A sequence of public operations is followed by either an ESP or, at the very end of the protocol, a 2-party decryption. From the strong soundness of the ESP, any honest player is guaranteed that his output of the ESP is necessarily a twin ciphertext of his input, or an abort is triggered. Similarly, in the 2-party decryption protocol, an honest player is guaranteed that the output is the plaintext of his input ciphertext, unless an error is raised.

Therefore, an honest player that correctly performs his (local) homomorphic evaluations is also guaranteed of the correct evaluation of the switches and of the decryption: the final answer is necessarily correct, or an error is raised in case of a misbehaving partner.

6.5 Exponential Relations Among Committed Values

We describe several applications of our preprocessing technique. In the following applications, we use a commitment scheme $\mathsf{com}(\cdot)$ we assume to be additively homomorphic. This can either be $\mathscr{E}_{\oplus}(\cdot)$ (perfectly binding) or the previous $\mathsf{com}_{\oplus}(\cdot)$ (perfectly hiding).

Proof of Knowledge of an Exponential Relation over Committed Values. A prover has sent a tuple $(C_a = \mathsf{com}(a), C_b = \mathsf{com}(b), d)$ and wishes to prove that $b = a^d$. Let C'_a and C'_b be twin ciphertexts under $\mathscr{E}_{\otimes}()$ of C_a and C_b; the prover sends them and proves them with two TCPs. Both players can compute $D \leftarrow \boxtimes^d C'_a = \mathscr{E}_{\otimes}(a^d)$. She then proves that D encrypts the same value as C'_b, which can be done since she knows all the random coins.

Extension to the Case of a Committed Exponent. Let us now suppose d has also been committed in $C_d = \mathsf{com}(d)$. The prover sends $C'_a = (C'_{a0}, C'_{a1}, \alpha)$, C'_b, and C'_d, twin ciphertexts of C_a and C_d respectively, and proves them with two TCPs. The prover computes $D = (D_0, D_1, D_2) \leftarrow \boxtimes^d C'_a = \mathscr{E}_{\otimes}(a^d)$, and proves its knowledge of (b, r) such that $C_b = \mathsf{com}(b; r)$, $D_0 = (C'_{a0})^d$, $D_1 = (C'_{a1})^d$, and $D_2 = \alpha^d$. She then proves that C'_b and D encrypt the same plaintext.

Proof of Knowledge of a Double Logarithm (or Double Decker Exponentiation). In this case, the prover wants to prove her knowledge of x that satisfies $X = g^{(h^x)}$, for public values (g, h, X). Such proofs are required for example in some publicly verifiable secret sharing schemes [38], in group signature or group encryption [24]. Let $n = pq$ be an RSA modulus such that $\pi = 2n + 1$

is a prime. Let g be a generator of a subgroup of $\mathbb{Z}_\pi^*$ of order n. Let h be a generator of $\mathbb{J}_n$ and x be an element of $\mathbb{Z}_\lambda$. The prover computes $H \leftarrow h^x$, $C \leftarrow \mathbb{Z}_n\text{-}\mathsf{P.Enc}(H)$, and $C' \leftarrow \mathbb{J}_n\text{-}\mathsf{EG.Enc}(H)$. She sends (C, C'), proves that she knows the discrete log of this encrypted value in C' (a classical Schnorr-like proof), and makes a TCP on (C, C'). She then proves her knowledge of H and r such that $X = g^H \bmod \pi$ and $C = \mathbb{Z}_n\text{-}\mathsf{P.Enc}(H; r)$ (again a Schnorr-like proof).

Proof that a Committed Value Is Prime. In [5], the authors design a zero-knowledge proof that a committed value is a product of two safe primes, which has applications in numerous RSA-based protocols. The idea is the following: to prove that a committed number π is a prime, one proves that it passed each step of the Lehmann's primality test [26,37], *i.e.* commit to κ random numbers (in an interactive way to ensure they are random) and for each of the κ random committed a, prove that $a^{(\pi-1)/2} = \pm 1 \bmod \pi$ by committing to each bit of $(\pi - 1)/2$, and by using a zero-knowledge proof for each step of the square-and-multiply algorithm. This can be done way more efficiently with our above proof of knowledge of an exponential relation over a committed exponent, enhanced using the technique from [29] to work modulo a committed value. Overall, we improve [5] by a factor of $O(\log(\pi))$. In typical applications, π will be 1024 to 2048 bit-long.

Acknowledgments. We thank Fabrice Ben Hamouda for the fruitful discussions on the ElGamal variant. This work was supported in part by the European Research Council under the European Community's Seventh Framework Programme (FP7/2007-2013 Grant Agreement no. 339563 – CryptoCloud). The second author is supported by the F.R.S-FNRS as a postdoctoral researcher.

References

1. Asharov, G., Lindell, Y., Schneider, T., Zohner, M.: More efficient oblivious transfer extensions with security for malicious adversaries. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9056, pp. 673–701. Springer, Heidelberg (2015)
2. Barić, N., Pfitzmann, B.: Collision-free accumulators and fail-stop signature schemes without trees. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 480–494. Springer, Heidelberg (1997)
3. Bayer, S., Groth, J.: Efficient zero-knowledge argument for correctness of a shuffle. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 263–280. Springer, Heidelberg (2012)
4. Boudot, F.: Efficient proofs that a committed number lies in an interval. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 431–444. Springer, Heidelberg (2000)
5. Camenisch, J.L., Michels, M.: Proving in zero-knowledge that a number is the product of two safe primes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 107–122. Springer, Heidelberg (1999)
6. Couteau, G., Peters, T., Pointcheval, D.: Encryption switching protocols. Cryptology ePrint Archive, Report 2015/990 (2015). http://eprint.iacr.org/2015/990

7. Dachman-Soled, D., Malkin, T., Raykova, M., Yung, M.: Efficient robust private set intersection. In: Abdalla, M., Pointcheval, D., Fouque, P.-A., Vergnaud, D. (eds.) ACNS 2009. LNCS, vol. 5536, pp. 125–142. Springer, Heidelberg (2009)
8. Damgård, I.B., Fujisaki, E.: A statistically-hiding integer commitment scheme based on groups with hidden order. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 125–142. Springer, Heidelberg (2002)
9. Damgård, I.B., Jurik, M.: Client/server tradeoffs for online elections. In: Naccache, D., Paillier, P. (eds.) PKC 2002. LNCS, vol. 2274, pp. 125–140. Springer, Heidelberg (2002)
10. Damgård, I.B., Nielsen, J.B.: Universally composable efficient multiparty computation from threshold homomorphic encryption. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 247–264. Springer, Heidelberg (2003)
11. Damgård, I., Pastro, V., Smart, N., Zakarias, S.: Multiparty computation from somewhat homomorphic encryption. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 643–662. Springer, Heidelberg (2012)
12. Damgård, I., Zakarias, S.: Constant-overhead secure computation of Boolean circuits using preprocessing. In: Sahai, A. (ed.) TCC 2013. LNCS, vol. 7785, pp. 621–641. Springer, Heidelberg (2013)
13. Demmler, D., Schneider, T., Zohner, M.: ABY-a framework for efficient mixed-protocol secure two-party computation. In: Network and Distributed System Security, NDSS (2015)
14. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Tran. Inf. Theory **31**, 469–472 (1985)
15. Frederiksen, T.K., Nielsen, J.B., Orlandi, C.: Privacy-free garbled circuits with applications to efficient zero-knowledge. Cryptology ePrint Archive, Report 2014/598 (2014). http://eprint.iacr.org/2014/598
16. Fujisaki, E., Okamoto, T.: Statistical zero knowledge protocols to prove modular polynomial relations. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 16–30. Springer, Heidelberg (1997)
17. Gavin, G., Minier, M.: Oblivious multi-variate polynomial evaluation. In: Roy, B., Sendrier, N. (eds.) INDOCRYPT 2009. LNCS, vol. 5922, pp. 430–442. Springer, Heidelberg (2009)
18. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Mitzenmacher, M. (ed.) 41st ACM STOC, pp. 169–178. ACM Press, May/June 2009
19. Gentry, C., Halevi, S., Jutla, C., Raykova, M.: Private database access with HE-over-ORAM architecture. Cryptology ePrint Archive, Report 2014/345 (2014). http://eprint.iacr.org/2014/345
20. Hazay, C., Mikkelsen, G.L., Rabin, T., Toft, T.: Efficient RSA key generation and threshold paillier in the two-party setting. In: Dunkelman, O. (ed.) CT-RSA 2012. LNCS, vol. 7178, pp. 313–331. Springer, Heidelberg (2012)
21. Henecka, W., Kögl, S., Sadeghi, A.R., Schneider, T., Wehrenberg, I.: TASTY: tool for automating secure two-party computations. In: Al-Shaer, E., Keromytis, A.D., Shmatikov, V. (eds.) ACM CCS 2010, pp. 451–462. ACM Press, October 2010
22. Huang, Y., Evans, D., Katz, J.: Private set intersection: are garbled circuits better than custom protocols? In: NDSS 2012. The Internet Society, February 2012
23. Jawurek, M., Kerschbaum, F., Orlandi, C.: Zero-knowledge using garbled circuits: how to prove non-algebraic statements efficiently. Cryptology ePrint Archive, Report 2013/073 (2013). http://eprint.iacr.org/2013/073

24. Kiayias, A., Tsiounis, Y., Yung, M.: Group encryption. Cryptology ePrint Archive, Report 2007/015 (2007). http://eprint.iacr.org/2007/015
25. Kolesnikov, V., Schneider, T.: Improved garbled circuit: free XOR gates and applications. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 486–498. Springer, Heidelberg (2008)
26. Kranakis, E.: Primality and Cryptography. Wiley, Hoboken (1986)
27. 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). http://eprint.iacr.org/2014/539
28. Lindell, Y., Pinkas, B.: Secure two-party computation via cut-and-choose oblivious transfer. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 329–346. Springer, Heidelberg (2011)
29. Lipmaa, H.: On diophantine complexity and statistical zero-knowledge arguments. Cryptology ePrint Archive, Report 2003/105 (2003). http://eprint.iacr.org/2003/105
30. Lipmaa, H., Toft, T.: Secure equality and greater-than tests with sublinear online complexity. In: Fomin, F.V., Freivalds, R., Kwiatkowska, M., Peleg, D. (eds.) ICALP 2013, Part II. LNCS, vol. 7966, pp. 645–656. Springer, Heidelberg (2013)
31. Naor, M., Pinkas, B.: Oblivious polynomial evaluation. SIAM J. Comput. **35**, 1254–1281 (2006)
32. Nielsen, J.B., Nordholt, P.S., Orlandi, C., Burra, S.S.: A new approach to practical active-secure two-party computation. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 681–700. Springer, Heidelberg (2012)
33. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999)
34. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (1992)
35. Ranellucci, S., Tapp, A., Zakarias, R.W.: Efficient generic zero-knowledge proofs from commitments. Cryptology ePrint Archive, Report 2014/934 (2014). http://eprint.iacr.org/2014/934
36. Schnorr, C.-P.: Efficient identification and signatures for smart cards. In: Quisquater, J.-J., Vandewalle, J. (eds.) EUROCRYPT 1989. LNCS, vol. 434, pp. 688–689. Springer, Heidelberg (1990)
37. Solovay, R., Strassen, V.: A fast monte-carlo test for primality. SIAM J. Comput. **6**(1), 84–85 (1977)
38. Stadler, M.A.: Publicly verifiable secret sharing. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 190–199. Springer, Heidelberg (1996)
39. Tassa, T., Jarrous, A., Ben-Ya'akov, Y.: Oblivious evaluation of multivariate polynomials. J. Math. Cryptol. **7**, 1–29 (2013)
40. Tople, S., Shinde, S., Chen, Z., Saxena, P.: AUTOCRYPT: enabling homomorphic computation on servers to protect sensitive web content. In: Sadeghi, A.R., Gligor, V.D., Yung, M. (eds.) ACM CCS 2013, pp. 1297–1310. ACM Press, November 2013
41. Yao, A.C.C.: How to generate and exchange secrets (extended abstract). In: 27th FOCS, pp. 162–167. IEEE Computer Society Press, October 1986

42. Ye, Q., Wang, H., Pieprzyk, J., Zhang, X.-M.: Efficient disjointness tests for private datasets. In: Mu, Y., Susilo, W., Seberry, J. (eds.) ACISP 2008. LNCS, vol. 5107, pp. 155–169. Springer, Heidelberg (2008)
43. Yu, C.-H., Yang, B.-Y.: Probabilistically correct secure arithmetic computation for modular conversion, zero test, comparison, MOD and exponentiation. In: Visconti, I., De Prisco, R. (eds.) SCN 2012. LNCS, vol. 7485, pp. 426–444. Springer, Heidelberg (2012)