# Encryption Switching Protocols

Geoffroy Couteau, Thomas Peters, David Pointcheval

ENS, Paris, France [⋆]

**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.

**Keywords.** Two-party computation, homomorphic encryption, malicious adversary, zero-knowledge proof.

## 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.

**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 [Yao86], there have been a vast amount of publications targetting 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 [NNOB12], SPDZ [DPSZ12], MiniMac [DZ13]), oblivious transfers [ALSZ15], garbled circuits [LP11], or homomorphic encryption [DN03]. In addition, there are several hybrid constructions which combine various approaches (e.g. garbled circuit and homomorphic encryption in [HKS$^+$10], secret sharing and garbled circuits in [DSZ15]). 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

[⋆] CNRS – UMR 8548 and INRIA – EPI Cascade 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, [HKS$^+$10] and [DSZ15] 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 [ElG85]. 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 [GM09], 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 [TSCS13], a trusted software is used to switch between various homomorphic schemes. In a recent unpublished paper [LTSC14], the authors propose methods to switch from the ElGamal scheme to the Paillier scheme, to evaluate DNF formulae.

As [TSCS13] relies on a trusted software, it cannot be compared to our work, which does not make this assumption. Moreover, we found both [GM09] and [LTSC14] to be flawed: in [GM09], a variant of the ElGamal encryption scheme is proposed; however, the public key of the scheme contains a square root $\beta$ 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 [LTSC14], 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 $\Pi_1$ to an encryption of the same message with a cryptosystem $\Pi_2$, 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 independant interest. We extend our variant of the ElGamal cryptosystem to a space which is "almost" 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 $\mathsf{ESP}$ 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, $\mathsf{ESP}$s 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 $\mathsf{ESP}$s 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 $\mathsf{ESP}$ 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) [GM09, LTSC14] which design methods for switching between homomorphic schemes, and [TSCS13], which relies on a trusted software to achieve a comparable goal. Fully homomorphic encryption ($\mathsf{FHE}$), 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 [Gen09]. 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 ($\mathsf{HE}$s) to perform an unlimited number of *specific* operations, and then relying on a *switching* technique to replace one $\mathsf{HE}$ 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 $\mathsf{ESP}$ 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.

**Organization.** In Section 2, we formally define *encryption switching protocols*, and show how they can lead to secure two-party computation ($\mathsf{2\text{-}PC}$) once we have two complementary homomorphic encryption schemes ($\mathsf{HE}$s). We discuss some applications of $\mathsf{ESP}$s in Section 3. Then, in Section 4, we provide a variant of ElGamal that is multiplicatively homomorphic in $\mathbb{Z}_n^*$, to be combined with the Pailler encryption scheme that is additively homomorphic in $\mathbb{Z}_n$, where $n$ is a strong RSA modulus. In Section 5, we extend this construction to handle zero. In Section 6, we describe a pre-processing method to construct efficient $\mathsf{2\text{-}PC}$ protocols secure against malicious adversaries, and in particular our $\mathsf{ESP}$s. Because of lack of space, basics on classical tools, optimizations, and detailed proofs are postponed to the Appendix.

**Preliminaries.** For the classical primitives, the reader is recommended to refer to Appendix A. But in short, a public-key encryption scheme $\Pi$ 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, $\kappa$ 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

We start with a more formal terminology.

**Definition 1 (Twin-Ciphertext Pair).** *For $i = 1, 2$, let $\Pi_i$ 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 $\Pi_1$;*
2. *$c_2$ is an encryption of $m_2 \in \mathscr{M}_2$ under $\Pi_2$;*
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 $\Pi_i$ be an encryption scheme* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}_i, \mathsf{Dec}_i)$. *An* encryption switching protocol *(ESP) between $\Pi_1$ and $\Pi_2$, 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* $\mathsf{sk}_A$ *(resp.* $\mathsf{sk}_B$*);*

$\mathsf{Switch}_{\mathsf{par}}((\mathsf{pk}, \mathsf{sk}_A, c), (\mathsf{pk}, \mathsf{sk}_B, c))$ *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 $\Pi_1$ and $\Pi_2$ 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.*

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

### 2.2 Security Notions

We expect $\mathsf{ESP}$ 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 $\mathsf{pk}$, but since it plays against Alice or Bob it can choose either $\mathsf{sk}_B$ or $\mathsf{sk}_A$, 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 $\Pi_1$ nor $\Pi_2$. 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* $\mathscr{O}_A(i{\to}j, c, \mathit{Flow})$ *that emulates the honest player A: it provides the answers A would send back upon receiving the flow* $\mathit{Flow}$ *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 '*$\mathit{Start}$*' is used to initialize the protocol.*

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 $\mathsf{Switch}$ 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 $\mathsf{sk}$ (or both $\mathsf{sk}_A$ and $\mathsf{sk}_B$), 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* $\mathit{BadSwitch}$ *is raised when a full protocol execution of* $\mathsf{Switch}$ *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* $\mathsf{Succ}^{\textit{esp-sound}}(\kappa, t)$ *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* $\mathsf{Share}$ *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*

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

*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* $\Pi_i$*, a ciphertext* $\bar{c}$ *under the encryption scheme* $\Pi_j$*, and a message flow* $\mathit{Flow}$*, ignores* $\bar{c}$ *and runs* $\mathscr{O}_B(i{\to}j, c, \mathit{Flow})$*;*

**Simulator** $\mathscr{S}im_B(i{\to}j, c, \bar{c}, \mathit{Flow})$: *on the same inputs as above, emulates the output an honest player B would answer upon receiving the flow* $\mathit{Flow}$ *when running the protocol* $\mathsf{Switch}_{i\to j}((\mathsf{pk}, \mathsf{sk}_A, c),$ $(\mathsf{pk}, \mathsf{sk}_B, c))$*, without* $\mathsf{sk}_B$ *but possibly with* $\mathsf{sk}_A$*, 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 $\mathsf{Adv}^{\textsf{esp-zk}}(\kappa, t)$ 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 $\Pi_1$ and $\Pi_2$. In that sense, switching ciphertexts is a special kind of two-party computation. This 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 IND-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 $\mathsf{Succ}^{\textsf{comp-eq}}(\mathscr{A}) = \Pr[m \leftarrow \mathscr{A}(\mathscr{M}_1, \mathscr{M}_2, \mathsf{aux}) : m \in \mathscr{M}_1 \oplus \mathscr{M}_2]$, 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 $\mathsf{R}$ 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 $b \overset{\$}{\leftarrow} \{0,1\}$; 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 2-PC protocol is *input-indistinguishable*. This is formally defined in Definition 18 in the Appendix B.

Since we assume that Alice receives the outcome of the 2-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 $\Pi_2$ admits a 2-party decryption (as detailed in Appendix A.3) so that only Alice gets the plaintexts. A rigorous construction $\Pi_{2\mathrm{PC}}$ is proposed in Appendix B, 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 $\Pi_1$ and $\Pi_2$ 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 $\Pi_2$ 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 Appendix B.

**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 [YWPZ08], 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 [HEK12] (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 $\ell$ is the size of the items in $A$ and $B$ and $M(\kappa)$ the circuit size of modular multiplication (multiplications are perfomed modulo a $\kappa$-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 [HKS+10]) 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 [DMRY09] and the next items are secure against malicious adversaries if so are the ESPs performing the switches (and Section 6 provides an efficient technique to achieve this security).

**Oblivious Multivariate Polynomial Evaluation (OMPE).** This is the natural extension of oblivious polynomial evaluation [NP06] over multivariate polynomials [TJB13]. Once an ESP is available, constructing an OMPE protocol is straightforward (we use the notations of [TJB13]). 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 [TJB13]. 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 [TJB13], 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 × (or · and even nothing), but we still use the notations of the Section 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 Appendix A.1 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 $r \in \mathsf{R}$:

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

where ≡ 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 $\mathsf{DDH}$ (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 $\mathsf{QR}$ (Quadratic Residuosity) assumption in $\mathbb{Z}_n^*$, for an RSA modulus $n$, states that a random element in $\mathsf{QR}_n$ (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 $\mathsf{DCR}$ (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 $\mathsf{DDH}$ 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 $n$, $\mathsf{DDH}$ is also a reasonable assumption in $\mathsf{QR}_n$, since the order is $p'q'$ (see Appendix A.1 for more details). Adding the $\mathsf{QR}$ assumption in $\mathbb{Z}_n^*$, this makes the $\mathsf{DDH}$ assumption in $\mathbb{J}_n$ (of order $2p'q'$) reasonable too:

**Theorem 9.** *When $n = pq$ is a strong RSA modulus, the $\mathsf{DDH}$ assumption in $\mathbb{J}_n$ is implied by the $\mathsf{DDH}$ assumption in both the large prime-order subgroups of $\mathbb{Z}_p^*$ and $\mathbb{Z}_q^*$ and the $\mathsf{QR}$ assumption in $\mathbb{Z}_n^*$. (The proof is in Appendix A.1.)*

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

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

For the Paillier encryption scheme (denoted $\mathbb{Z}_n$-$\mathsf{P}$), 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 [Pai99], Paillier an encryption scheme $\mathbb{Z}_n$-$\mathsf{P}$ for a modulus $\mathsf{pk} = n$ as public key, and $\mathsf{sk} = d \leftarrow [\lambda^{-1} \mod n] \times \lambda \mod n\lambda$ as secret key: $\mathbb{Z}_n$-$\mathsf{P.Enc}(\mathsf{pk}, m; r)$, for a message $m \in \mathbb{Z}_n$ and random coins $r$ in $\mathbb{Z}_n^*$, outputs $c = (1+n)^m \cdot r^n \mod n^2$; $\mathbb{Z}_n$-$\mathsf{P.Dec}(\mathsf{sk}, c)$ returns $m = ([c^d \mod n^2]-1)/n$. (See details in Appendix A.2). This scheme is $\mathsf{IND\text{-}CPA}$ under the $\mathsf{DCR}$ assumption over $\mathbb{Z}_{n^2}^*$, and it is additively homomorphic in $\mathbb{Z}_n$. It satisfies equation (1), $\boxplus$ being the multiplication in $\mathbb{Z}_{n^2}^*$. The randomization algorithm $\mathsf{Rand}$ is given by $\mathbb{Z}_n$-$\mathsf{P.Rand}(c; r) = c \cdot r^n \mod n^2$, 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 Appendix A.3 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 [HMRT12]). The dealer generates random $d_A, d_B \in \mathbb{Z}_{n\lambda}$ subject to $d_A + d_B = d \mod n\lambda$ 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 $C_A \leftarrow C^{d_A} \mod n^2$, which allows Bob to get the plaintext $m \leftarrow ([C_A \times C^{d_B} \mod n^2] - 1)/n$. 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 $\lambda$) 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.

**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 $\lambda$);
  - $d \leftarrow [\lambda^{-1} \mod n] \cdot \lambda \bmod n\lambda$: $d = 0 \bmod \lambda$ and $d = 1 \bmod n$;
  - $v \leftarrow [p^{-1} \bmod q] \cdot p \bmod n$: $v = 0 \bmod p$ 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 $\Lambda$ 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 $C \leftarrow \mathbb{J}_n$-$\mathsf{EG}.\mathsf{Enc}(m_2; r) = (c_0 = g^r,\ c_1 = m_2 g_1^r)$.
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 $m_2 \leftarrow \mathbb{J}_n$-$\mathsf{EG}.\mathsf{Dec}(C) = c_1/c_0^s$ 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 $\mathbb{Z}_n^*$

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

**The ElGamal Cryptosystem.** In [ElG85], 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 equation (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 $\mathsf{QR}_n$-EG and $\mathbb{J}_n$-EG, the ElGamal encryption schemes in $\mathsf{QR}_n$ 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 Figure 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 $\chi$ 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 [GM09]). 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

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

clear. The intricate point in the decryption phase will be to reconstruct $\chi^a$ 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 $\chi^a$ as $(1-v)m_1^{t_p} + vm_1^{t_q} \bmod n$. The complete description of the scheme is described on Figure 1.

**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 $m_0 \leftarrow (1-v)m_1^{t_p} + vm_1^{t_q} \bmod n$, we also have $m_0 = g^{at_p} = \chi^a \bmod p$ and $m_0 = g^{at_q} = \chi^a \bmod q$, 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 $\mathbb{J}_n$-$\mathsf{EG}$. 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 $\mathsf{QR}$ 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 $\chi$ is completely independent from the $\mathbb{J}_n$-$\mathsf{EG}$ decryption key. This means that the $\mathsf{IND\text{-}CPA}$ security of the scheme just relies on the $\mathsf{DDH}$ assumption in $\mathbb{J}_n$ (Theorem 9) and the $\mathsf{QR}$ assumption in $\mathbb{Z}_n^*$.

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

For an $\mathsf{ESP}$, the general approach consists of four steps: Alice first randomizes the ciphertext, Bob gets the decryption and then reencrypts it under the second encryption scheme, and Alice eventually de-randomizes it. Figure 2 contains the full description of the two protocols, from $\mathbb{Z}_n$-$\mathsf{P}$ to $\mathbb{Z}_n^*$-$\mathsf{EG}$ and from $\mathbb{Z}_n^*$-$\mathsf{EG}$ to $\mathbb{Z}_n$-$\mathsf{P}$. The former is easy because of the simple 2-party $\mathbb{Z}_n$-$\mathsf{P}$ decryption. The latter requires a more intricate 2-party $\mathbb{Z}_n^*$-$\mathsf{EG}$ decryption, that needs to interactively compute $\chi^a$ 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 $A_4 = A_1B_1 + A_2B_2 = (1-v_A)B^{t_p} + v_AB^{t_q}$, hence $E_5$ and $E_6$ contain encryption of $B' \times (v_B(B^{t_q} - B^{t_p}) + ((1-v_A)B^{t_p} + v_AB^{t_q} + kn)) = B' \times ((1-v)B^{t_p} + vB^{t_p})$. But as already remarked, $(1-v)B^{t_p} + vB^{t_p} = \chi^{a+r_1} \bmod n$ 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 Appendix C. 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 intermediate values using either its input or its target output (both beeing a twin-ciphertext pair). The Paillier scheme with a second larger modulus $N$ is necessary to

**2-Party $\mathsf{ESP}_{+}^{\times}$ from $C = \mathscr{E}_{\oplus}(m)$ into $C' = \mathscr{E}_{\otimes}(m)$**

$$\begin{array}{lcl}
R_A \xleftarrow{\$} \mathbb{Z}_n^*, C'_A \leftarrow \mathscr{E}_{\otimes}(R_A^{-1}) & & \\
C_A \leftarrow \mathbb{Z}_n\text{-}\mathsf{P.Rand}(R_A \bullet C) & & \\
C_1 \leftarrow C_A^{d_A} \bmod n^2 & \xrightarrow{C'_A, C_A, C_1} & x \leftarrow ([C_1 \times C_A^{d_B} \bmod n^2] - 1)/n \\
 & \xleftarrow{\quad C' \quad} & C' \leftarrow \mathbb{Z}_n^*\text{-}\mathsf{EG.Rand}(x \bullet C'_A)
\end{array}$$

**2-Party $\mathsf{ESP}_{\times}^{+}$ from $C = \mathscr{E}_{\otimes}(m)$ into $C' = \mathscr{E}_{\oplus}(m)$**

$$\begin{array}{lcl}
R_A \xleftarrow{\$} \mathbb{Z}_n^*, C'_A \leftarrow \mathscr{E}_{\oplus}(R_A^{-1}) & & \\
C_A \leftarrow \mathbb{Z}_n^*\text{-}\mathsf{EG.Rand}(R_A \bullet C) & & \\
\quad = (C_0, C_1, \alpha) & & \\
C_2 \leftarrow C_0^{s_A} & \xrightarrow{C'_A, C_A, C_2} & C_3 \leftarrow C_0^{s_B}, \beta \leftarrow C_1/C_2C_3 \\
 & & r_1 \xleftarrow{\$} \mathbb{Z}_{n/2}, B \leftarrow \alpha g^{r_1}, B' \leftarrow \chi^{-r_1} \\
 & & (u_1, u_2) \leftarrow ([v_B B' \bmod n], [B' \bmod n]) \\
A_1 \leftarrow (1 - v_A) B^{t_{pA}} & & \\
A_2 \leftarrow v_A B^{t_{qA}} & \xleftarrow{B, B_1, B_2} & (B_1, B_2) \leftarrow (B^{t_{pB}}, B^{t_{qB}}) \\
A_3 \leftarrow -B^{t_{pA}} B_1 + B^{t_{qA}} B_2 & & \\
A_4 \leftarrow A_1 B_1 + A_2 B_2 & & \\
(r_3, r_4, k) \xleftarrow{\$} {\mathbb{Z}_N^*}^2 \times \mathbb{Z}_{2^{\kappa+1} n} & & \\
E_3 \leftarrow \mathbb{Z}_N\text{-}\mathsf{P.Enc}(A_3; r_3) & & \\
E_4 \leftarrow \mathbb{Z}_N\text{-}\mathsf{P.Enc}(A_4; r_4) & \xrightarrow{\quad E_3, E_4 \quad} & r_5 \xleftarrow{\$} \mathbb{Z}_N^* \\
E_6 \leftarrow \mathbb{Z}_N\text{-}\mathsf{P.Rand}(kn \boxplus E_5) & \xleftarrow{\quad E_5 \quad} & E_5 \leftarrow E_3^{u_1} E_4^{u_2} \times r_5^N \bmod N^2 \\
F_6 \leftarrow E_6^{D_A} \bmod N^2 & \xrightarrow{\quad E_6, F_6 \quad} & m_6 \leftarrow ([F_6 E_6^{D_B} \bmod N^2] - 1)/N \\
 & & x \leftarrow \beta[m_6 \bmod n] \bmod n \\
 & \xleftarrow{\quad C' \quad} & C' \leftarrow \mathbb{Z}_n\text{-}\mathsf{P.Rand}(x \bullet C'_A)
\end{array}$$

Fig. 2: Interactive Protocols for Encryption Switching in $\mathbb{Z}_n^*$

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 $\mathsf{QR}_n$).

**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 $\mathsf{QR}_n$, 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 $\mathbb{Z}_n^*$−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 $\mathsf{QR}_n$-EG and $\mathsf{QR}_n$-EG′, two ElGamal encryption schemes over $\mathsf{QR}_n$, and so with additional secret keys $s_2$, $s_3$, and $g_2 = g^{2s_2}$, $g_3 = g^{2s_3}$. $\mathsf{QR}_n$-EG and $\mathsf{QR}_n$-EG′ are clearly homomorphic in $(\mathsf{QR}_n, \times)$, and the IND-CPA security just relies on the DDH assumption in $\mathsf{QR}_n$, which is independent of the factorization of $n$. Note however that $\mathsf{QR}_n$-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).

**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 $\lambda$);
  - $d \leftarrow [\lambda^{-1} \mod n] \cdot \lambda \bmod n\lambda$: $d = 0 \bmod \lambda$ and $d = 1 \bmod n$;
  - $v \leftarrow [p^{-1} \bmod q] \cdot p \bmod n$: $v = 0 \bmod p$ 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 $\mathbb{J}_n$-$\mathsf{EG}$).
  - $s_2, s_3 \xleftarrow{\$} \mathbb{Z}_{\lambda/2}^2$, and set $g_2 \leftarrow g^{2s_2} \bmod n$ (for $\mathsf{QR}_n$-$\mathsf{EG}$) and $g_3 \leftarrow g^{2s_3} \bmod n$ (for $\mathsf{QR}_n$-$\mathsf{EG}'$).
- 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 $\Lambda$ 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: $(d_A, v_A, t_{pA}, t_{qA}, s_A, s_{2A}, D_A) \xleftarrow{\$} \mathbb{Z}_{n\lambda} \times \mathbb{Z}_n \times \mathbb{Z}_\lambda^3 \times \mathbb{Z}_{\lambda/2} \times \mathbb{Z}_{N\Lambda}$, and $d_B \leftarrow d - d_A \bmod n\lambda$, $v_B \leftarrow v - v_A \bmod n$, $t_{pB} \leftarrow t_p - t_{pA} \bmod \lambda$, $t_{qB} \leftarrow t_q - t_{qA} \bmod \lambda$, $s_B \leftarrow s - s_A \bmod \lambda$, $s_{2B} \leftarrow s_2 - s_{2A} \bmod \lambda/2$, and $D_B \leftarrow D - D_A \bmod N\Lambda$.

**$\mathscr{E}_\otimes^0(\cdot) = \mathbb{Z}_n$-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 $C_1 \leftarrow \mathscr{E}_\otimes(m + b)$, $C_2 \leftarrow \mathsf{QR}_n$-$\mathsf{EG.Enc}(T^b)$, $C_3 \leftarrow \mathsf{QR}_n$-$\mathsf{EG}'.\mathsf{Enc}(T'^b)$.
Return the ciphertext $C = \mathscr{E}_\otimes^0(m) = (C_1, C_2, C_3)$.

$\mathsf{Rand}(\mathsf{pk}, C = (C_1, C_2, C_3))$ : Choose random $r_2, r_3 \xleftarrow{\$} \mathbb{Z}_{n/4}$, and compute $C_1' \leftarrow \mathbb{Z}_n^*$-$\mathsf{EG.Rand}(C_1)$, $C_2' \leftarrow \mathsf{QR}_n$-$\mathsf{EG.Rand}(C_2^{r_2})$, and $C_3' \leftarrow \mathsf{QR}_n$-$\mathsf{EG}'.\mathsf{Rand}(C_3^{r_3})$. 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 $T'' \leftarrow \mathsf{QR}_n$-$\mathsf{EG.Dec}(C_2)$. 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 $c \leftarrow (1+n)^m \cdot r^n \bmod n^2$. Output $c \in \mathbb{Z}_{n^2}^*$;

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

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

Fig. 3: Setup and Encryption Schemes in $\mathbb{Z}_n$

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

The global setup and the algorithms are represented in Figure 3, but our $\mathbb{Z}_n$-$\mathsf{EG}$ encryption scheme essentially uses $\mathbb{Z}_n^*$-$\mathsf{EG}$ 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 $b$: $C_2 \leftarrow \mathsf{QR}_n$-$\mathsf{EG.Enc}(T^b)$ and $C_3 \leftarrow \mathsf{QR}_n$-$\mathsf{EG}'.\mathsf{Enc}(T'^b)$, 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 $\mathsf{ESP}$ (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 $\mathsf{IND}$-$\mathsf{CPA}$. The 2-party decryption algorithms of $\mathbb{Z}_n^*$-$\mathsf{EG}$ and $\mathsf{QR}_n$-$\mathsf{EG}$ immediately give rise to a 2-party decryption algorithm for $\mathbb{Z}_n$-$\mathsf{EG}$: 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 $\mathbb{Z}_n^*$-$\mathsf{EG}$ 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 Figure 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 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 $\mathbb{Z}_n$-$\mathsf{EG}$ 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* ($\mathsf{EZT}$). An $\mathsf{EZT}$ 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 $\mathsf{EZT}$ 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 $\mathsf{EZT}$ 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 $\mathsf{ESP}$ protocols, the simulators of the $\mathsf{ESP}$ are given twin-ciphertext pairs (the input $C$ of the $\mathsf{ESP}$ and an expected output $C'$), the simulator of the $\mathsf{EZT}$ 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 [YY12, LT13, GHJR14]. Garbled circuits for testing the equality of strings, as proposed in [KS08], can also be used to construct an $\mathsf{EZT}$ with a better communication: given a ciphertext $C$ encrypting a plaintext $m$,

- Alice picks $x \overset{\$}{\leftarrow} \mathbb{Z}_n$ and sends $C_A \overset{\$}{\leftarrow} \mathsf{Rand}(C \boxplus x) = \mathsf{Enc}(mx)$ 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 $b_A \overset{\$}{\leftarrow} \{0, 1\}$ and builds a garbled circuit computing $b_A$ $\mathsf{xor}$ $f(x', y')$. Using [KS08], the resulting circuit has $2\kappa$ 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 $C' \leftarrow \mathsf{Rand}(b_A \boxplus C_B \boxminus (2b_A) \bullet C_B) = \mathsf{Enc}(b_A + b_B - 2b_A b_B) = \mathsf{Enc}(b_A \oplus b_B) = \mathsf{Enc}(f(x', y'))$.

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 [KS08], and of the protocol of [LT13], which is the most efficient solution based on homomorphic encryption. Both protocols involve three rounds of on-line communication.

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

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

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

| Authors | Preprocessing | Communication | Assumptions |
|---|---|---|---|
| [LT13] | $2\kappa$ ciphertexts | 3 ciphertexts | DCR |
| [KS08] | $8\kappa^2$ bits | $\kappa^2$ bits + $\kappa$ oblivious transfers | Oblivious transfers |

Fig. 4: Comparison of two $\mathsf{EZT}$ protocols

**2-Party $\mathsf{ESP}_{+}^{\times}$ from $C = \mathscr{E}_{\oplus}(m)$ into $(C_{1\times}, C_{2\times}, C_{3\times}) = \mathscr{E}_{\otimes}^{0}(m)$**

| | | |
|---|---|---|
| Alice gets $C_{\mathsf{EZT}} = \mathscr{E}_{\oplus}(b)$ | $\xleftrightarrow{C_{\mathsf{EZT}} \leftarrow \mathsf{EZT}(C)}$ | Bob gets nothing |
| $C_{1+} \leftarrow C \boxplus C_{\mathsf{EZT}}$ | $\xrightarrow{C_{1+}}$ | |
| Alice gets $C_{1\times}$ | $\xleftrightarrow{C_{1\times} \leftarrow \mathbb{Z}_n^*\text{-}\mathsf{ESP}(C_{1+})}$ | Bob gets $C_{1\times}$ |
| $T, T', R_2, R_3, k \xleftarrow{\$} \mathsf{QR}_n$ | | |
| $c \leftarrow \mathsf{com}(k)$ | | |
| $C_{2+} \leftarrow \mathscr{E}_{\oplus}(1) \boxplus (T-1) \bullet C_{\mathsf{EZT}}$ | | |
| $C_{3+} \leftarrow \mathscr{E}_{\oplus}(1) \boxplus (T'-1) \bullet C_{\mathsf{EZT}}$ | | |
| $C_2 \leftarrow \mathbb{Z}_n\text{-}\mathsf{P.Rand}(R_2 \bullet C_{2+})$ | | |
| $C_2' \leftarrow \mathsf{QR}_n\text{-}\mathsf{EG.Enc}(R_2^{-1})$ | | |
| $C_3 \leftarrow \mathbb{Z}_n\text{-}\mathsf{P.Rand}(k \cdot R_3 \bullet C_{3+})$ | | |
| $C_3' \leftarrow \mathsf{QR}_n\text{-}\mathsf{EG'.Enc}(R_3^{-1})$ | | |
| $D_2 \leftarrow C_2'^{d_A} \bmod n^2$ | | |
| $D_3 \leftarrow C_3'^{d_A} \bmod n^2$ | $\xrightarrow{c, C_2, C_2', D_2, C_3, C_3', D_3}$ | $x_2 \leftarrow ([C_2'^{d_B} D_2 \bmod n^2] - 1)/n$ |
| | | $x_3 \leftarrow ([C_3'^{d_B} D_3 \bmod n^2] - 1)/n$ |
| | | $C_{2\times} \leftarrow \mathsf{QR}_n\text{-}\mathsf{EG.Rand}(x_2 \bullet C_2')$ |
| $C_{3\times}'' \leftarrow k^{-1} \bullet C_{3\times}'$ | $\xleftarrow{C_{2\times}, C_{3\times}'}$ | $C_{3\times}' \leftarrow \mathsf{QR}_n\text{-}\mathsf{EG'.Rand}(x_3 \bullet C_3')$ |
| $k' \xleftarrow{\$} \mathbb{Z}_{n/4}$ | | |
| $C_{3\times} \leftarrow \mathsf{QR}_n\text{-}\mathsf{EG'.Rand}({C_{3\times}''}^{k'})$ | $\xrightarrow{C_{3\times}}$ | |

**2-Party $\mathsf{ESP}_{\times}^{+}$ from $(C_{1\times}, C_{2\times}, C_{3\times}) = \mathscr{E}_{\otimes}^{0}(m)$ into $C' = \mathscr{E}_{\oplus}(m)$**

| | | |
|---|---|---|
| Alice gets $C_{1+}$ | $\xleftrightarrow{C_{1+} \leftarrow \mathbb{Z}_n^*\text{-}\mathsf{ESP}(C_{1\times})}$ | Bob gets $C_{1+}$ |
| $R_2 \xleftarrow{\$} \mathsf{QR}_n,$ | | |
| $k' \xleftarrow{\$} \mathbb{Z}_{n/4}$ | | |
| $C_2 \leftarrow \mathsf{QR}_n\text{-}\mathsf{EG.Rand}(R_2 \bullet C_{2\times}^{k'})$ | | |
| $= (c_0, c_1)$ | | |
| $C_2' \leftarrow \mathscr{E}_{\oplus}(R_2^{-1})$ | | |
| $d_1 \leftarrow c_0^{s_{2A}}$ | $\xrightarrow{C_2, C_2', d_1}$ | $x_2 \leftarrow c_1/d_1 c_0^{s_{2B}}$ |
| | | $k \xleftarrow{\$} \mathbb{Z}_n^*$ |
| | $\xleftarrow{C_{2+}'}$ | $C_{2+}' \leftarrow k \bullet (x_2 \bullet C_2' \boxminus \mathscr{E}_{\oplus}(1))$ |
| Alice gets nothing | $\xleftrightarrow{C_{\mathsf{EZT}} \leftarrow \mathsf{EZT}(C_{2+}')}$ | Bob gets $C_{\mathsf{EZT}} = \mathscr{E}_{\oplus}(b')$ |
| | | $(\rho_0, \rho_1) \xleftarrow{\$} \mathbb{Z}_n^2$ |
| | | $B_1 \leftarrow C_{1+} \boxplus \mathscr{E}_{\oplus}(\rho_0)$ |
| | | $B_2 \leftarrow \mathscr{E}_{\oplus}(\rho_1) \boxplus C_{\mathsf{EZT}}$ |
| | | $B_3 \leftarrow \rho_0 \bullet C_{\mathsf{EZT}}$ |
| | | $B_4 \leftarrow \rho_1 \bullet C_{1+}$ |
| | | $B_5 \leftarrow \mathscr{E}_{\oplus}(\rho_0 \rho_1)$ |
| | | $B_6 \leftarrow B_3 \boxplus B_4 \boxplus B_5$ |
| | | $A_1' \leftarrow B_1^{d_B} \bmod n^2$ |
| $A_1 \leftarrow ([B_1^{d_A} A_1' \bmod n^2] - 1)/n$ | $\xleftarrow{B_1, B_2, A_1', A_2', B_6}$ | $A_2' \leftarrow B_2^{d_B} \bmod n^2$ |
| $A_2 \leftarrow ([B_2^{d_A} A_2' \bmod n^2] - 1)/n$ | | |
| $C' \leftarrow \mathscr{E}_{\oplus}(A_1 \cdot A_2) \boxminus B_6$ | $\xrightarrow{C'}$ | |

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

Instantiating the $\mathbb{Z}_n^*$−ESP with our construction of Section 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 [JKO13, FNO14, RTZ14], 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 $\mathsf{com}_\oplus(m; r) = g_t^m h_t^r$.

**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 Appendix D how to amortize $\ell$ TCP*s* using only a *single* cut-and-choose protocol, for any arbitrarily large $\ell$. It relies on the techniques developed by Groth and Bayer on generalized Pedersen commitments [Ped92, BG12]. But we use a new zero-knowledge proof on multi-exponentiation with committed base, of independent interest: we ca create a pool of $\ell$ 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 $\kappa$ 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 $\mu_i$ picked at random over $\mathbb{Z}_n$, and commits to those pairs (using any commitment scheme);
– Bob sends a challenge $c = c_1 \cdots c_\kappa \overset{\$}{\leftarrow} \{0,1\}^\kappa$;
– Alice opens the $\kappa$ commitments on the twin-ciphertext pairs, and for each $i \leq \kappa$, she sends
  - the plaintext $\mu_i$ 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 Section 2.4;
– Bob checks the openings of commitments and
  - either checks the validity of $(C_i, C'_i)$ with $\mu_i$ 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 Appendix D.

*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 $\kappa$ pairs, each pair consisting of an additive commitment and a $\mathbb{Z}_n^*$-$\mathsf{EG}$ 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 TCP*s* 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 can be use to generate $\ell$ 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 TCP*s* efficiently. Apart from TCP*s*, 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 $h_0, r_0 \overset{\$}{\leftarrow} \mathbb{Z}_n^*$, $R_0 \overset{\$}{\leftarrow} \mathbb{Z}_N^*$ 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 [Bou00] 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 [FO97, DF02]: 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 [DJ02], and the range proof is performed on the committed value. The soundness of this proof relies on the *strong RSA assumption* [BP97, FO97] 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 [DJ02], 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 $\mathsf{ESP}$ 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 $\mathsf{TCP}$ and range proofs, all the proofs are classical zero-knowledge proofs *à la* Schnorr [Sch90]: 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 $\mathsf{TCP}$ are used on the $\mathbb{Z}_n^*$-$\mathsf{ESP}$ from Paillier to $\mathbb{Z}_n^*$-$\mathsf{EG}$ (see Figure 2): Alice sends $(C_A, C'_A)$ and commits to $R_A$: $c \leftarrow \mathsf{com}_{\oplus}(R_A)$. She makes a $\mathsf{TCP}$ 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 pre-processing 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 $\mathsf{TCP}s$. 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 $\mathsf{TCP}s$. 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 [Sta96], in group signature or group encryption [KTY07]. 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 $\mathsf{TCP}$ 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 [CM99], 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 $\pi$ is a prime, one proves that it passed each step of the Lehmann's primality test [SS77, Kra86], *i.e.* commit to $\kappa$ random numbers (in an interactive way to ensure they are random) and for each of the $\kappa$ 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 [Lip03] to work modulo a committed value. Overall, we improve [CM99] by a factor of $O(\log(\pi))$. In typical applications, $\pi$ 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).

## References

ALSZ15. G. Asharov, Y. Lindell, T. Schneider, and M. Zohner. More efficient oblivious transfer extensions with security for malicious adversaries. In *EUROCRYPT 2015, Part I*, *LNCS* 9056, pages 673–701. Springer, Heidelberg, April 2015.

BG12. S. Bayer and J. Groth. Efficient zero-knowledge argument for correctness of a shuffle. In *EUROCRYPT 2012*, *LNCS* 7237, pages 263–280. Springer, Heidelberg, April 2012.

Bou00. F. Boudot. Efficient proofs that a committed number lies in an interval. In *EUROCRYPT 2000*, *LNCS* 1807, pages 431–444. Springer, Heidelberg, May 2000.

BP97. N. Bari and B. Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In *EUROCRYPT'97*, *LNCS* 1233, pages 480–494. Springer, Heidelberg, May 1997.

CM99. J. Camenisch and M. Michels. Proving in zero-knowledge that a number is the product of two safe primes. In *EUROCRYPT'99*, *LNCS* 1592, pages 107–122. Springer, Heidelberg, May 1999.

DF02. I. Damgård and E. Fujisaki. A statistically-hiding integer commitment scheme based on groups with hidden order. In *ASIACRYPT 2002*, *LNCS* 2501, pages 125–142. Springer, Heidelberg, December 2002.

DJ02. I. Damgård and M. Jurik. Client/server tradeoffs for online elections. In *PKC 2002*, *LNCS* 2274, pages 125–140. Springer, Heidelberg, February 2002.

DMRY09. D. Dachman-Soled, T. Malkin, M. Raykova, and M. Yung. Efficient robust private set intersection. In *ACNS 09*, *LNCS* 5536, pages 125–142. Springer, Heidelberg, June 2009.

DN03. I. Damgård and J. B. Nielsen. Universally composable efficient multiparty computation from threshold homomorphic encryption. In *CRYPTO 2003*, *LNCS* 2729, pages 247–264. Springer, Heidelberg, August 2003.

DPSZ12. I. Damgård, V. Pastro, N. P. Smart, and S. Zakarias. Multiparty computation from somewhat homomorphic encryption. In *CRYPTO 2012*, *LNCS* 7417, pages 643–662. Springer, Heidelberg, August 2012.

DSZ15. D. Demmler, T. Schneider, and M. Zohner. Aby–a framework for efficient mixed-protocol secure two-party computation. In *Network and Distributed System Security, NDSS*, 2015.

DZ13. I. Damgård and S. Zakarias. Constant-overhead secure computation of boolean circuits using preprocessing. In *TCC 2013*, *LNCS* 7785, pages 621–641. Springer, Heidelberg, March 2013.

ElG85. T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31:469–472, 1985.

FNO14. T. K. Frederiksen, J. B. Nielsen, and C. Orlandi. Privacy-free garbled circuits with applications to efficient zero-knowledge. Cryptology ePrint Archive, Report 2014/598, 2014. `http://eprint.iacr.org/2014/598`.

FO97. E. Fujisaki and T. Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In *CRYPTO'97*, *LNCS* 1294, pages 16–30. Springer, Heidelberg, August 1997.

Gen09. C. Gentry. Fully homomorphic encryption using ideal lattices. In *41st ACM STOC*, pages 169–178. ACM Press, May / June 2009.

GHJR14. C. Gentry, S. Halevi, C. Jutla, and M. Raykova. Private database access with HE-over-ORAM architecture. Cryptology ePrint Archive, Report 2014/345, 2014. `http://eprint.iacr.org/2014/345`.

GM09. G. Gavin and M. Minier. Oblivious multi-variate polynomial evaluation. In *INDOCRYPT 2009*, *LNCS* 5922, pages 430–442. Springer, Heidelberg, December 2009.

HEK12. Y. Huang, D. Evans, and J. Katz. Private set intersection: Are garbled circuits better than custom protocols? In *NDSS 2012*. The Internet Society, February 2012.

HKS$^+$10. W. Henecka, S. Kögl, A.-R. Sadeghi, T. Schneider, and I. Wehrenberg. TASTY: tool for automating secure two-party computations. In *ACM CCS 10*, pages 451–462. ACM Press, October 2010.

HMRT12. C. Hazay, G. L. Mikkelsen, T. Rabin, and T. Toft. Efficient RSA key generation and threshold Paillier in the two-party setting. In *CT-RSA 2012*, *LNCS* 7178, pages 313–331. Springer, Heidelberg, February / March 2012.

JKO13. M. Jawurek, F. Kerschbaum, and C. Orlandi. 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`.

Kra86. E. Kranakis. *Primality and cryptography.* John Wiley & Sons, Inc., 1986.

KS08. V. Kolesnikov and T. Schneider. Improved garbled circuit: Free XOR gates and applications. In *ICALP 2008, Part II*, *LNCS* 5126, pages 486–498. Springer, Heidelberg, July 2008.

KTY07. A. Kiayias, Y. Tsiounis, and M. Yung. Group encryption. Cryptology ePrint Archive, Report 2007/015, 2007. `http://eprint.iacr.org/2007/015`.

Lip03. H. Lipmaa. On diophantine complexity and statistical zero-knowledge arguments. Cryptology ePrint Archive, Report 2003/105, 2003. `http://eprint.iacr.org/2003/105`.

LP11. Y. Lindell and B. Pinkas. Secure two-party computation via cut-and-choose oblivious transfer. In *TCC 2011*, *LNCS* 6597, pages 329–346. Springer, Heidelberg, March 2011.

LT13. H. Lipmaa and T. Toft. Secure equality and greater-than tests with sublinear online complexity. In *ICALP 2013, Part II*, *LNCS* 7966, pages 645–656. Springer, Heidelberg, July 2013.

LTSC14. H. W. Lim, S. Tople, P. Saxena, and E.-C. Chang. Faster secure arithmetic computation using switchable homomorphic encryption. Cryptology ePrint Archive, Report 2014/539, 2014. `http://eprint.iacr.org/2014/539`.

NNOB12. J. B. Nielsen, P. S. Nordholt, C. Orlandi, and S. S. Burra. A new approach to practical active-secure two-party computation. In *CRYPTO 2012*, *LNCS* 7417, pages 681–700. Springer, Heidelberg, August 2012.

NP06. M. Naor and B. Pinkas. Oblivious polynomial evaluation. *SIAM J. Comput*, 35:1254–1281, 2006.

Pai99. P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT'99*, *LNCS* 1592, pages 223–238. Springer, Heidelberg, May 1999.

Ped92. T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *CRYPTO'91*, *LNCS* 576, pages 129–140. Springer, Heidelberg, August 1992.

RTZ14. S. Ranellucci, A. Tapp, and R. W. Zakarias. Efficient generic zero-knowledge proofs from commitments. Cryptology ePrint Archive, Report 2014/934, 2014. `http://eprint.iacr.org/2014/934`.

Sch90. C.-P. Schnorr. Efficient identification and signatures for smart cards (abstract) (rump session). In *EUROCRYPT'89*, *LNCS* 434, pages 688–689. Springer, Heidelberg, April 1990.

SS77. R. Solovay and V. Strassen. A fast monte-carlo test for primality. *SIAM J. Comput.*, 6(1):84–85, 1977.

Sta96. M. Stadler. Publicly verifiable secret sharing. In *EUROCRYPT'96*, *LNCS* 1070, pages 190–199. Springer, Heidelberg, May 1996.

TJB13. T. Tassa, A. Jarrous, and Y. Ben-Ya'akov. Oblivious evaluation of multivariate polynomials. *Journal of Mathematical Cryptology*, 7:1–29, 2013.

TSCS13. S. Tople, S. Shinde, Z. Chen, and P. Saxena. AUTOCRYPT: enabling homomorphic computation on servers to protect sensitive web content. In *ACM CCS 13*, pages 1297–1310. ACM Press, November 2013.

Yao86. A. C.-C. Yao. How to generate and exchange secrets (extended abstract). In *27th FOCS*, pages 162–167. IEEE Computer Society Press, October 1986.

YWPZ08. Q. Ye, H. Wang, J. Pieprzyk, and X.-M. Zhang. Efficient disjointness tests for private datasets. In *ACISP 08*, *LNCS* 5107, pages 155–169. Springer, Heidelberg, July 2008.

YY12. C.-H. Yu and B.-Y. Yang. Probabilistically correct secure arithmetic computation for modular conversion, zero test, comparison, MOD and exponentiation. In *SCN 12*, *LNCS* 7485, pages 426–444. Springer, Heidelberg, September 2012.

# A Preliminaries

An algorithm is *efficient* when it runs in polynomial time in the (implicit) security parameter $\kappa$. A positive function $f$ is *negligible* if for any polynomial $p \in \text{poly}$ there exists a bound $B > 0$ such that, for any integers $k \geq B$, $f(k) \leq 1/|p(k)|$. It holds that a function $f$ is negligible if and only if there exists a negligible function $\varepsilon$ verifying $f(\kappa) \leq \varepsilon(\kappa)$ beyond some bound. An event depending on $\kappa$ occurs with *overwhelming probability* when its probability is at least $1 - \varepsilon(\kappa)$ for a negligible function $\varepsilon$.

Given a finite set $S$, the notation $x \xleftarrow{\$} S$ means a uniformly random affectation of an element of $S$ to the variable $x$, then for any $s \in S$ we have $\Pr_S[x = s] = \Pr[x = s \,|\, x \xleftarrow{\$} S] = \Pr[x \xleftarrow{\$} S \,:\, x = s] = 1/|S|$ where $|S|$ denotes the cardinality of $S$. When an element $s$ is represented by an integer, $|s|$ is the bit-length of the integer. The integer range $[a..b]$ stands for $\{x \in \mathbb{Z} \,|\, a \leq x \leq b\}$. For any integers $a = a(\kappa)$ and $b = b(\kappa)$ such that $a \leq b$, the statistical distance between two uniform distributions over $U_a = [1..a]$ and $U_b = [1..b]$, is given by $\sum_{i=1}^{b} |\Pr_{U_a}[x = i] - \Pr_{U_b}[x = i]| = \sum_{i=1}^{a} 1/a - 1/b + \sum_{i=a+1}^{b} 1/b \leq 2(b-a)/b$. Hence, we will say that $a = a(\kappa)$ and $b = b(\kappa)$ are *statistically closed* if $(b - a)/b$ is negligible.

## A.1 Some Facts on the Ring $\mathbb{Z}_n$

The notation $(\mathbb{Z}_n, +, \times)$ refers to the ring of integers modulo $n$. More precisely, $\mathbb{Z}_n$ is the quotient set $\mathbb{Z}/\equiv_n$ for the relation induced by $a \equiv_n b$ if $a = b \bmod n$. Operations in $\mathbb{Z}$ are carried into $\mathbb{Z}_n$ by reduction modulo $n$ and these are well-defined. In this paper we only focus on $\mathbb{Z}_n$ for a strong RSA modulus $n = pq$ where $p, q$ are distinct safe primes. That means that $p = 2p' + 1$ and $q = 2q' + 1$ for two other primes so that $p, p', q, q'$ are all distinct. One can note that in such a case, $p$ and $q$ are Blum primes: $p = q = 3 \bmod 4$. Some facts arise from this specific choice of $n$. The Charmichael number $\lambda(n) = \text{lcm}(\varphi(p), \varphi(q)) = \varphi(n)/2 = 2p'q'$, where $\varphi$ is the Euler totient, gives the order of the maximal cyclic subgroup of $(\mathbb{Z}_n^*, \times)$ (consequently, for any $a \in \mathbb{Z}_n^*$, its order is among the set $\{1, 2, p', q', 2p', 2q', p'q', 2p'q'\}$). Such a multiplicative subgroup of maximal order is given by $\mathbb{J}_n = \{a \in \mathbb{Z}_n \,|\, J_n(a) = 1\}$, the set of invertible elements $a$ whose Jacobi symbol $J_n(a) = (a|n) = (a|p) \cdot (a|q)$ is equal to 1. Furthermore, if we let $\mathsf{QR}_n = \{a \in \mathbb{Z}_n^* \,|\, \exists b \in \mathbb{Z}_n^*, a = b^2\} = \{a \in \mathbb{Z}_n^* \,|\, (a|p) = 1 = (a|q)\}$ be set of the quadratic residues modulo $n$, $\mathsf{QR}_n$ is a cyclic subgroup of $\mathbb{J}_n$ of order $\varphi(n)/4 = p'q'$ and we have $-1 \in \mathbb{J}_n \setminus \mathsf{QR}_n$ —since both $p$ and $q$ are Blum primes—. Finally, any $a \in \mathsf{QR}_n$ has four square roots with exactly one in $\mathsf{QR}_n$, one in $\mathbb{J}_n \setminus \mathsf{QR}_n$ and two in $\mathbb{Z}_n^* \setminus \mathbb{J}_n$. We also remind that two square roots with distinct Jacobi symbol values (one in $\mathbb{J}_n$ and one in $\mathbb{Z}_n^* \setminus \mathbb{J}_n$) allows efficiently factoring $n$.

By $\mathsf{GenMod}(\kappa)$, we denote a probabilistic efficient algorithm that, given the security parameter $\kappa$, generates public parameters $(n, g)$ and secret parameters $(p, q)$ of at least $\kappa$ bits each with the specification that $n = pq$ is a strong RSA modulus and that $g$ is a random generator of $\mathbb{J}_n$. The generator $g$ can be defined by setting $g = -g_0^2$ for an efficiently samplable $g_0 \xleftarrow{\$} \mathbb{Z}_n^*$ since the order of $g_0$ is $p'q'$ or $2p'q'$ with overwhelming probability. In the following, we write $((n, g), (p, q)) \leftarrow \mathsf{GenMod}(\kappa)$.

**Theorem 9.** *The DDH assumption in $\mathbb{J}_n$ is implied by the DDH assumption in large prime order subgroups of $\mathbb{Z}_n^*$ and the* QR *assumption in $\mathbb{Z}_n^*$.*

With the first hybrid sequence of games, on the left in Figure 6, we essentially show that the DDH assumption in $\mathbb{J}_n$ is implied by the DDH assumption in $\mathsf{QR}_n$ and the QR assumption. Each of these distributions generates a generator of either $\mathsf{QR}_n$ or $\mathbb{J}_n$, two scalars taken uniformly in $[1..2p'q']$ or $[1..n/2]$ and finally either set $c = ab$ in the integers or take $c$ at random in the same range as $a$ and $b$. Most of the gaps being statistically indistinguishable ($\approx_{\mathsf{stat.}}$).

In the second hybrid sequence of games, on the right in Figure 6, if one defines $\mathbb{G}_1$ as the $q'$-powers in $\mathsf{QR}_n$ and $\mathbb{G}_2$ as the $p'$-powers in $\mathsf{QR}_n$, while the factorization of $n$ is known, this shows that the DDH assumption in $\mathsf{QR}_n$ is implied by the DDH assumptions in both $\mathbb{G}_1$ and $\mathbb{G}_2$. One can note that,

| | $g$ | $a, b$ | $c = ab$ |
|---|---|---|---|
| $\approx_{\mathsf{stat.}}$ | $\mathbb{J}_n$ | $2p'q'$ | ✓ |
| $\mathsf{QR}$ | $\mathbb{J}_n$ | $n/2$ | ✓ |
| $\approx_{\mathsf{stat.}}$ | $\mathsf{QR}_n$ | $n/2$ | ✓ |
| $\mathsf{DDH}$ in $\mathsf{QR}_n$ | $\mathsf{QR}_n$ | $2p'q'$ | ✓ |
| $\approx_{\mathsf{stat.}}$ | $\mathsf{QR}_n$ | $2p'q'$ | ✗ |
| $\mathsf{QR}$ | $\mathsf{QR}_n$ | $n/2$ | ✗ |
| $\approx_{\mathsf{stat.}}$ | $\mathbb{J}_n$ | $n/2$ | ✗ |
| | $\mathbb{J}_n$ | $2p'q'$ | ✗ |

| | $a, b$ | $c = ab$ | | |
|---|---|---|---|---|
| | | mod $p'$ | mod $q'$ | |
| $\mathscr{D}_0$ | $2p'q'$ | ✓ | ✓ | $\approx_{\mathsf{stat.}}$ |
| $\mathscr{D}_1$ | $p'q'$ | ✓ | ✓ | $\approx_{\mathsf{stat.}}$ |
| $\mathscr{D}_2$ | $p'$ | ✓ | ✓ | $\mathsf{DDH}$ in $\mathbb{G}_1$ |
| $\mathscr{D}_3$ | $p'$ | ✗ | ✓ | $\approx_{\mathsf{stat.}}$ |
| $\mathscr{D}_4$ | $p'q'$ | ✗ | ✓ | $\approx_{\mathsf{stat.}}$ |
| $\mathscr{D}_5$ | $q'$ | ✗ | ✓ | $\mathsf{DDH}$ in $\mathbb{G}_2$ |
| $\mathscr{D}_6$ | $q'$ | ✗ | ✗ | $\approx_{\mathsf{stat.}}$ |
| $\mathscr{D}_7$ | $p'q'$ | ✗ | ✗ | $\approx_{\mathsf{stat.}}$ |
| $\mathscr{D}_8$ | $2p'q'$ | ✗ | ✗ | |

Fig. 6: Hybrid Sequence of Games

*Proof.* given two generators $g_1$ and $g_2$, of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively, $g \leftarrow g_1 g_2$ is a generator of $\mathsf{QR}_n$. Then, a tuple $(A \leftarrow g_1^a, B \leftarrow g_1^b, C \leftarrow g_1^c)$ in $\mathbb{G}_1$, where either $c = ab \bmod p'$ or not, can be converted into $(A' \leftarrow Ag_2^\alpha, B' \leftarrow Bg_2^\beta, C' \leftarrow Cg_2^{\alpha\beta})$, which is either a Diffie-Hellman tuple in $\mathsf{QR}_n$ or not. Which explains the gap between $\mathscr{D}_2$ and $\mathscr{D}_3$. A similar analysis leads to the gap between $\mathscr{D}_5$ and $\mathscr{D}_6$. □

### A.2 Encryption Scheme

A common setup may be used to generate different key pairs. Then, we split the $\mathsf{Setup}$ from the $\mathsf{KeyGen}$ algorithms.

**Definition 12 (Encryption Scheme).** *An encryption scheme $\Pi$ consists of four algorithms* ($\mathsf{Setup}$, $\mathsf{KeyGen}$, $\mathsf{Enc}$, $\mathsf{Dec}$)*:*

- $\mathsf{Setup}(\kappa)$ *generates the public parameters* $\mathsf{pp}$*;*
- $\mathsf{KeyGen}(\mathsf{pp})$ *outputs a public key* $\mathsf{pk}$ *and a secret key* $\mathsf{sk}$*,* $\mathsf{pk}$ *specifies the message space $\mathscr{M}$, the ciphertext space $\mathscr{C}$, and the random source $\mathscr{R}$;*
- $\mathsf{Enc}(\mathsf{pk}, m; r)$*, given the message $m$, outputs a ciphertext $c$, under the encryption key* $\mathsf{pk}$ *with the randomness $r$. When there is no ambiguity, we use the notation* $\mathsf{Enc}(m)$*, where the coins $r$ are chosen uniformly at random;*
- $\mathsf{Dec}(\mathsf{sk}, c)$*, outputs a plaintext $m$, encrypted in the ciphertext $c$ using the decryption key* $\mathsf{sk}$*, or $\perp$ in case of failure. Similarly as above, when there is no ambiguity on the decryption key, we use the notation* $\mathsf{Dec}(c)$*.*

Encryption schemes are assumed to satisfy the following properties:

*Correctness.* An encryption scheme $\Pi$ is *correct* if for any pair of keys $(\mathsf{pk}, \mathsf{sk})$ generated by $\mathsf{KeyGen}$ from $\mathsf{pp} \leftarrow \mathsf{Setup}(\kappa)$ and any message $m \in \mathscr{M}$, we always have $\mathsf{Dec}(\mathsf{Enc}(m)) = m$.

*Semantic Security (IND-CPA).* The classical security notion for encryption is the indistinguishability of ciphertexts: no adversary can distinguish the encryptions of the plaintexts $m_0$ and $m_1$ of its choice, given just access to the public parameters.

*Homomorphic Encryptions.* An encryption scheme $\Pi$ is homomorphic if, for an internal law $\oplus$ in $\mathscr{M}$, it further provides an algorithm $\mathsf{Eval}$ such that, for any $m_1, m_2 \in \mathscr{M}$, $\mathsf{Dec}(\mathsf{sk}, \mathsf{Eval}(\mathsf{pk}, \mathsf{Enc}(\mathsf{pk}, m_1), \mathsf{Enc}(\mathsf{pk}, m_2))) = m_1 \oplus m_2$, and the two distributions $\mathscr{D}_0 = \{\mathsf{Eval}(\mathsf{pk}, \mathsf{Enc}(\mathsf{pk}, m_1), \mathsf{Enc}(\mathsf{pk}, m_2))\}$ and $\mathscr{D}_1 = \{\mathsf{Enc}(\mathsf{pk}, m_1 \oplus m_2)\}$ are computationally/statistically/perfectly indistinguishable over $\mathscr{C}$.

**Examples.** The classical examples of homomorphic encryption schemes are the famous ElGamal encryption scheme [ElG85] which is multiplicatively homomorphic in any subgroup where the DDH assumption holds, and the Paillier encryption scheme [Pai99], which is additively homomorphic in $\mathbb{Z}_n$, where $n$ is an RSA modulus.

*The Paillier Cryptosystem.* The Paillier's encryption scheme [Pai99] is usually defined as follows:

- $\mathsf{KeyGen}(\kappa)$: run $((n,g),(p,q)) \overset{\$}{\leftarrow} \mathsf{GenMod}(\kappa)$ and compute $\delta \leftarrow n^{-1} \bmod \lambda$ ($n$ and $\lambda = \lambda(n)$ are relatively prime). Return $\mathsf{pk} = n$ and $\mathsf{sk} = \delta$;
- $\mathsf{Enc}(\mathsf{pk}, m; r)$: given $m \in \mathbb{Z}_n$, for a random $r \overset{\$}{\leftarrow} \mathbb{Z}_n^*$, compute and output $c \leftarrow (1+n)^m \cdot r^n \bmod n^2$;
- $\mathsf{Dec}(\mathsf{sk}, c)$: compute $r \leftarrow c^{\mathsf{sk}} \bmod n$ and then $c_0 \leftarrow [c \cdot r^{-n} \bmod n^2]$. Return $m \leftarrow (c_0 - 1)/n$.

Alternatively, $\mathsf{KeyGen}'(\kappa)$ outputs $\mathsf{pk} = n$ and $\mathsf{sk} = d$ where $d \leftarrow [\lambda^{-1} \bmod n]\cdot\lambda \bmod n\lambda$, and $\mathsf{Dec}'(\mathsf{sk}, c)$ outputs $m = ([c^d \bmod n^2] - 1)/n$, which is the Paillier variant we will use in this paper. This scheme is $\mathsf{IND\text{-}CPA}$ under the $\mathsf{DCR}$ assumption, and it is additively homomorphic in $\mathbb{Z}_n$. All along this paper, this encryption scheme is called the Paillier encryption scheme, and the encryption algorithm is usually denoted $\mathbb{Z}_n$-$\mathsf{P}$, or $\mathscr{E}_{\oplus}(\cdot)$, when the public key $n$ can be omitted.

*The ElGamal Cryptosystem.* Let $\mathbb{G}$ be a group of order $M$ and $T$ be a public integer statistically close to $M$. Given $T$, the ElGamal's encryption scheme [ElG85] over $\mathbb{G}$ is defined as follows:

$\mathsf{KeyGen}(\kappa)$: choose a random $x \overset{\$}{\leftarrow} \mathbb{Z}_M$, set and return $\mathsf{sk} = x$ and $\mathsf{pk} = y = g^x$;
$\mathsf{Enc}(\mathsf{pk}, m; r)$: given $m \in \mathbb{G}$, for a random $r \overset{\$}{\leftarrow} \mathbb{Z}_T$, compute and output $c \leftarrow (c_1 \leftarrow g^r, c_2 \leftarrow y^r \cdot m)$;
$\mathsf{Dec}(\mathsf{sk}, c)$: compute and output $m = c_2/c_1^x$.

This scheme is $\mathsf{IND\text{-}CPA}$ under the $\mathsf{DDH}$ assumption in $\mathbb{G}$, and it is (multiplicatively) homomorphic in $\mathbb{G}$. All along this paper, for a strong RSA modulus $n = pq$, this encryption scheme in $\mathbb{G} = \mathsf{QR}_n$ is called the $\mathsf{QR}_n$-$\mathsf{EG}$ encryption scheme and this encryption scheme in $\mathbb{G} = \mathbb{J}_n$ is called the $\mathbb{J}_n$-$\mathsf{EG}$ encryption scheme. Over $\mathsf{QR}_n$, writing $\lambda \leftarrow \lambda(n)$, one can set $T = M = \lambda/2$, in which case the ElGamal is performed over a group of known order (and in particular, the factorization of $n$ is known). Alternatively, if one wants to keep secret the factorization of the modulus (as we do in this paper), we set $T = (n-1)/4 = M + (p+q-2)/4$ (which is statistically close to $M$). Over $\mathbb{J}_n$, with $M = \lambda$, as the security of the $\mathsf{DDH}$ assumption relies on the quadratic residuosity assumption (hence the factorization must be kept hidden), we set $T = (n-1)/2$. Our goal is to extend the ElGamal scheme to $\mathbb{Z}_n^*$ and even $\mathbb{Z}_n$ (or most of it).

### A.3 2-Party Encryption Scheme

Two-party encryption scheme is a special case of $(L, T)$-threshold encryption scheme: $L$ users are able to decrypt ciphertexts as long as at least $T$ of them collaborate; nobody gets information on the plaintexts if the threshold $T$ is not reached. Most of the threshold encryption schemes in the literature are designed from an ordinary encryption scheme, with a *secret sharing* of the decryption key. Here we explicitly follows this constructive approach, with $L = T = 2$, since we define the key generation as a secret sharing of a pre-existing secret key, and the decryption algorithm is replaced by a 2-party interactive decryption protocol.

**Definition 13 (2-party Encryption Scheme).** *A 2-party encryption scheme $\Pi$ consists of four algorithms* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$*:*

- $\mathsf{Setup}(\kappa)$ *generates the public parameters* $\mathsf{pp}$*;*
- $\mathsf{KeyGen}(\mathsf{pp})$ *gets* $(\mathsf{pk}, \mathsf{sk})$ *from the key generation of the ordinary encryption scheme and outputs* $(\mathsf{pk}, \mathsf{sk}_A, \mathsf{sk}_B) \leftarrow \mathsf{Share}(\mathsf{pk}, \mathsf{sk})$ *where* $\mathsf{pk}$ *is the updated public key* $\mathsf{pk}$ *and* $(\mathsf{sk}_A, \mathsf{sk}_B)$ *are the two shares of the secret key* $\mathsf{sk}$*. The party $A$ (resp. $B$) is intended to be given* $\mathsf{sk}_A$ *(resp.* $\mathsf{sk}_B$*). The message space $\mathscr{M}$, the ciphertext space $\mathscr{C}$ and the random source $\mathscr{R}$ are specified by* $\mathsf{pk}$*;*
- $\mathsf{Enc}(\mathsf{pk}, m; r)$*, given the message $m$, outputs a ciphertext $c$, under the encryption key* $\mathsf{pk}$ *with the randomness $r$ as done by the encryption algorithm of the ordinary scheme (but we might write $c \leftarrow$* $\mathsf{Enc}(\mathsf{pk}, m)$ *or even $c \leftarrow$* $\mathsf{Enc}(m)$*, when there is no ambiguity);*
- $\mathsf{Dec}((\mathsf{pk}, \mathsf{sk}_A, c), (\mathsf{pk}, \mathsf{sk}_B, c))$ *is a 2-party interactive protocol that outputs either a plaintext $m =$* $\mathsf{Dec}(\mathsf{sk}, c)$ *with an implicit use of the decryption key* $\mathsf{sk}$ *or $\perp$ in case of failure.*

*At the end of the 2-party decryption protocol, only one party or both could get the plaintext.*

*Correctness.* A 2-party encryption scheme $\mathit{\Pi}$ is *correct* if for any tuple of keys $(\mathsf{pk}, \mathsf{sk}_A, \mathsf{sk}_B)$ generated by $\mathsf{KeyGen}(\mathsf{pp})$ from $\mathsf{pp} \leftarrow \mathsf{Setup}(\kappa)$ and any message $m \in \mathscr{M}$, with $c \leftarrow \mathsf{Enc}(m)$ we always have $\mathsf{Dec}((\mathsf{pk}, \mathsf{sk}_A, c), (\mathsf{pk}, \mathsf{sk}_B, c)) = m$.

The following security notions consist of the usual guarantees we can expect for a 2-party encryption scheme. Only informal descriptions of these notions are given below since we will rely on sufficient conditions (*i.e.* stronger properties) which are formally defined afterward.

*Verifiable Decryption.* In the malicious setting, an adversary can play the role of at most one of the two parties and try to deviate from the specification of $\mathsf{Dec}$. If indeed the adversary is trying to cheat we want to detect it with overwhelming probability.

*Semantic Security (IND-CPA).* The semantic security of a 2-party encryption scheme corresponds to the usual IND-CPA security notion of ordinary encryption schemes except that the adversary may have access to one of the both secret key shares. Moreover, the adversary also has access to an additional decryption-share oracle (modeling the other honest party which supposedly holds the other secret-key share). The oracle indistinguishably emulates the flow of the execution of the interactive protocol $\mathsf{Dec}$ (depending on the adversarial setting) but with the restriction that it must only be invoked on ciphertexts of *known* plaintexts. A formal definition follows from [?].

**Example: Semi-Honest 2-Party Paillier Scheme.** Let $\mathsf{pk} = n$ and $\mathsf{sk} = d$ be the output of the key generation algorithm of the variant of the Paillier encryption $(\mathsf{Setup}, \mathsf{KeyGen}', \mathsf{Enc}, \mathsf{Dec}')$ described above in Appendix A.2. The setup algorithm and the encryption algorithm of the 2-party variant in the semi-honest case are $\mathsf{Setup}$ and $\mathsf{Enc}$. The other algorithms are given by

- $\mathsf{KeyGen}(\mathsf{pp})$: From $n = pq$ and $d = [\lambda^{-1} \bmod n] \cdot \lambda \bmod n\lambda$, a trusted dealer runs $\mathsf{Share}$ to generate a secret sharing of $\mathsf{sk} = d$. It picks $d_A \overset{\$}{\leftarrow} \mathbb{Z}_{n\lambda}$ and sets $\mathsf{sk}_A \leftarrow d_A$, $\mathsf{sk}_B \leftarrow d - d_A \bmod n\lambda$. It outputs $(n, \mathsf{sk}_A, \mathsf{sk}_B)$;
- $\mathsf{Dec}((n, \mathsf{sk}_A, c), (n, \mathsf{sk}_B, c))$: To decrypt a ciphertext $c \in \mathbb{Z}^*_{n^2}$, Alice sends $M_A = c^{\mathsf{sk}_A} \bmod n^2$ to Bob, and Bob computes $M_B = c^{\mathsf{sk}_B} \bmod n^2$ and then the value $M = [M_A M_B \bmod n^2]$ equal to $[c^d \bmod n^2]$. Finally, Bob recovers $m \leftarrow (M-1)/n$.

The ElGamal encryption scheme also allows 2-party decryption with an additive secret sharing of $x$, with $\mathsf{CDH}$ proofs (in the group $\mathbb{G}$, that is with an order that is either known or unknown) for valid partial decryption.

*Sufficient Security Notions.* The usual way to show the IND-CPA security of the threshold encryption scheme is to show that the interactive decryption protocol does not degrade the IND-CPA security of the ordinary scheme, when the plaintext is already known [?]. Then, in the 2-party case and as in the switching protocols defined in Section 2, the adversary playing one of the both parties is allowed to query an oracle playing the other party during the interactive run of the protocols, here in the case of decryption. Let us formally define the partial decryption oracle.

**Definition 14 ($\mathscr{O}_A^{\mathsf{Dec}}$ and $\mathscr{O}_B^{\mathsf{Dec}}$ Oracles).** *For appropriate keys $(\mathsf{pk}, \mathsf{sk}_A, \mathsf{sk}_B)$, we denote the stateful oracle $\mathscr{O}_A^{\mathsf{Dec}}(c, \mathit{Flow})$ that emulates the honest player A: it provides the answers A would send back upon receiving the flow Flow when running the protocol $\mathsf{Dec}((\mathsf{pk}, \mathsf{sk}_A, c), (\mathsf{pk}, \mathsf{sk}_B, c))$. We similarly define the oracle $\mathscr{O}_B^{\mathsf{Dec}}$ that emulates the honest player B. A special flow 'Start' is used to initialize to the protocol.*

The soundness property guarantees that no malicious player can make the decryption protocol output a plaintext that is not encrypted in the input ciphertext. In some extent, it states that the correctness is preserved in front of any single dishonest party. Here we make a stronger requirement with respect to some usual notions of soundness since we allow the adversary to get the whole secret key $\mathsf{sk}$ instead of only one secret shares. This strengthening is motivated by our generic 2-party computation.

**Definition 15 (Strong Soundness).** *A 2-party encryption scheme satisfies the **strong soundness**, 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* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}$*, any secret key shares* $(\mathsf{sk}_A, \mathsf{sk}_B) \leftarrow \mathsf{Share}(\mathsf{pk}, \mathsf{sk})$*, for all PPT adversary* $\mathscr{A}$ *playing the role of* $A$*, the success*

$$\mathsf{Succ}_B^{\textsf{\textit{dec-sound}}}(\mathscr{A}) = \Pr[\, \textsf{\textit{BadDec}} \,|\, \mathscr{A}^{\mathscr{O}_B^{\mathsf{Dec}}(\cdot,\cdots)}(\mathsf{pk}, \mathsf{sk})]$$

*is negligible, where the event* $\textsf{\textit{BadDec}}$ *is raised when a full protocol execution of* $\mathsf{Dec}$ *with* $\mathscr{O}_B^{Dec}$ *on input ciphertext* $c$*, for which it exists* $m$ *such that* $c = \mathsf{Enc}(\mathsf{pk}, m)$*, successfully outputs* $m^\star \neq m$*. We denote* $\mathsf{Succ}^{\textsf{\textit{dec-sound}}}(\kappa, t)$ *the maximal success an adversary can get against* $A$ *or* $B$ *in time* $t$*.*

The zero-knowledge property guarantees that no information leaks about the decryption keys to a malicious player: its view can be simulated without any additional information than its own secret key share.

**Definition 16 (Zero-Knowledge).** *A 2-party encryption scheme is said* ***zero-knowledge****, if it is zero-knowledge for* $A$ *and zero-knowledge for* $B$*. The scheme is zero-knowledge for* $B$ *if there exists an efficient simulator* $\mathscr{S}im_B^{\mathsf{Dec}}$ *of the oracle* $\mathscr{O}_B^{\mathsf{Dec}}$ *with the following property: for any* $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\kappa)$*, any keys* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}$*, any secret key shares* $(\mathsf{pk}, \mathsf{sk}_A, \mathsf{sk}_B) \leftarrow \mathsf{Share}(\mathsf{pk}, \mathsf{sk})$*, and for any PPT adversary* $\mathscr{A}$ *playing the role of* $A$*, the advantage* $\mathsf{Adv}_B^{\textsf{\textit{dec-zk}}}(\mathscr{A})$ *defined as*

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

*is negligible, where the adversary* $\mathscr{A}$ *is given unbounded access to either the simulator* $\mathscr{S}im_B^{\mathsf{Dec}}$ *or the stateful oracle* $\tilde{\mathscr{O}}_B^{\mathsf{Dec}}$ *described below, and when the input tuples* $(c, m, \textsf{\textit{Flow}})$ *are consistent* i.e. *if* $c$ *is an encryption of* $m$*:*

**Oracle** $\tilde{\mathscr{O}}_B^{\mathsf{Dec}}(c, m, \textsf{\textit{Flow}})$*: on input a ciphertext* $c$*, a plaintext* $m$*, and a message flow* $\textsf{\textit{Flow}}$*, ignores* $m$ *and runs* $\mathscr{O}_B^{\mathsf{Dec}}(c, \textsf{\textit{Flow}})$*;*

**Simulator** $\mathscr{S}im_B^{\mathsf{Dec}}(c, m, \textsf{\textit{Flow}})$*: on the same inputs as above, emulates the output an honest player* $B$ *would answer upon receiving the flow* $\textsf{\textit{Flow}}$ *when running the protocol* $\mathsf{Dec}((\mathsf{pk}, \mathsf{sk}_A, c), (\mathsf{pk}, \mathsf{sk}_B, c))$*, without* $\mathsf{sk}_B$ *but possibly with* $\mathsf{sk}_A$*, and forcing the output to be* $m$*.*

*When the adversary* $\mathscr{A}$ *is unbounded, the scheme is statistically zero-knowledge. We denote* $\mathsf{Succ}^{\textsf{\textit{dec-zk}}}(\kappa, t)$ *the maximal advantage an adversary can get against* $A$ *or* $B$ *in time* $t$*.*

It is worth noticing that the Zero-Knowledge property here only guarantees that a curious party cannot get *more* information on the other secret share that it could not derived from its *share* and its output of the 2-party decryption. For instance a 2-party encryption with a secret share $(\mathsf{sk}_A, \mathsf{sk}_B) = (\mathsf{sk}, \perp)$ of $\mathsf{sk}$ could be zero-knowledge but not $\mathsf{IND\text{-}CPA}$. The last property we need is the simulatability of the key shares: from the public key only, a simulator can indistinguishably output each secret share separately.

**2-Party Paillier Scheme Secure Against Malicious Adversaries.** We enhance the security of our previously described 2-party Paillier encryption scheme so that it can withstand malicious adversaries:

- For $((n, g), (p, q)) \leftarrow \mathsf{GenMod}(\kappa)$, where $n = pq$ and $p = 2p' + 1, q = 2q' + 1 > 2^\kappa$, a trusted dealer generates $\lambda = 2p'q'$, $d \leftarrow [\lambda^{-1} \bmod n] \cdot \lambda \bmod n\lambda$, $z \overset{\$}{\leftarrow} \mathsf{QR}_{n^2}$ and $d_A \overset{\$}{\leftarrow} \mathbb{Z}_{n\lambda}$. He sets $\mathsf{sk}_A \leftarrow d_A$, $\mathsf{sk}_B \leftarrow d - d_A \bmod n\lambda$ as well as $\mathsf{vk}_A \leftarrow z^{\mathsf{sk}_A}, \mathsf{vk}_B \leftarrow z^{\mathsf{sk}_B}$. The public verification keys consist of $(z, \mathsf{vk}_A, \mathsf{vk}_B)$.
- To decrypt a ciphertext $c \in \mathbb{Z}_{n^2}^*$,
  - Alice picks $r \overset{\$}{\leftarrow} \mathbb{Z}_{2^\kappa n^2}$ and sends $H = z^r, C = c^r$ and $M_A = c^{\mathsf{sk}_A} \bmod n^2$.
  - Bob sends a random $\kappa'$-bit string $x$ as challenge, where $\kappa' + 4 = \kappa$.
  - Alice sends $D = \mathsf{sk}_A x + r$; Bob checks that $z^D = Hv_A^x$ and $c^{2D} = C^2 M_A^{2x}$.
  - If the verifications succeed, Bob computes $M_B = c^{2\mathsf{sk}_B} \bmod n^2$, which gives him $M = M_A^2 M_B = c^{2d} \bmod n^2$. Then, Bob recovers $m_2 \leftarrow (M - 1)/n$ and sets $m \leftarrow m_2/2 \bmod n$.

*Strong Soundness.* First, let us note that the proof involved in this protocol is a $\mathsf{CDH}$ proof over $(z, c^2, \mathsf{vk}_A, M_A^2)$, and $z$ is a square. Working over the squares allows us to ensure the existence of a unique witness so that this tuple is a Diffie-Hellman tuple: let us suppose that Alice was able to answer two distinct challenges $x, x'$; then Bob can check that $z^{D-D'} = v_A^{x-x'}$ and $M_A^{2(x-x')} = c^{2(D-D')}$. By working over the squares, we ensure that $(x - x') < p', q'$ is invertible over $\mathbb{Z}_{\lambda/2}$, hence there is indeed a unique witness $w = (D - D')(x - x')^{-1} \bmod n\lambda/2$ so that $z^w = v_A$ and $M_A^{2w} = c^2$.

We now argue that working over $\mathsf{QR}_{n^2}$ is sufficient to get a secure 2-party decryption even in face of adversary who knows the factorization of $n$. Indeed, even if some malicious adversaries could have sent some $M'_A$ such that $M'_A \neq M_A$ but ${M'_A}^2 = M_A^2$, he cannot force an incorrect decryption by doing so: the proof ensures that computing $M = M_A^2 M_B$ and then $m_2 \leftarrow (M - 1)/n$ is indeed a valid decryption of $c^2$, which, by the homomorphism of the Paillier scheme, is an encryption of $2m$: $m_2 = 2m$, hence Bob recovers $m$ by computing $m \leftarrow m_2/2 \bmod n$: the protocol is *statistically sound* for $A$ (hence, it is also strongly sound for $A$); the soundness is independent of $\mathsf{sk} = (p, q)$.

As Bob does not interact at all with Alice in the decryption procedure (he sends a challenge which is part of the proof, but not of the decryption itself), he cannot force the decryption of an incorrect value, hence the protocol is trivially (statistically) sound for $B$.

*Zero-Knowledge.* For the simulation of Alice, the verification key can be generated at random: $u$ as above, $u_A \leftarrow u^{\mathsf{sk}_A} \bmod n^2$, for $\mathsf{sk}_A \xleftarrow{\$} \mathbb{Z}_{n^2/2}$, and $u_B \leftarrow (n + 1)/u_A \bmod n^2$. During the decryption, $\mathscr{S}im$ sends $M' \cdot c^{-\mathsf{sk}_B}$ to force the decryption of $m' = (1 + nM')$. Together with the zero-knowledge property of the $\mathsf{CDH}$ proof, this shows that the protocol is zero-knowledge for Alice. Note that the $\mathsf{CDH}$ proof we described here is only honest-verifier zero-knowledge, and can be made zero-knowledge against malicious adversaries via standard techniques (such as committing to the challenge before the first flow with an extractable commitment scheme).

### A.4 Commitment Scheme

Informally, a commitment scheme is a tool that prevents the owner of the value to change the value he should be using in a protocol, without revealing the value: in a first phase, he *commits* to this value with a scheme that ensures that the value remains hidden, and in a second phase, he *opens* the commitment to the original value, in a way that anyone can check it. A classical way involves two non-interactive algorithms: $c \leftarrow \mathsf{com}(m; r)$ is the commitment algorithm, that generates the commitment $c$ on the value $m$ with some randomness $r$. In order to open $c$, one just reveals $m$ and $r$, so that anybody can check whether $c = \mathsf{com}(m; r)$. Such a commitment scheme should then satisfy the following properties:

- Perfectly (resp. statistically, computationally) hiding: given $(m, m') \in \mathscr{S}^2$, the two distributions $\{\mathsf{com}(m; r)\}_{r \in \mathscr{R}}$ and $\{\mathsf{com}(m'; r)\}_{r \in \mathscr{R}}$ are perfectly (resp. statistically, computationally) indistinguishable.
- Perfectly (resp. computationally) binding: given a commitment $c$, it is infeasible (resp. computationally infeasible) to output $(m; r)$ and $(m'; r')$ with $m' \neq m$ such that $\mathsf{com}(m'; r') = \mathsf{com}(m; r)$.

An encryption scheme that is $\mathsf{IND\text{-}CPA}$ is a computationally hiding and perfectly binding commitment. Some additional properties are sometimes required, like equivocality and extractability.

## B Two-Party Computation from ESP

After recalling a two-party computation model, this section proposes a framework to instantiate a two-party computation protocol from a secure encryption switching protocol on two complementary homomorphic encryption schemes. Assuming the computational equality of the underlying plaintext spaces, we show the security of our methodology in front of any malicious adversary, as stated in Theorem 8, Section 2. As already explained, we will also assume that one of the two encryption schemes admits a 2-party decryption protocol that is both *strongly sound* and *zero-knowledge* (in a similar meaning than for $\mathsf{ESP}$, see Section A.3).

### B.1 Definition

A two-party computation, 2-PC for short, consists of an interactive protocol between two players allowing them to jointly compute an evaluation of a given function $f$ on their private inputs. A setup might first define additional private inputs to the players or additional common inputs, but they will be defined by the model later. We consider a fonction $f \in \mathscr{F}$, $f : \mathscr{X} \times \mathscr{Y} \to \mathscr{Z}$.

**Definition 17 (Two-Party Computation).** *A two-party $f$-computation is an interactive protocol $\mathsf{Eval}_f(A(x), B(y))$ between $A$ and $B$ so that $A$ eventually receives $z = f(x,y)$ but $B$ does not learn anything (the convention could have been in the other way).*

The *correctness* ensures that $A$ ends with the correct value $z = f(x,y)$ if both players behave honestly.

A 2-PC is secure if it is *input-indistinguishable*: no party can extract additional information of the other party's private input from those deriving from the received output (either the result $f(x,y)$ or nothing) and its own input. In order to ease the reading, we choose a *game-based* definition.

**Definition 18 (Input-Indistinguishability).** *A two-party computation protocol is* ***input-indistinguishable*** *if it is simultaneously input-indistinguishable to $B$ and input-indistinguishable to $A$. A 2-PC is input-indistinguishable to $A$ if for all active adversary $\mathscr{A}$ playing the role of $A$ in the game:*

1. *The adversary $\mathscr{A}$ chooses $x \in \mathscr{X}$ and $y_0, y_1 \in \mathscr{Y}$, under the restriction that $f(x, y_0) = f(x, y_1)$ (since the party $A$ is supposed to eventually obtain the output of $f$), and outputs the triple $(x, y_0, y_1)$;*
2. *The challenger $\mathscr{C}$ picks $b \xleftarrow{\$} \{0,1\}$;*
3. *$\mathscr{A}$ and $\mathscr{C}$ run the interactive protocol $\mathsf{Eval}_f(\mathscr{A}(x), \mathscr{C}(y_b))$, where $\mathscr{C}$ honestly plays the role of $B$ with private input $y = y_b$;*
4. *Eventually, $\mathscr{A}$ outputs its guess $b'$;*

*its advantage $\mathsf{Adv}^{\mathsf{input\text{-}ind}}(\mathscr{A}) = 2 \times \Pr[b' = b] - 1$ is negligible.*

*A similar game is defined for the other case, against $A$, but without any restriction on the choice of $(x, y_0, y_1)$, since the party $B$ should not learn anything at the end of the protocol execution.*

In the following, we consider a specific scenario, with a setup that runs $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\kappa)$, $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}$, and $(\mathsf{pk}, \mathsf{sk}_A, \mathsf{sk}_B) \leftarrow \mathsf{Share}(\mathsf{pk}, \mathsf{sk})$. The pairs $(\mathsf{pk}, \mathsf{sk}_A)$ and $(\mathsf{pk}, \mathsf{sk}_B)$ are respectively given to $A$ and $B$, in a trustfully way.

Then, to be sure that the malicious adversary $\mathscr{A}$ engages the evaluation correctly with input $x$ in the above *input-indistinguishability* game, the challenger gives it the initial $C_x \leftarrow \mathscr{E}(x; r)$ and the random coins $r$, and of course $C_y \leftarrow \mathscr{E}(y_b; s)$, but then without the random coins $s$.

We stress that the generation and the distribution of the keys are performed by a third-trusted party. Stronger requirements allow leaving the adversary to be part of a *distributed* parameter generation. Here, we do not consider this case as it can be handled separately to our design.

### B.2 Design

We now present our framework for building a 2-PC protocol for a set $\mathscr{F}$ of functions which can be evaluated by any sequence of $\oplus$ and $\otimes$ operations, two binary laws on $\mathscr{R}$. More precisely, given complementary homomorphic encryption schemes $\mathit{\Pi}_1$ and $\mathit{\Pi}_2$, on respective computationally-equal plaintext spaces $(\mathscr{M}_1, \oplus)$ and $(\mathscr{M}_2, \otimes)$, we build a secure 2-PC for $f \in \mathscr{F}$ assuming that $\mathscr{R} = \mathscr{M}_1 \cup \mathscr{M}_2$. Without loss of generality, we suppose that $f : \mathscr{R}^{\ell_A} \times \mathscr{R}^{\ell_B} \to \mathscr{R}^{\ell}$ describes a deterministic order for these operations so that $\mathsf{Eval}_f$ can also specify when and which $\mathsf{Switch}$ has to be performed: on the inputs $x = (x_i)_i$ and $y = (y_j)_j$, that we can unify as $(m_i)_i$, for $i = 1, \ldots \ell_A + \ell_B$, we assume the evaluation of $f$ can be iteratively performed, for $k = \ell_A + \ell_B + 1, \ldots, n$, with two already computed elements $i, j < k$,

$$\mathsf{Op}'_\oplus(i,j;k)\colon m_k \leftarrow m_i \oplus m_j; \qquad\qquad \mathsf{Op}'_\otimes(i,j;k)\colon m_k \leftarrow m_i \otimes m_j.$$

Now, given the ciphertexts $(c_i)_i$ of the inputs $(m_i)_i \in \mathscr{R}^{\ell_A} \times \mathscr{R}^{\ell_B}$, under the encryption scheme $\mathit{\Pi}_1$, each new step consists, for $k > i, j$

- $\mathsf{Op}_\oplus(i,j;k)$: $c_k \leftarrow c_i \boxplus c_j$, with the deterministic homomorphism of $\Pi_1$, computing first $c_i$ and $c_j$, using $\mathsf{Sw}_\oplus$ below, if needed;
- $\mathsf{Op}_\otimes(i,j;k)$: $c'_k \leftarrow c'_i \boxtimes c'_j$, with the deterministic homomorphism of $\Pi_2$, computing first $c'_i$ and $c'_j$, using $\mathsf{Sw}_\otimes$ below, if needed;
- $\mathsf{Sw}_\otimes(i)$: $c'_i \leftarrow \mathsf{Switch}_{1\to 2}(c_i)$;
- $\mathsf{Sw}_\oplus(i)$: $c_i \leftarrow \mathsf{Switch}_{2\to 1}(c'_i)$.

This construction of the 2-PC thus starts with encryptions of $x \in \mathscr{R}^{\ell_A}$ and $y \in \mathscr{R}^{\ell_B}$ and ends with an encryption of $z = f(x,y) \in \mathscr{R}^\ell$. Hence a 2-party decryption provides the result to $A$ only.

**Construction $\boldsymbol{\Pi_{2\mathrm{PC}}}$.** We assume that we have $\Pi_1 \rightleftharpoons \Pi_2 = (\mathsf{Share}, \mathsf{Switch})$ an ESP on $(\mathscr{R}, \oplus, \otimes)$ for appropriate homomorphic public-key encryption schemes $\Pi_i = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}_i, \mathsf{Dec}_i)$, for $i = 1, 2$. Furthermore, we assume that one of the two encryption schemes admits an efficient 2-party decryption protocol that gives the result to one party only, where the shares of the decryption key are the same as for the ESP. Finally, let $f$ be a function described by a sequence of operations among $\mathsf{Op}_\oplus(i,j;k)$, $\mathsf{Op}_\otimes(i,j;k)$, $\mathsf{Sw}_\otimes(i)$, and $\mathsf{Sw}_\oplus(i)$.

To initialize the process, we run $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\kappa)$, $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$, and $(\mathsf{pk}, \mathsf{sk}_A, \mathsf{sk}_B) \leftarrow \mathsf{Share}(\mathsf{pk}, \mathsf{sk})$. The evaluation of $f(x = (x_i)_i, y = (y_j)_j) \in \mathscr{R}^\ell$ proceeds as follows:

1. $A$ runs $c_i \leftarrow \mathsf{Enc}_1(\mathsf{pk}, x_i)$, for $i = 1, \dots, \ell_A$;
2. $B$ runs $c_{j+\ell_A} \leftarrow \mathsf{Enc}_1(\mathsf{pk}, y_j)$, for $j = 1, \dots, \ell_B$;
3. for $\mathsf{Op}_\oplus$ and $\mathsf{Op}_\otimes$ operations, $A$ and $B$ use the deterministic homomorphic properties of the corresponding $\Pi_1$ and $\Pi_2$;
4. for $\mathsf{Sw}_\otimes$ and $\mathsf{Sw}_\oplus$ operations, to switch between $\mathsf{Enc}_1$ and $\mathsf{Enc}_2$ in the appropriate direction $\mathsf{par} = {\scriptstyle 1\to 2}$ or $\mathsf{par} = {\scriptstyle 2\to 1}$, they run $\mathsf{Switch}_{\mathsf{par}}((\mathsf{pk}, \mathsf{sk}_A, c), (\mathsf{pk}, \mathsf{sk}_B, c))$ on the appropriate ciphertext $c$;
5. the final 2-party decryptions $\mathsf{Dec}((\mathsf{sk}_A, \boldsymbol{c}), (\mathsf{sk}_B, \boldsymbol{c}))$ are performed on the $\ell$-block ciphertext that contains the final plaintext $z \in \mathscr{R}^\ell$, so that $A$ only gets the result.

The 2-PC correctness of $\Pi_{2\mathrm{PC}}$ follows from homomorphisms of the two encryption schemes and the correctness of both the encryption switching protocols and the 2-party decryption.

We note that if $\mathscr{M}_1 \oplus \mathscr{M}_2$ would have a non-negligible size relatively to $\mathscr{M}_1 \cap \mathscr{M}_2$, a random sample in $\mathscr{R} = \mathscr{M}_1 \cup \mathscr{M}_2$ would have a non-negligible chance to pick a message in the former set, which would contradict the computational-equality assumption. However, inputs are not chosen at random, hence the additional computational assumption. We show below this assumption is enough, together with the ESP security and the IND-CPA security of the two encryption schemes, as well as the security (soundness and simulatability) of the final 2-party decryption.

**Theorem 8.** *Let $\Pi_1$ and $\Pi_2$ 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 $\Pi_2$ 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 $(\mathscr{R}, \oplus, \otimes)$.*

*Proof.* The challenger $\mathscr{C}$ is given the public-key $\mathsf{pk}$ of the ESP which can be used to encrypt any $m \in \mathscr{R}$ by running the encryption algorithm of $\Pi_1$. It can also compute the homomorphic operations on ciphertexts of $\mathscr{C}_1$ or $\mathscr{C}_2$. From $f \in \mathscr{F}$, one derives the sequence of operations to be performed to evaluate $f(x, y_b)$.

We now assume that $\mathscr{A}$ chooses to play the role of $A$, and thus gets $\mathsf{sk}_A$ (including $A$'s decryption key share), in order to show that 2-PC is input-indistinguishable to $A$ —a simpler argument works for the choice of $B$. Then, the adversary chooses the private inputs $x \in \mathscr{R}^{\ell_A}$, $y_0, y_1 \in \mathscr{R}^{\ell_B}$ satisfying $f(x, y_0) = f(x, y_1)$. In the following, we will denote $(\pi_1^b, \dots, \pi_{q_S}^b)$ the sequence of plaintexts of the ordered ciphertexts involved in the switching protocols during an honest execution to evaluate $f(x, y_b)$, where $q_S$ is the number of switches. We will also denote $(y_b^{(1)}, \dots, y_b^{(\ell_B)})$ the sequence of individual plaintexts in $y_b$.

Hereunder, we describe a sequence of games to show the indistinguishability of the execution on $(x, y_0)$ and the execution on $(x, y_1)$ while $\mathscr{A}$ still believes it is playing with the first input pair. In each game $\mathbf{G}_j$, let $S$ be the success event: the bit $b'$ output by the adversary is equal to 0. We will also denote $t$, the overall time of the games, including the execution time of the adversary and the execution time of the simulation.

**Game $\mathbf{G}_0$:** This is the real game where the 2-PC game is run on the internal bit $b = 0$: the simulator $\mathscr{S}im$ emulates the challenger, by first generating all the keys, then the adversary $\mathscr{A}$ chooses $(x, y_0, y_1)$ and $\mathscr{S}im$ chooses $b = 0$. It honestly runs the interactive protocol with $\mathscr{A}$ on $(x, y_0)$, using all the private informations. We have $S$ the event in which $\mathscr{A}$ wins the 2-PC game by returning $b' = b = 0$.

**Game $\mathbf{G}_1$:** In this game, we just introduce a failure event $F$ which causes the simulator to abort with output 1 if, among the sequences $(y_0^{(1)}, \dots, y_0^{(\ell_B)})$ and $(\pi_1^0, \dots, \pi_{q_S}^0)$ of plaintexts, one message falls outside $\mathscr{M}_1 \cap \mathscr{M}_2$. We thus have

$$|\Pr_{\mathbf{G}_1}[S] - \Pr_{\mathbf{G}_0}[S]| \le \Pr_{\mathbf{G}_1}[F],$$

but we cannot evaluate yet $\Pr_{\mathbf{G}_1}[F]$. We will be able to do it later, when we will be able to apply the computationally equality between $\mathscr{M}_1$ and $\mathscr{M}_2$, without the knowledge of the secret keys.

**Game $\mathbf{G}_2$:** In this game, the simulator aborts and outputs 1 if at some point in the execution the output of a switch does not encrypt the expected plaintext in the sequence $(\pi_1^0, \dots, \pi_{q_S}^0)$. Since the simulation knows the secret keys, it can check that. But then, this would mean that the adversary has broken the soundness of the $\mathsf{ESP}$, which does not rely one any problem that becomes easy when the secret keys are known:

$$|\Pr_{\mathbf{G}_2}[S] - \Pr_{\mathbf{G}_1}[S]| \le \mathsf{Succ}^{\textsf{esp-sound}}(\kappa, t)$$

(as well as for the event $F$).

**Game $\mathbf{G}_3$:** In this game, the simulator aborts and outputs 1 if the last $\ell$-block ciphertext does not decrypt to the appropriate result $z \in \mathscr{R}^\ell$. Note that the simulator can again check the validity with the secret keys. But then, as above, this would mean that the adversary has broken the soundness of the 2-party decryption, which is assumed to be statistically sound:

$$|\Pr_{\mathbf{G}_3}[S] - \Pr_{\mathbf{G}_2}[S]| \le \mathsf{Succ}^{\textsf{dec-sound}}(\kappa, t)$$

(as well as for the event $F$).

**Game $\mathbf{G}_4$:** Here, we bring another modification by replacing the honest run of the 2-party decryption on the last $\ell$-block ciphertext with the decrypting simulator $\mathscr{S}im_B^{\mathsf{Dec}}$ using the expected output $z = f(x, y_0) = f(x, y_1)$. Any noticeable difference would break the *statistical* zero-knowledge property of this decryption protocol:

$$|\Pr_{\mathbf{G}_4}[S] - \Pr_{\mathbf{G}_3}[S]| \le \mathsf{Adv}^{\textsf{dec-zk}}(\kappa, t)$$

(as well as for the event $F$).

**Game $\mathbf{G}_5$:** In this game, we do as above, but for the $\mathsf{ESP}$s: we use the pair of simulators $(\mathscr{S}im_B^{\mathsf{share}}, \mathscr{S}im_B^{\mathsf{ESP}})$ using the expected output that $\mathsf{Switch}$ would do as in the previous game. Note that these outputs can be computed from the known sequence of plaintexts $(\pi_1^0, \dots, \pi_{q_S}^0)$. Any noticeable difference would break the zero-knowledge property of the $\mathsf{ESP}$:

$$|\Pr_{\mathbf{G}_5}[S] - \Pr_{\mathbf{G}_4}[S]| \le \mathsf{Adv}^{\textsf{esp-zk}}(\kappa, t)$$

(as well as for the event $F$).

Note that now, because of the $\mathsf{ESP}$ simulation, no secret keys are known anymore since the simulator just receives as input an appropriate $\mathsf{pk}$. Hence, we can evaluate the event $F$, since the bad plaintext would break the computational equality: $\Pr_{\mathbf{G}_5}[F] \le \mathsf{Succ}^{\textsf{comp-eq}}(\kappa, t)$. Since $|\Pr_{\mathbf{G}_5}[F] - \Pr_{\mathbf{G}_1}[F]| \le \mathsf{Succ}^{\textsf{esp-sound}}(\kappa, t) + \mathsf{Succ}^{\textsf{dec-sound}}(\kappa, t) + \mathsf{Adv}^{\textsf{dec-zk}}(\kappa, t) + \mathsf{Adv}^{\textsf{esp-zk}}(\kappa, t)$, we get:

$$\begin{aligned}\Pr_{\mathbf{G}_1}[F] \le \mathsf{Succ}^{\textsf{esp-sound}}(\kappa, t) + \mathsf{Succ}^{\textsf{dec-sound}}(\kappa, t)\\ + \mathsf{Adv}^{\textsf{dec-zk}}(\kappa, t) + \mathsf{Adv}^{\textsf{esp-zk}}(\kappa, t) + \mathsf{Succ}^{\textsf{comp-eq}}(\kappa, t)\end{aligned}$$

and then

$$|\Pr_{\mathbf{G}_5}[S] - \Pr_{\mathbf{G}_0}[S]| \leq 2 \times \mathsf{Succ}^{\mathsf{esp\text{-}sound}}(\kappa, t) + 2 \times \mathsf{Succ}^{\mathsf{dec\text{-}sound}}(\kappa, t) \\ + 2 \times \mathsf{Adv}^{\mathsf{dec\text{-}zk}}(\kappa, t) + 2 \times \mathsf{Adv}^{\mathsf{esp\text{-}zk}}(\kappa, t) + \mathsf{Succ}^{\mathsf{comp\text{-}eq}}(\kappa, t).$$

**Game $\mathbf{G}_6$:** Now, we just replace the input $y_0$ by $y_1$, as input to $\Pi_1$ during the initial phase of the protocol, but we still use the sequence of plaintexts $(\pi_1^0, \ldots, \pi_{q_S}^0)$ to generate the additional inputs (to force the output) of the ESP simulators. That means that the additional input ciphertexts of $\mathcal{S}im_B^{\mathsf{ESP}}$ computed by $\mathcal{S}im$ remain the same as in the previous game. In order to show the indistinguishability with the previous game, we use an hybrid argument on the $\ell_B$ entries of $y_0$ where one can apply the IND-CPA game of $\Pi_1$ on the message pair $(y_0^{(i)}, y_1^{(i)})$, but under the condition they both lie in the intersection $\mathcal{M}_1 \cap \mathcal{M}_2$, otherwise one breaks the computational equality. This can be checked once on the sequence $(y_1^{(1)}, \ldots, y_1^{(\ell_B)})$:

$$|\Pr_{\mathbf{G}_6}[S] - \Pr_{\mathbf{G}_5}[S]| \leq \ell_B \times \mathsf{Adv}_{\Pi_1}^{\mathsf{ind\text{-}cpa}}(\kappa, t) + \mathsf{Succ}^{\mathsf{comp\text{-}eq}}(\kappa, t).$$

**Game $\mathbf{G}_7$:** For $i = 1$ to $q_S$, we gradually replace the $i^{th}$ plaintext $\pi_i^0$, in the sequence of plaintexts, by $\pi_i^1$ to generate the ciphertext for the output of the $i^{th}$ ESP simulation. This leads to $q_S$ hybrid games, where one can apply the IND-CPA game of either $\Pi_1$ or $\Pi_2$ on the message pair $(\pi_i^0, \pi_i^1)$, but under the condition they both lie in the intersection $\mathcal{M}_1 \cap \mathcal{M}_2$, otherwise one breaks the computational equality. This can be checked once on the sequence $(\pi_1^1, \ldots, \pi_{q_S}^1)$:

$$|\Pr_{\mathbf{G}_7}[S] - \Pr_{\mathbf{G}_6}[S]| \leq q_S \times \max\{\mathsf{Adv}_{\Pi_1}^{\mathsf{ind\text{-}cpa}}(\kappa, t), \mathsf{Adv}_{\Pi_2}^{\mathsf{ind\text{-}cpa}}(\kappa, t)\} \\ + \mathsf{Succ}^{\mathsf{comp\text{-}eq}}(\kappa, t).$$

**Game $\mathbf{G}_8$:** In this game, the simulator is given again the secret keys, and runs honestly the ESPs. Since the input-output ciphertexts were consistent, any noticeable difference would break the zero-knowledge property of the ESP, as in the Game $\mathbf{G}_5$:

$$|\Pr_{\mathbf{G}_8}[S] - \Pr_{\mathbf{G}_7}[S]| \leq \mathsf{Adv}^{\mathsf{esp\text{-}zk}}(\kappa, t).$$

**Game $\mathbf{G}_9$:** Here, we remove the modification brought in $\mathbf{G}_4$, *i.e.* honestly run the decryption protocol:

$$|\Pr_{\mathbf{G}_9}[S] - \Pr_{\mathbf{G}_8}[S]| \leq \mathsf{Adv}^{\mathsf{dec\text{-}zk}}(\kappa, t).$$

**Game $\mathbf{G}_{10}$:** We now remove the early aborts that led to a success for the adversary:

$$\Pr_{\mathbf{G}_{10}}[S] \leq \Pr_{\mathbf{G}_9}[S].$$

This game is now exactly the real game with $b = 1$: $\Pr_{\mathbf{G}_0}[S] = \Pr[b' = 0 | b = 0]$ and $\Pr_{\mathbf{G}_{10}}[S] = \Pr[b' = 0 | b = 1]$.

Hence, this shows that

$$\mathsf{Adv}_{\Pi_{2\mathrm{PC}}}^{\mathsf{input\text{-}ind}}(\kappa, t) \leq 2 \times \mathsf{Succ}^{\mathsf{esp\text{-}sound}}(\kappa, t) + 2 \times \mathsf{Succ}^{\mathsf{dec\text{-}sound}}(\kappa, t) \\ + 2 \times \mathsf{Adv}^{\mathsf{dec\text{-}zk}}(\kappa, t) + 2 \times \mathsf{Adv}^{\mathsf{esp\text{-}zk}}(\kappa, t) + \mathsf{Succ}^{\mathsf{comp\text{-}eq}}(\kappa, t) \\ + \ell_B \times \mathsf{Adv}_{\Pi_1}^{\mathsf{ind\text{-}cpa}}(\kappa, t) + \mathsf{Succ}^{\mathsf{comp\text{-}eq}}(\kappa, t) \\ + q_S \times \max\{\mathsf{Adv}_{\Pi_1}^{\mathsf{ind\text{-}cpa}}(\kappa, t), \mathsf{Adv}_{\Pi_2}^{\mathsf{ind\text{-}cpa}}(\kappa, t)\} \\ + \mathsf{Succ}^{\mathsf{comp\text{-}eq}}(\kappa, t) + \mathsf{Adv}^{\mathsf{esp\text{-}zk}}(\kappa, t) + \mathsf{Adv}^{\mathsf{dec\text{-}zk}}(\kappa, t).$$

It simplifies to

$$\mathsf{Adv}_{\Pi_{2\mathrm{PC}}}^{\mathsf{input\text{-}ind}}(\kappa, t) \leq 2 \times \mathsf{Succ}^{\mathsf{esp\text{-}sound}}(\kappa, t) + 2 \times \mathsf{Succ}^{\mathsf{dec\text{-}sound}}(\kappa, t) \\ + 3 \times \mathsf{Adv}^{\mathsf{dec\text{-}zk}}(\kappa, t) + 3 \times \mathsf{Adv}^{\mathsf{esp\text{-}zk}}(\kappa, t) + 3 \times \mathsf{Succ}^{\mathsf{comp\text{-}eq}}(\kappa, t) \\ + (\ell_B + q_S) \times \max\{\mathsf{Adv}_{\Pi_1}^{\mathsf{ind\text{-}cpa}}(\kappa, t), \mathsf{Adv}_{\Pi_2}^{\mathsf{ind\text{-}cpa}}(\kappa, t)\}$$

□

### B.3 Evaluating Any Function on a Ring Structure

Any function $f$ can be expressed as a circuit composed of NAND gates only. On a ring structure, a NAND gate can be evaluated by using the identity element for the law $\otimes$ as 1, the identity element for the law $\oplus$ as 0, and the following relation: for all $(b, b') \in \{0,1\}^2, b \ \mathsf{NAND}\ b' \leftarrow 1 \oplus (-b \otimes b')$. Nevertheless, ESPs are tailored to get more efficient two-party computation protocols of any sequence of operations $\oplus$ and $\otimes$ over a ring structure $(\mathscr{R}, \oplus, \otimes)$ whereas other cryptographic primitives like garbled circuits are tailored to get efficient two-party protocols computation evaluating boolean functions.

## C Security of the Encryption Switching Protocols

In this Appendix, we provide the full proofs of Theorems 10 and 11 that claim the zero-knowledge property for our encryption switching protocols on both $\mathbb{Z}_n^*$ and $\mathbb{Z}_n$.

### C.1 Encryption Switching Protocols on $\mathbb{Z}_n^*$

Let us recall Theorem 10:

**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 $\mathsf{QR}_n$, the $\mathsf{QR}$ assumption in $\mathbb{Z}_n^*$, the $\mathsf{DCR}$ assumption over $\mathbb{Z}_n^*$, and the $\mathsf{DCR}$ assumption over $\mathbb{Z}_N^*$.*

**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 $\lambda$);
  - $d \leftarrow [\lambda^{-1} \mod n] \cdot \lambda \bmod n\lambda$: $d = 0 \bmod \lambda$ and $d = 1 \bmod n$;
  - $v \leftarrow [p^{-1} \bmod q] \cdot p \bmod n$: $v = 0 \bmod p$ 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 $\mathbb{J}_n$-$\mathsf{EG}$).

– The additional modulus $N$:
  - $P, Q$ two strong primes, $N \leftarrow PQ$ (such that $N > (2 + 2^{\kappa+1})n^2$);
  - $D \leftarrow [\mathit{\Lambda}^{-1} \mod N] \cdot \mathit{\Lambda} \bmod N\mathit{\Lambda}$, where $\mathit{\Lambda}$ 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: $(d_A, v_A, t_{pA}, t_{qA}, s_A, D_A) \xleftarrow{\$} \mathbb{Z}_{n\lambda} \times \mathbb{Z}_n \times \mathbb{Z}_\lambda^3 \times \mathbb{Z}_{N\mathit{\Lambda}}$, 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 $D_B \leftarrow D - D_A \bmod N\mathit{\Lambda}$.

**$\mathscr{E}_\otimes(\cdot) = \mathbb{Z}_n^*$-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 $C \leftarrow \mathbb{J}_n\text{-}\mathsf{EG}.\mathsf{Enc}(m_2; r) = (c_0 = g^r,\ c_1 = m_2 g_1^r)$.
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 $m_2 \leftarrow \mathbb{J}_n\text{-}\mathsf{EG}.\mathsf{Dec}(C) = c_1/c_0^s$ 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 $\mathbb{Z}_n^*$ (repeated from page 11)

**2-Party $\mathsf{ESP}^{\times}_{+}$ from $C = \mathscr{E}_{\oplus}(m)$ into $C' = \mathscr{E}_{\otimes}(m)$**

$$\begin{array}{lcl}
R_A \overset{\$}{\leftarrow} \mathbb{Z}_n^*, C'_A \leftarrow \mathscr{E}_{\otimes}(R_A^{-1}) & & \\
C_A \leftarrow \mathbb{Z}_n\text{-}\mathsf{P.Rand}(R_A \bullet C) & & \\
C_1 \leftarrow C_A^{d_A} \bmod n^2 & \xrightarrow{C'_A, C_A, C_1} & x \leftarrow ([C_1 \times C_A^{d_B} \bmod n^2] - 1)/n \\
 & \xleftarrow{\quad C' \quad} & C' \leftarrow \mathbb{Z}_n^*\text{-}\mathsf{EG.Rand}(x \bullet C'_A)
\end{array}$$

**2-Party $\mathsf{ESP}^{+}_{\times}$ from $C = \mathscr{E}_{\otimes}(m)$ into $C' = \mathscr{E}_{\oplus}(m)$**

$$\begin{array}{lcr}
R_A \overset{\$}{\leftarrow} \mathbb{Z}_n^*, C'_A \leftarrow \mathscr{E}_{\oplus}(R_A^{-1}) & & \\
C_A \leftarrow \mathbb{Z}_n^*\text{-}\mathsf{EG.Rand}(R_A \bullet C) & & \\
\quad = (C_0, C_1, \alpha) & & \\
C_2 \leftarrow C_0^{s_A} & \xrightarrow{C'_A, C_A, C_2} & C_3 \leftarrow C_0^{s_B}, \beta \leftarrow C_1/C_2C_3 \\
 & & r_1 \overset{\$}{\leftarrow} \mathbb{Z}_{n/2}, B \leftarrow \alpha g^{r_1}, B' \leftarrow \chi^{-r_1} \\
 & & (u_1, u_2) \leftarrow ([v_B B' \bmod n], [B' \bmod n]) \\
A_1 \leftarrow (1 - v_A)B^{t_{pA}} & & \\
A_2 \leftarrow v_A B^{t_{qA}} & \xleftarrow{B, B_1, B_2} & (B_1, B_2) \leftarrow (B^{t_{pB}}, B^{t_{qB}}) \\
A_3 \leftarrow -B^{t_{pA}} B_1 + B^{t_{qA}} B_2 & & \\
A_4 \leftarrow A_1 B_1 + A_2 B_2 & & \\
(r_3, r_4, k) \overset{\$}{\leftarrow} {\mathbb{Z}_N^*}^2 \times \mathbb{Z}_{2^{\kappa+1}n} & & \\
E_3 \leftarrow \mathbb{Z}_N\text{-}\mathsf{P.Enc}(A_3; r_3) & & \\
E_4 \leftarrow \mathbb{Z}_N\text{-}\mathsf{P.Enc}(A_4; r_4) & \xrightarrow{\quad E_3, E_4 \quad} & r_5 \overset{\$}{\leftarrow} \mathbb{Z}_N^* \\
E_6 \leftarrow \mathbb{Z}_N\text{-}\mathsf{P.Rand}(kn \boxplus E_5) & \xleftarrow{\quad E_5 \quad} & E_5 \leftarrow E_3^{u_1} E_4^{u_2} \times r_5^N \bmod N^2 \\
F_6 \leftarrow E_6^{D_A} \bmod N^2 & \xrightarrow{\quad E_6, F_6 \quad} & m_6 \leftarrow ([F_6 E_6^{D_B} \bmod N^2] - 1)/N \\
 & & x \leftarrow \beta[m_6 \bmod n] \bmod n \\
 & \xleftarrow{\quad C' \quad} & C' \leftarrow \mathbb{Z}_n\text{-}\mathsf{P.Rand}(x \bullet C'_A)
\end{array}$$

Fig. 2: Interactive Protocols for Decryption and Encryption Switching in $\mathbb{Z}_n^*$ (repeated from page 13)

The proof is done in two parts: first we show how to simulate Alice in front of an adversarial (honest-but-curious) Bob, which will prove the zero-knowledge property for Alice, and then how to simulate Bob in front of an adversarial (honest-but-curious) Alice, which will prove the zero-knowledge property for Bob.

Note that in order to anticipate the security proof in the malicious setting, we will incorporate $(h, h_p = h^{t_p}, h_q = h^{t_q})$ in the public key. It will indeed later be used to prove the honest behavior with additional zero-knowledge proofs.

*$\mathbb{Z}_n^*$−ESP is Zero-Knowledge for Alice.* Let us first exhibit the simulator, we then prove that the behavior of the simulator is indistinguishable from Alice's behavior.

$\mathscr{S}im$ first receives as input two strong RSA moduli $n$ and $N$ with $N > (2 + 2^{\kappa+1})n^2$. It picks $g, g_1 \overset{\$}{\leftarrow} \mathbb{J}_n$ and $\chi \overset{\$}{\leftarrow} \mathbb{Z}_n^* \setminus \mathbb{J}_n$. It also picks $h, h_p, h_q \overset{\$}{\leftarrow} \mathbb{J}_n$. It then sets $\mathsf{pk} \leftarrow (n, g, \chi, g_1, N, h, h_p, h_q)$. It picks $(d_B, v_B, t_{pB}, t_{qB}, s_B, D_B) \overset{\$}{\leftarrow} \mathbb{Z}_{n^2/2} \times \mathbb{Z}_n \times \mathbb{Z}_{n/2}^3 \times \mathbb{Z}_{N^2/2}$ and sends them to the adversarial Bob, together with $\mathsf{pk}$.

Each time $\mathscr{S}im$ is asked to participate to an instance of a $\mathbb{Z}_n^*$−$\mathsf{ESP}$ on an input $C$, it additionally receives a target ciphertext $\bar{C}$, and the expected output is $\bar{C}$ or an equivalent ciphertext:

– If the direction is from additive to multiplicative,
  - $\mathscr{S}im$ picks $x \overset{\$}{\leftarrow} \mathbb{Z}_n^*$ and $C_A \overset{\$}{\leftarrow} \mathbb{Z}_{n^2}^*$, and sends $(C'_A \leftarrow x^{-1} \bullet \bar{C}, C_A, C_1 \leftarrow (1 + nx) \times {C_A}^{-d_B} \bmod n^2)$ to Bob.

– If the direction is from multiplicative to additive,
  - $\mathscr{S}im$ picks $x \overset{\$}{\leftarrow} \mathbb{Z}_n^*$, $C_A \overset{\$}{\leftarrow} \mathbb{J}_n^3$, and $\beta \overset{\$}{\leftarrow} \mathbb{J}_n$, and sends $(C'_A \leftarrow x^{-1} \bullet \bar{C}, C_A, C_2 \leftarrow C_1 C_0^{-s_B}/\beta)$ to Bob;
  - $\mathscr{S}im$ picks and sends $E_3, E_4 \overset{\$}{\leftarrow} \mathbb{Z}_{N^2}^*$;
  - $\mathscr{S}im$ picks $(k', \rho) \overset{\$}{\leftarrow} \mathbb{Z}_{2^{\kappa+1}n} \times \mathbb{Z}_N$; he sends $E_6 \leftarrow \mathbb{Z}_N\text{-}\mathsf{P.Enc}(x/\beta + k'n; \rho)$ and $F_6 \leftarrow (1 + N(x/\beta + k'n)) \cdot E_6^{-D_B} \bmod N^2$.

With the following sequence of games, we show that the simulator $\mathscr{S}im$ has the same behavior as Alice, even for many instances. The simulators will be constructed step by step, starting from a simulator that knows all the secrets, and then behaves exactly as Alice.

**Sketch of the Proof.** The proof is done by exhibiting a sequence of games, so that in the end, the actual simulation does not use any secret information, but the expected output $\bar{C}$. First, $\mathscr{S}im$ replaces the flow $(C_A = \mathscr{E}(R_A \cdot m), C'_A = \bar{\mathscr{E}}(R_A^{-1}))$ by $(C_A = \mathscr{E}(x), C'_A = x \bullet \bar{C})$ for a random $x$. This allows it to know the plaintext $x$ to be decrypted in $C_A$, and that is necessary to simulate the decryption phase. The simulation is perfect as there is some $R_A$ such that $x = R_A \cdot m$ and if $\bar{C}$ encrypts $m$, then $x^{-1} \bullet \bar{C}$ encrypts $R_A^{-1}$.

The additive to multiplicative direction is then simple, thanks to the zero-knowledge property of the 2-party Paillier decryption. For the multiplicative to additive direction, during the 2-party $\mathbb{Z}_n^*$-$\mathsf{EG}$ decryption, the output is reconstructed by using the homomorphic properties of another Paillier scheme (in $\mathbb{Z}_{N^2}^*$), so that the $\mathsf{IND\text{-}CPA}$ security of the Paillier scheme ensures that no intermediate information is leaked, even when knowing the factorization of $n$. These intermediate ciphertexts will thereafter be replaced by random ciphertexts and the output will be computed from $x$. $\mathscr{S}im$ will also generate the public values at random, in particular $\chi = (1 - v) \cdot g^{t_p} + v \cdot g^{t_q}$. Thus, we need an additional lemma which shows that if $\mathsf{DDH}$ holds over $\mathsf{QR}_n$, then taking these values independently at random is indistinguishable from an honest generation from the keys, even knowing the factorization of $n$. A last crucial detail of the proof is that we know the parity of $t_p$ and $t_q$ (to make $\chi \in \mathbb{Z}_n^* \setminus \mathbb{J}_n$). The simulator manipulates elements of $\mathbb{J}_n$ but relies on the security of $\mathsf{DDH}$ over $\mathsf{QR}_n$, and elements of $\mathbb{J}_n$ can be seen as elements of the form $(-1)^b S$ with $S \in \mathsf{QR}_n$. When constructing $\alpha^{t_p}$ and $\alpha^{t_q}$ from a $\mathsf{DDH}$ challenge over $\mathsf{QR}_n$, it is necessary to know the parity of the exponent to generate an element of $\mathbb{J}_n$ with the correct sign bit. In the end, when the factorization of $n$ is unknown, under the $\mathsf{QR}$ assumption, we can take all the random elements in $\mathbb{J}_n$ instead of $\mathsf{QR}_n$.

**Game $\mathbf{G}_0$:** This is the real game.

**Input.** In this game, $\mathscr{S}im$ knows the factorization of both $n$ and $N$, with two generators $g, h$ of $\mathbb{J}_n$.

**Setup.** $\mathscr{S}im$ generates all the secret keys for both Alice and Bob in an honest way, gives his secret key to Bob.

**Switches.** Each time it is asked to participate to an instance of the switching protocol on an input $(C, \bar{C})$, $\mathscr{S}im$ plays honestly the real game with the secret key of Alice.

**Game $\mathbf{G}_1$:** We start by modifying the simulation in the **additive to multiplicative direction**.

**Input and Setup.** As in the previous game.

**Switches.** For each $\mathsf{ESP}_+^\times(C, \bar{C})$, from additive to multiplicative, $\mathscr{S}im$ generates $x \xleftarrow{\$} \mathbb{Z}_n^*$, sets and sends $(C'_A, C_A) \leftarrow (x^{-1} \bullet \bar{C}, \mathscr{E}_\oplus(x))$ to Bob. It then honestly plays the rest of the protocol.

When $(C, \bar{C})$ are twin ciphertexts, with a random $\bar{C}$, this game is perfectly indistinguishable from the previous one, since this corresponds to use $R_A = x/m \bmod n$, where $m$ is the plaintext in $C$, and $x$ follows a uniform distribution in $\mathbb{Z}_n^*$. In addition, the actual value of $R_A$ is not required in the rest of the protocol.

**Game $\mathbf{G}_2$:** We continue by modifying the simulation of the decryption of the Paillier ciphertext.

**Input and Setup.** As in the previous game.

**Switches.** For each $\mathsf{ESP}_+^\times(C, \bar{C})$, $\mathscr{S}im$ generates $x$ and $(C'_A, C_A)$ as above. For the Paillier decryption, it sends $C_1 \leftarrow (1 + nx) \times C_A^{-d_B}$ to Bob.

We should have $C_1 = C^{d_A} \bmod n^2$, so that $C_1 \times C_A^{d_B} = C_A^d = (1 + nx) \bmod n^2$. Hence, $C_1 = (1+nx)/C_A^{d_B} \bmod n^2$, which shows that this game is perfectly indistinguishable from the previous one.

**Game $\mathbf{G}_3$:** We now start to address the **multiplicative to additive direction**.

**Input and Setup.** As in the previous game.

**Switches.** For each $\mathsf{ESP}_\times^+(C, \bar{C})$, from multiplicative to additive, $\mathscr{S}im$ picks a random encoding $(\alpha = g^a, \beta)$, for $a \xleftarrow{\$} \mathbb{Z}_{n/2}$ and $\beta \xleftarrow{\$} \mathbb{J}_n$, which defines $x = \beta \cdot \chi^a$ to be used to generate $(C'_A, C_A) \leftarrow (x^{-1} \bullet \bar{C}, \mathscr{E}_\otimes(x))$ to Bob. It then honestly plays the rest of the protocol.

When $(C, \bar{C})$ are twin ciphertexts, this game is perfectly indistinguishable from the previous one, since this corresponds to use $R_A = x/m \bmod n$, where $m$ is the plaintext in $C$, and $x$ follows a (statistically) uniform distribution in $\mathbb{Z}_n^*$, with a random encoding for the encryptions. In addition, the actual value of $R_A$ is not required in the rest of the protocol.

**Game $\mathbf{G}_4$:** We continue by modifying the simulation of the decryption of the ElGamal ciphertext.

**Input and Setup.** As in the previous game.

**Switches.** For each $\mathsf{ESP}_\times^+(C, \bar{C})$, $\mathscr{S}im$ generates $x$ as above with $(\alpha, \beta)$, computes $C_2 \leftarrow C_1 C_0^{-s_B}/\beta$ for the first flow, and plays honestly the rest of the protocol.

We should have $\beta = C_1/C_2C_3$ where $C_3 = C_0^{s_B}$, hence this game is perfectly indistinguishable from the previous one. Thus we do not need to know $s_A$ anymore.

**Game $\mathbf{G}_5$:** We alter the simulation of $E_6$.

**Input and Setup.** As in the previous game.

**Switches.** For each $\mathsf{ESP}_\times^+(C, \bar{C})$, $\mathscr{S}im$ does as above until the generation of $E_6$: he picks $(k', \rho) \overset{\$}{\leftarrow} \mathbb{Z}_{2^{\kappa+1}n} \times \mathbb{Z}_N$. Then, he sets $E_6 \leftarrow \mathbb{Z}_N\text{-}\mathsf{P.Enc}(x/\beta + k'n; \rho)$ and $F_6 \leftarrow E_6^{D_A} \bmod N^2$.

$E_6$ should be an encryption of $A_3u_1 + A_4u_2 + kn = x/\beta + k''n$ for some $k'' \geq k$. As $A_3u_1 + A_4u_2 \leq 2n^2$, the spaces $\{x/\beta + k'n \mid k' \in \mathbb{Z}_{2^{\kappa+1}n}\}$ and $\{A_3u_1 + A_4u_2 + kn \mid k \in \mathbb{Z}_{2^{\kappa+1}n}\}$ are statistically close: computing $E_6$ as $E_6 \leftarrow \mathbb{Z}_N\text{-}\mathsf{P.Enc}(x/\beta + k'n; \rho)$ is statistically indistinguishable from the honest generation of $E_6$, hence this game is statistically indistinguishable from the previous one.

**Game $\mathbf{G}_6$:** We alter the simulation of the decryption of $E_6$.

**Input and Setup.** As in the previous game.

**Switches.** For each $\mathsf{ESP}_\times^+(C, \bar{C})$, $\mathscr{S}im$ does as above until the generation of $F_6$: he sets $F_6 \leftarrow (1 + (x/\beta + k'n)N) \times E_6^{-D_B} \bmod N^2$.

We should have $E_6^{D_A} = E_6^D E_6^{-D_B}$ and $E_6^D = (1 + (x/\beta + k'n)N) \bmod N^2$, hence this game is perfectly indistinguishable from the previous one. One can note that $\mathscr{S}im$ does not need anymore the factorization of $N$.

**Game $\mathbf{G}_7$:** We continue with the second flow of Alice.

**Input.** As above, excepted that $\mathscr{S}im$ does not know anymore the factorization of $N$.

**Setup.** As above, excepted $D_B \overset{\$}{\leftarrow} \mathbb{Z}_{N^2/2}$ (note that $D_A$ is not needed since the previous game).

**Switches.** For each $\mathsf{ESP}_\times^+(C, \bar{C})$, $\mathscr{S}im$ does as above until the generation of $A_3$ and $A_4$: $A_3 \leftarrow B^{t_q} - B^{t_p}$ and $A_4 \leftarrow (1 - v_A)B^{t_p} + v_A B^{t_q}$.

We should have $A_3 = B_2 B^{t_{qA}} - B_1 B^{t_{pA}} = B^{t_{qB}} B^{t_{qA}} - B^{t_{pB}} B^{t_{pA}}) = B^{t_q} - B^{t_p}$ and $A_4 = B_1(1 - v_A)B^{t_{pA}} + B_2 v_A B^{t_{qA}} = B^{t_{pB}}(1 - v_A)B^{t_{pA}} + B^{t_{qB}} v_A B^{t_{qA}} = (1 - v_A)B^{t_p} + v_A B^{t_q}$, hence this game is statistically indistinguishable from the previous one, because of $D_B \overset{\$}{\leftarrow} \mathbb{Z}_{N^2/2}$ instead of $\mathbb{Z}_{N\Lambda}$.

**Game $\mathbf{G}_8$:** We remplace the ciphertexts $E_3, E_4$ and $E_6$ with random contents.

**Input and Setup.** As in the previous game.

**Switches.** For each $\mathsf{ESP}_\times^+(C, \bar{C})$, $\mathscr{S}im$ does as above until the generation of $E_3, E_4$ and $E_6$: $E_3, E_4, E_6 \leftarrow \mathbb{Z}_{N^2}^*$.

Under the $\mathsf{IND\text{-}CPA}$ security of the Paillier encryption scheme modulo $N^2$ (thanks to the $\mathsf{DCR}$ assumption over $\mathbb{Z}_N^*$), this game is indistinguishable from the previous one. One can note that $D_B$ follows a distribution that is statistically close to the previous one. Note that we do not need to generate $A_3$ and $A_4$, and thus we do not need to know $t_p$ nor $t_q$ anymore.

**Game $\mathbf{G}_9$:** We simulate the public parameters in a random way.

**Input.** $\mathscr{S}im$ still knows the factorization of $n$, but not of $N$.

**Setup.** $\mathscr{S}im$ randomly chooses $\chi \leftarrow \mathbb{Z}_n^* \setminus \mathbb{J}_n$, and this virtually defines $p$ such that $\chi$ is a square modulo $p$. $\mathscr{S}im$ picks at random $g_0 \overset{\$}{\leftarrow} \mathbb{Z}_n^*$ to set $g \leftarrow -g_0^2 \bmod n$, a generator of $\mathbb{J}_n$. It also picks $g_1 \in \mathbb{J}_n$.

$\mathscr{S}im$ picks at random $h_0 \overset{\$}{\leftarrow} \mathbb{Z}_n^*$ to set $h \leftarrow -h_0^2 \bmod n$, another generator of $\mathbb{J}_n$, as well as $v_p, v_q \overset{\$}{\leftarrow} \mathbb{Z}_n^*$ to set $h_p \leftarrow v_p^2$ and $h_q \leftarrow -v_q^2$.

Most of the secret keys are not known anymore, but no needed anyway, so Bob's secret key is generated randomly for $(t_{pB}, t_{qB}, s_B, D_B) \overset{\$}{\leftarrow} \mathbb{Z}_\lambda^3 \times \mathbb{Z}_{N\Lambda}$.

The rest of the game is unchanged.

This corresponds to choose $u_p, u_q \overset{\$}{\leftarrow} \mathbb{Z}_n^*$, to set $g_p \leftarrow u_p^2 \in \mathsf{QR}_n$ and $g_q \leftarrow -u_q^2 \in \mathbb{J}_n \setminus \mathsf{QR}_n$, and $\chi \leftarrow (1 - v)g_p + vg_q$ as before, but without knowing them nor the secret keys, but just the partial secret keys. Lemma 19 shows that this game is indistinguishable from the previous one under the $\mathsf{DDH}$ assumption in $\mathsf{QR}_n$.

**Game $\mathbf{G}_{10}$:** We can now choose random partial secret keys.

**Input.** $\mathscr{S}im$ does not need to know the factorization of $n$, nor of $N$.

**Setup.** $\mathscr{S}im$ randomly chooses $\chi \leftarrow \mathbb{Z}_n^* \setminus \mathbb{J}_n$, and this virtually defines $p$ such that $\chi$ is a square modulo $p$. $\mathscr{S}im$ chooses $(d_B, v_B, t_{pB}, t_{qB}, s_B, D_B) \overset{\$}{\leftarrow} \mathbb{Z}_{n^2/2} \times \mathbb{Z}_n \times \mathbb{Z}_{n/2}^3 \times \mathbb{Z}_{N^2/2}$. $\mathscr{S}im$ picks at random $g_0 \overset{\$}{\leftarrow} \mathbb{Z}_n^*$ to set $g \leftarrow -g_0^2 \bmod n$, a generator of $\mathbb{J}_n$. It also picks $g_1 \in \mathbb{J}_n$.

$\mathscr{S}im$ picks at random $h_0 \overset{\$}{\leftarrow} \mathbb{Z}_n^*$ to set $h \leftarrow -h_0^2 \bmod n$, another generator of $\mathbb{J}_n$, as well as $v_p, v_q \overset{\$}{\leftarrow} \mathbb{Z}_n^*$ to set $h_p \leftarrow v_p^2$ and $h_q \leftarrow -v_q^2$.

The rest of the game is unchanged.

Since the distributions of the partial keys are statistically close to the original ones, this game is statistically indistinguishable from the previous one.

**Game $\mathbf{G}_{11}$:** We can now choose random public parameters in $\mathbb{J}_n$.

**Input.** $\mathscr{S}im$ receives two strong RSA moduli $n$ and $N$ with $N > 2n^2$.

**Setup.** $\mathscr{S}im$ randomly chooses $\chi \leftarrow \mathbb{Z}_n^* \setminus \mathbb{J}_n$, as well as $g, g_1 \overset{\$}{\leftarrow} \mathbb{J}_n$. It then picks at random $h, h_p, h_q \overset{\$}{\leftarrow} \mathbb{J}_n$.

The rest of the game is unchanged.

Under the $\mathsf{QR}$ assumption in $\mathbb{Z}_n^*$, this game is indistinguishable from the previous one.

**Game $\mathbf{G}_{12}$:** Without the secret keys, we can replace the ciphertexts by random ciphertexts in the additive to multiplicative direction.

**Input and Setup.** As in the previous game.

**Switches.** For each $\mathsf{ESP}_+^\times(C, \bar{C})$, $\mathscr{S}im$ picks $C_A \overset{\$}{\leftarrow} \mathbb{Z}_{n^2}^*$ and plays as before.

Under the $\mathsf{IND}\text{-}\mathsf{CPA}$ security of the Paillier encryption scheme modulo $n$ for $C_A$ (thanks to the $\mathsf{DCR}$ assumption over $\mathbb{Z}_n^*$), this game is indistinguishable from the previous one.

**Game $\mathbf{G}_{13}$:** Without the secret keys, we can replace the ciphertexts by random ciphertexts in the multiplicative to additive direction.

**Input and Setup.** As in the previous game.

**Switches.** For each $\mathsf{ESP}_\times^+(C, \bar{C})$, $\mathscr{S}im$ picks $C_A \overset{\$}{\leftarrow} \mathbb{J}_n^3$ and plays as before.

Under the $\mathsf{IND}\text{-}\mathsf{CPA}$ security of the $\mathbb{Z}_n^*$-$\mathsf{EG}$ encryption scheme for $C_A$ (thanks to the $\mathsf{DDH}$ assumption over $\mathbb{J}_n$), this game is indistinguishable from the previous one.

**Lemma 19.** *For a strong RSA modulus $n$, with known factorization $n = qp$, choosing $g_0, h_0 \overset{\$}{\leftarrow} \mathbb{Z}_n^*$ to set $g \leftarrow -g_0^2$ as well as $h \leftarrow -h_0^2 \bmod n$, and two random scalars $t_p, t_q \overset{\$}{\leftarrow} \mathbb{Z}_\lambda$, respectively even and odd to set $g_p \leftarrow g^{t_p}$, $h_p \leftarrow h^{t_p}$, $g_q \leftarrow g^{t_q}$, $h_q \leftarrow h^{t_q}$ is indistinguishable from choosing $G_0, H_0, U_p, V_p, U_q, V_q \overset{\$}{\leftarrow} \mathsf{QR}_n$ and setting $g \leftarrow -G_0$, $h \leftarrow -H_0$, as well as $g_p \leftarrow U_p$, $h_p \leftarrow V_p$, and $g_q \leftarrow -U_q$, $h_q \leftarrow -V_q$.*

**Game $\mathbf{G}_0$:** In the real setup, we are given a strong RSA modulus $n$, and we choose $g_0, h_0 \overset{\$}{\leftarrow} \mathbb{Z}_n^*$ to set $g \leftarrow -g_0^2$ as well as $h \leftarrow -h_0^2 \bmod n$. We choose two random scalars $a_p, a_q \overset{\$}{\leftarrow} \mathbb{Z}_{\lambda/2}$, to define $t_p \leftarrow 2a_p$ and $t_q \leftarrow 2a_q + 1$: $g_p \leftarrow g^{2a_p} = (g_0^2)^{2a_p}$, $h_p \leftarrow h^{2a_p} = (h_0^2)^{2a_p}$, $g_q \leftarrow g^{2a_q+1} = -(g_0^2)^{2a_q+1}$ and $h_q \leftarrow h^{2a_q+1} = -(h_0^2)^{2a_q+1}$.

**Game $\mathbf{G}_1$:** We still choose $g_0, h_0 \overset{\$}{\leftarrow} \mathbb{Z}_n^*$ to set $g \leftarrow -g_0^2$ as well as $h \leftarrow -h_0^2 \bmod n$. But we choose two random scalars $a_p, a_q \overset{\$}{\leftarrow} \mathbb{Z}_{\lambda/2}$, to define $g_p \leftarrow (g_0^2)^{a_p}$, $h_p \leftarrow (h_0^2)^{a_p}$, $g_q \leftarrow -(g_0^2)^{a_q}$ and $h_q \leftarrow -(h_0^2)^{a_q}$. Since $\lambda/2 = p'q'$ is odd, the distribution of $2a_p$ modulo $\lambda/2$ is uniform in $\mathbb{Z}_{\lambda/2}$, and this is the same for $2b_q + 1$.

**Game $\mathbf{G}_2$:** We simulate the public parameters with a multi-Diffie-Hellman tuple $(A, B, C, D, C', D')$ in the squares $\mathsf{QR}_n$: $(g, h, g_p, h_p, g_q, h_q) \leftarrow (-A, -B, C, D, -C', -D')$. With $a_p = \log_A C = \log_B D$ and $a_q = \log_A C' = \log_B D'$, we have exactly the same setting as in the previous game.

**Game $\mathbf{G}_3$:** We simulate the multi-Diffie-Hellman tuple $(A, B, C, D, C', D')$ in $\mathsf{QR}_n$ from a single Diffie-Hellman tuple $(A, B, C, D)$ in $\mathsf{QR}_n$: we choose random scalars $u, v \overset{\$}{\leftarrow} \mathbb{Z}_{\lambda/2}$ and set $C' \leftarrow A^u C^v$ and $D' \leftarrow B^u D^v$. We have exactly the same setting as in the previous game, since with $a_p \leftarrow \log_A C = \log_B D$, we have and $C' = A^{u + a_p v}$ and $D' = B^{u + a_p v}$, hence $a_q = u + a_p v$.

**Game $\mathbf{G}_4$:** We do as above, but with a random $D \xleftarrow{\$} \mathsf{QR}_n$. This game is computationally indistinguishable from the previous one, under the DDH assumption in $\mathsf{QR}_n$.

**Game $\mathbf{G}_5$:** We do as above with random tuple $(A, B, C, D, C', D')$ in $\mathsf{QR}_n$. Since in the previous game we had $a_p = \log_A C \neq b_p = \log_B D$, we have and $C' = A^{u+a_p v}$ and $D' = B^{u+a_q v}$: $b_p \leftarrow u + a_p + v$ and $b_q \leftarrow u + a_q v$ are independent random scalars. Hence, this game provides exactly the same setting as the previous game.

*$\mathbb{Z}_n^*$−ESP is Zero-Knowledge for Bob.* As above, we first exhibit the simulator, and then prove that the behavior of the simulator is indistinguishable from Bob's behavior.

$\mathscr{S}im$ first receives as input two strong RSA moduli $n$ and $N$ with $N > (2 + 2^{\kappa+1})n^2$. It picks $g, g_1 \xleftarrow{\$} \mathbb{J}_n$ and $\chi \xleftarrow{\$} \mathbb{Z}_n^* \setminus \mathbb{J}_n$. It also picks $h, h_p, h_q \xleftarrow{\$} \mathbb{J}_n$. It then sets $\mathsf{pk} \leftarrow (n, g, \chi, g_1, N, h, h_p, h_q)$. It picks $(d_B, v_B, t_{pB}, t_{qB}, s_B, D_B) \xleftarrow{\$} \mathbb{Z}_{n^2/2} \times \mathbb{Z}_n \times \mathbb{Z}_{n/2}^3 \times \mathbb{Z}_{N^2/2}$ and sends them to the adversarial Alice, together with $\mathsf{pk}$.

Each time $\mathscr{S}im$ is asked to participate to an instance of a $\mathbb{Z}_n^*$−ESP on an input $C$, it additionally receives a target ciphertext $\bar{C}$, and the expected output is $\bar{C}$ or an equivalent ciphertext:

- If the direction is from additive to multiplicative, $\mathscr{S}im$ does not interact at all with Alice.
- If the direction is from multiplicative to additive,
  - $\mathscr{S}im$ picks and sends $B, B_1, B_2 \xleftarrow{\$} \mathbb{J}_n$;
  - $\mathscr{S}im$ sends a random $E_5 \xleftarrow{\$} \mathbb{Z}_{N^2}^*$.

Eventually, $\mathscr{S}im$ sends $\bar{C}$ to Alice as the output of the protocol.

**Sketch of the Proof.** The proof of Bob's simulation is similar to Alice's proof on several aspects. We provide here an intuition of the main differences. First, unlike for Alice, Bob's simulator $\mathscr{S}im$ cannot force the value $x = R_A \cdot m$, and in particular, the first element $\alpha$ of its decomposition, in the multiplicative to additive direction. But as the construction of $x/\beta$ from $\alpha$ is somehow "linear in the exponent", Bob proceed as follows: he first hides $\alpha$ with a random $r_1$, to make the computations on $\alpha \cdot g^{r_1}$ instead of $\alpha$, and then Bob simply multiply the output by $B' = \chi^{-r_1}$. As for any value $B \in \mathbb{J}_n$, there is a $r_1$ such that $B = \alpha g^{r_1}$, $\mathscr{S}im$ can send a random $B \in \mathbb{J}_n$ instead of $\alpha \cdot g^r$, and thus does not have to control the value of $\alpha$. Then, an additional lemma is required to show that generating public values at random is indistinguishable from constructing them honestly, if the DDH assumption holds in $\mathsf{QR}_n$ (the lemma is not the same as in the simulation of Alice). The proof of the lemma relies in part on the fact that from a DDH challenge, any number of random-looking DDH challenge with the same answer (DDH tuple or random) than the original challenge can be generated.

**Game $\mathbf{G}_0$:** This is the real game.

**Input.** In this game, $\mathscr{S}im$ knows the factorization of both $n$ and $N$, with two generators $g, h$ of $\mathbb{J}_n$.

**Setup.** $\mathscr{S}im$ generates all the secret keys for both Alice and Bob in an honest way, gives his secret key to Alice.

**Switches.** Each time it is asked to participate to an instance of the switching protocol on an input $(C, \bar{C})$, $\mathscr{S}im$ plays honestly the real game with the secret key of Bob.

**Game $\mathbf{G}_1$:** We first address the **additive to multiplicative direction**, with the final flow.

**Input and Setup.** As in the real game.

**Switches.** For each $\mathsf{ESP}_{+}^{\times}(C, \bar{C})$, in the last flow, $\mathscr{S}im$ sends $C' \leftarrow \bar{C}$.

When $(C, \bar{C})$ are twin ciphertexts, with a random $\bar{C}$, this game is perfectly indistinguishable from the previous one, since $\bar{C}$ contains the same plaintext as $C$.

**Game $\mathbf{G}_2$:** We now address the **multiplicative to additive direction**, with the final flow.

**Input and Setup.** As in the real game.

**Switches.** For each $\mathsf{ESP}_{\times}^{+}(C, \bar{C})$, in the last flow, $\mathscr{S}im$ sends $C' \leftarrow \bar{C}$.

When $(C, \bar{C})$ are twin ciphertexts, with a random $\bar{C}$, as above, this game is perfectly indistinguishable from the previous one, since $\bar{C}$ contains the same plaintext as $C$.

**Game $\mathbf{G}_3$:** We continue with the Paillier ciphertext $E_5$.

**Input and Setup.** As in the real game.

**Switches.** For each $\mathsf{ESP}^{+}_{\times}(C, \bar{C})$, when $\mathscr{S}im$ receives $C_A$, it computes $A \leftarrow (1-v)\alpha^{tp} + v\alpha^{tq}$, and generates a fresh Paillier ciphertext $E_5$ of $A$.

Since $E_5$ should contain the encryption of $A' = (v_B A_3 + A_4)B' = ((1-v)B^{t_p} + vB^{t_q})\chi^{-r_1}$, where we also know that $\chi^{-r_1} = (1-v)B_0^{-t_p} + vB_0^{-t_q}$, where $B_0 = g^{r_1}$, and so $\alpha = B/B_0$, we indeed have $A' = (1-v)(B/B_0)^{t_p} + v(B/B_0)^{t_q} = (1-v)\alpha^{t_p} + v\alpha^{t_q} = A$.

**Game $\mathbf{G}_4$:** We replace $E_5$ by a random ciphertext.

**Input.** As above, excepted that $\mathscr{S}im$ does not know anymore the factorization of $N$.

**Setup.** As above, excepted $D_A \xleftarrow{\$} \mathbb{Z}_{N^2/2}$.

**Switches.** For each $\mathsf{ESP}^{+}_{\times}(C, \bar{C})$, when $\mathscr{S}im$ receives $C_A$, it replaces $E_5$ by a random ciphertext: $E_5 \leftarrow \mathbb{Z}^*_{N^2}$.

Under the $\mathsf{IND\text{-}CPA}$ security of the Paillier encryption scheme modulo $N^2$ (thanks to the $\mathsf{DCR}$ assumption over $\mathbb{Z}_N$), this game is indistinguishable from the previous one. One can note that $D_A$ follows a distribution that is statistically close to the previous one. We note that we do not need to know $r_1$ anymore, nor $B'$.

**Game $\mathbf{G}_5$:** We replace $B$ by a random element.

**Input and Setup.** As in the previous game, with the explicit values $g_p \leftarrow g^{t_p}$ and $g_q \leftarrow g^{t_q}$.

**Switches.** For each $\mathsf{ESP}^{+}_{\times}(C, \bar{C})$, when $\mathscr{S}im$ receives $C_A$, it chooses a random $B \leftarrow g^a$ for $a \xleftarrow{\$} \mathbb{Z}_{n/2}$, and compute $B_1 \leftarrow g_p^a / B^{t_{pA}}$ and $B_2 \leftarrow g_q^a / B^{t_{qA}}$.

This corresponds to take $g^{r_1} = g^a/\alpha$, which is statistically indistinguishable from the previous game. In addition, $B_1$ should be $B^{t_{pB}} = B^{t_p - t_{pA}} = B^{t_p}/B^{t_{pA}} = (g^a)^{t_p}/B^{t_{pA}} = (g^{t_p})^a/B^{t_{pA}} = g_p^a/B^{t_{pA}}$, and the same for $B_2$.

**Game $\mathbf{G}_6$:** We simulate the public parameters in a random way.

**Input.** $\mathscr{S}im$ still knows the factorization of $n$, but not of $N$.

**Setup.** $\mathscr{S}im$ randomly chooses $\chi \leftarrow \mathbb{Z}^*_n \setminus \mathbb{J}_n$, and this virtually defines $p$ such that $\chi$ is a square modulo $p$. $\mathscr{S}im$ picks at random $g_0 \xleftarrow{\$} \mathbb{Z}^*_n$ to set $g \leftarrow -g_0^2 \bmod n$, a generator of $\mathbb{J}_n$. It also picks $g_1 \in \mathbb{J}_n$.

$\mathscr{S}im$ picks at random $h_0 \xleftarrow{\$} \mathbb{Z}^*_n$ to set $h \leftarrow -h_0^2 \bmod n$, another generator of $\mathbb{J}_n$, as well as $v_p, v_q \xleftarrow{\$} \mathbb{Z}^*_n$ to set $h_p \leftarrow v_p^2$ and $h_q \leftarrow -v_q^2$.

Most of the secret keys are not known anymore, but no needed anyway, so Alice's secret key is generated randomly for $(t_{pA}, t_{qA}, s_A, D_A) \xleftarrow{\$} \times\mathbb{Z}^3_\lambda \times \mathbb{Z}_{N\lambda}$.

The rest of the game is unchanged.

This corresponds to choose $u_p, u_q \xleftarrow{\$} \mathbb{Z}^*_n$, to set $g_p \leftarrow u_p^2 \in \mathsf{QR}_n$ and $g_q \leftarrow -u_q^2 \in \mathbb{J}_n \setminus \mathsf{QR}_n$, and $\chi \leftarrow (1-v)g_p + vg_q$ as before, but without knowing them nor the secret keys, but just the partial secret keys. Lemma 19 (from the previous simulation) shows that this game is indistinguishable from the previous one under the $\mathsf{DDH}$ assumption in $\mathsf{QR}_n$.

**Game $\mathbf{G}_7$:** We continue with a random simulation of $B_1$ and $B_2$.

**Input and Setup.** As in the previous game.

**Switches.** For each $\mathsf{ESP}^{+}_{\times}(C, \bar{C})$, when $\mathscr{S}im$ receives $C_A$, it chooses a random bit $b \xleftarrow{\$} \{0,1\}$, and three random squares $W, U', V' \xleftarrow{\$} \mathsf{QR}_n$, and sets $B \leftarrow (-1)^b W$, $B_1 \leftarrow U'/B^{t_{pA}}$, and $B_2 \leftarrow (-1)^b V'/B^{t_{pA}}$.

Lemma 20 shows that this game is indistinguishable from the previous one under the $\mathsf{DDH}$ assumption in $\mathsf{QR}_n$. The elements $g_p$ and $g_q$ are not needed anymore by the simulator.

**Game $\mathbf{G}_8$:** We can now choose random partial secret keys.

**Input.** $\mathscr{S}im$ does not need to know the factorization of $n$, nor of $N$.

**Setup.** $\mathscr{S}im$ randomly chooses $\chi \leftarrow \mathbb{Z}^*_n \setminus \mathbb{J}_n$, and this virtually defines $p$ such that $\chi$ is a square modulo $p$. $\mathscr{S}im$ chooses $(d_A, v_A, t_{pA}, t_{qA}, s_A, D_A) \xleftarrow{\$} \mathbb{Z}_{n^2/2} \times \mathbb{Z}_n \times \mathbb{Z}^3_{n/2} \times \mathbb{Z}_{N^2/2}$. $\mathscr{S}im$ picks at random $g_0 \xleftarrow{\$} \mathbb{Z}^*_n$ to set $g \leftarrow -g_0^2 \bmod n$, a generator of $\mathbb{J}_n$. It also picks $g_1 \in \mathbb{J}_n$.

$\mathscr{S}im$ picks at random $h_0 \xleftarrow{\$} \mathbb{Z}^*_n$ to set $h \leftarrow -h_0^2 \bmod n$, another generator of $\mathbb{J}_n$, as well as $v_p, v_q \xleftarrow{\$} \mathbb{Z}^*_n$ to set $h_p \leftarrow v_p^2$ and $h_q \leftarrow -v_q^2$.

The rest of the game is unchanged.

Since the distributions of the partial keys are statistically close to the original ones, this game is statistically indistinguishable from the previous one.

**Game $\mathbf{G}_9$:** We can now choose random public parameters in $\mathbb{J}_n$.

**Input.** $\mathscr{S}im$ receives two strong RSA moduli $n$ and $N$ with $N > 2n^2$.

**Setup.** $\mathscr{S}im$ randomly chooses $\chi \leftarrow \mathbb{Z}_n^* \setminus \mathbb{J}_n$, as well as $g, g_1 \overset{\$}{\leftarrow} \mathbb{J}_n$. It then picks at random $h, h_p, h_q \overset{\$}{\leftarrow} \mathbb{J}_n$.

The rest of the game is unchanged.

Under the $\mathsf{QR}$ assumption in $\mathbb{Z}_n^*$, this game is indistinguishable from the previous one.

**Game $\mathbf{G}_{10}$:** We can now randomly choose the $B_i$'s in $\mathbb{J}_n$.

**Input and Setup.** As in the previous game.

**Switches.** For each $\mathsf{ESP}^+_\times(C, \bar{C})$, when $\mathscr{S}im$ receives $C_A$, it randomly picks $B, B_1, B_2 \overset{\$}{\leftarrow} \mathbb{J}_n$.

This does not change anything for $B$, but for $B_1$ and $B_2$, this corresponds to randomly take $U', V' \overset{\$}{\leftarrow} \mathbb{J}_n$ instead of $\mathsf{QR}_n$, which is indistinguishable under the $\mathsf{QR}$ assumption in $\mathbb{Z}_n^*$.

**Lemma 20.** *For a generator $g$ of $\mathbb{J}_n$, and $g_p = g^{t_p}$ (for even $t_p$), $g_q = g^{t_q}$ (for odd $t_q$), choosing multiple $B_i \leftarrow g^{a_i}$ for $a_i \overset{\$}{\leftarrow} \mathbb{Z}_\lambda$, and setting $U_i \leftarrow g_p^{a_i}$, and $V_i \leftarrow g_q^{a_i}$, or choosing $b_i \overset{\$}{\leftarrow} \{0,1\}$, $W_i, U'_i, V'_i \overset{\$}{\leftarrow} \mathsf{QR}_n$, and setting $B_i \leftarrow (-1)^{b_i} W_i$, $U_i \leftarrow U'_i$, and $V_i \leftarrow (-1)^{b_i} V'_i$ are indistinguishable.*

**Game $\mathbf{G}_0$:** We simulate $(g, g_p, g_q, \{B_i, U_i, V_i\}_i)$ with a random Diffie-Hellman tuple $(A, B, C, \{W_i, S_i, T_i\})$ in $\mathsf{QR}_n$, where for all $i$, $\gamma_i = \log_A W_i = \log_B S_i = \log_C T_i \in \mathbb{Z}_{\lambda/2}$. We can indeed set $(g, g_p, g_q) \leftarrow (-A, B, -C)$, and for all $i$, $(B_i, U_i, V_i) \leftarrow ((-1)^{b_i} W_i, S_i, (-1)^{b_i} T_i)$. Since the group $\mathsf{QR}_n$ of the squares in $\mathbb{Z}_n^*$ is of odd order $\delta \leftarrow \lambda/2 = p'q'$, with overwhelming probability, $A$ is a generator, and so we can denote $u_p \leftarrow \log_A B$ and $u_q \leftarrow \log_A C$. Then $g_p = B = A^{u_p} = (-A)^{\delta u_p}(-1)^{u_p} A^{u_p} = (-A)^{(\delta+1)u_p} = g^{t_p}$, with $t_p \leftarrow (\delta+1)u_p$ (that is even), and $g_q = -C = -A^{u_q} = (-A)^\delta (-A)^{(\delta+1)u_q} = g^{t_q}$, as just before, with $t_q \leftarrow \delta + (\delta+1)u_q$ (that is odd).

Then $B_i = (-1)^{b_i} A^{\gamma_i} = (-A)^{\delta b_i} (-A)^{\delta \gamma_i} (-A)^{\gamma_i} = g^{\gamma_i + \delta(b_i + \gamma_i)} = g^{a_i}$, for $a_i = \gamma_i + \delta(b_i + \gamma_i) \bmod \lambda$. But in addition, $U_i = S_i = B^{\gamma_i} = g_p^{a_i}$ since $g_p = B$ is a square, of order $\delta$, and $V_i = (-1)^{b_i} T_i = (-1)^{b_i} C^{\gamma_i} = g_q^{a_i}$, as above for $B_i$.

**Game $\mathbf{G}_1$:** We do as above, but generating the multi-Diffie-Hellman tuple $(A, B, C, \{W_i, S_i, T_i\})$ from a simpler one $(A, B, \{W_i, S_i\}_i)$ over $\mathsf{QR}_n$, by choosing a random $u_q \overset{\$}{\leftarrow} \mathbb{Z}_{\lambda/2}$, and setting $C \leftarrow A^{u_q}$, and $T_i \leftarrow W_i^{u_q}$ for all $i$. This game is perfectly indistinguishable from the previous one.

**Game $\mathbf{G}_2$:** We do as above, but with a unique random Diffie-Hellman tuple $(A, B, W, S)$, by choosing random $u_i, v_i \overset{\$}{\leftarrow} \mathbb{Z}_{\lambda/2}$ to set $W_i \leftarrow A^{u_i} W^{v_i}$ and $S_i \leftarrow B^{u_i} S^{v_i}$. This game is perfectly indistinguishable from the previous one.

**Game $\mathbf{G}_3$:** We do as above, but with a random $S$. This game is computationally indistinguishable from the previous one, under the $\mathsf{DDH}$ assumption in $\mathsf{QR}_n$.

**Game $\mathbf{G}_4$:** We do as above, but generating the tuple $(A, B, C, \{W_i, S_i, T_i\})$ from a simpler one $(A, C, \{W_i, T_i\}_i)$ over $\mathsf{QR}_n$, by choosing a random $B \overset{\$}{\leftarrow} \mathsf{QR}_n$ and random $S_i \overset{\$}{\leftarrow} \mathsf{QR}_n$ for all $i$. Since $B$ and the $S_i$ were random and independent in the previous game, this game remains perfectly indistinguishable.

**Game $\mathbf{G}_5$:** We do as above, but with a unique random Diffie-Hellman tuple $(A, C, W, T)$, by choosing random $u_i, v_i \overset{\$}{\leftarrow} \mathbb{Z}_{\lambda/2}$ to set $W_i \leftarrow A^{u_i} W^{v_i}$ and $T_i \leftarrow C^{u_i} T^{v_i}$. This game is perfectly indistinguishable from the previous one.

**Game $\mathbf{G}_6$:** We do as above, but with a random $T$. This game is computationally indistinguishable from the previous one, under the $\mathsf{DDH}$ assumption in $\mathsf{QR}_n$.

**Game $\mathbf{G}_7$:** We do as above, but generating the tuple $(A, B, C, \{W_i, S_i, T_i\})$ totally at random, which is exactly the second distribution of the setting, which concludes the proof of the lemma.

### C.2 Encryption Switching Protocols on $\mathbb{Z}_n$

Let us recall Theorem 11:

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

**Setup and Key Generation**

- The main strong RSA modulus $n$:
  - $p, q$ two safe primes, $n \leftarrow pq$;
  - $g_0 \stackrel{\$}{\leftarrow} \mathbb{Z}_n^*$, $g \leftarrow -g_0^2$ (a generator of $\mathbb{J}_n$, of order $\lambda$);
  - $d \leftarrow [\lambda^{-1} \mod n] \cdot \lambda \bmod n\lambda$: $d = 0 \bmod \lambda$ and $d = 1 \bmod n$;
  - $v \leftarrow [p^{-1} \bmod q] \cdot p \bmod n$: $v = 0 \bmod p$ and $v = 1 \bmod q$;
  - an even $t_p \stackrel{\$}{\leftarrow} \mathbb{Z}_\lambda$ and an odd $t_q \stackrel{\$}{\leftarrow} \mathbb{Z}_\lambda$: $\chi \leftarrow (1-v) \cdot g^{t_p} + v \cdot g^{t_q} \bmod n$;
  - $s \stackrel{\$}{\leftarrow} \mathbb{Z}_\lambda$, and set $g_1 \leftarrow g^s \bmod n$ (for $\mathbb{J}_n$-$\mathsf{EG}$).
  - $s_2, s_3 \stackrel{\$}{\leftarrow} \mathbb{Z}_{\lambda/2}^2$, and set $g_2 \leftarrow g^{2s_2} \bmod n$ (for $\mathsf{QR}_n$-$\mathsf{EG}$) and $g_3 \leftarrow g^{2s_3} \bmod n$ (for $\mathsf{QR}_n$-$\mathsf{EG}'$).
- 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 $\Lambda$ 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: $(d_A, v_A, t_{pA}, t_{qA}, s_A, s_{2A}, D_A) \stackrel{\$}{\leftarrow} \mathbb{Z}_{n\lambda} \times \mathbb{Z}_n \times \mathbb{Z}_\lambda^3 \times \mathbb{Z}_{\lambda/2} \times \mathbb{Z}_{N\Lambda}$, and $d_B \leftarrow d - d_A \bmod n\lambda$, $v_B \leftarrow v - v_A \bmod n$, $t_{pB} \leftarrow t_p - t_{pA} \bmod \lambda$, $t_{qB} \leftarrow t_q - t_{qA} \bmod \lambda$, $s_B \leftarrow s - s_A \bmod \lambda$, $s_{2B} \leftarrow s_2 - s_{2A} \bmod \lambda/2$, and $D_B \leftarrow D - D_A \bmod N\Lambda$.

**$\mathscr{E}_\otimes^0(\cdot) = \mathbb{Z}_n$-$\mathsf{EG}$: 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 $T, T' \stackrel{\$}{\leftarrow} \mathsf{QR}_n$ and compute $C_1 \leftarrow \mathscr{E}_\otimes(m + b)$, $C_2 \leftarrow \mathsf{QR}_n$-$\mathsf{EG}.\mathsf{Enc}(T^b)$, $C_3 \leftarrow \mathsf{QR}_n$-$\mathsf{EG}'.\mathsf{Enc}(T'^b)$.
Return the ciphertext $C = \mathscr{E}_\otimes^0(m) = (C_1, C_2, C_3)$.

$\mathsf{Rand}(\mathsf{pk}, C = (C_1, C_2, C_3))$ : Choose random $r_2, r_3 \stackrel{\$}{\leftarrow} \mathbb{Z}_{n/4}$, and compute $C_1' \leftarrow \mathbb{Z}_n^*$-$\mathsf{EG}.\mathsf{Rand}(C_1)$, $C_2' \leftarrow \mathsf{QR}_n$-$\mathsf{EG}.\mathsf{Rand}(C_2^{r_2})$, and $C_3' \leftarrow \mathsf{QR}_n$-$\mathsf{EG}'.\mathsf{Rand}(C_3^{r_3})$. 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 $T'' \leftarrow \mathsf{QR}_n$-$\mathsf{EG}.\mathsf{Dec}(C_2)$. If $T'' = \perp$, return $\perp$; if $T'' = 1$, return 0; otherwise compute $m \leftarrow \mathscr{D}_\otimes(C_1)$ and return $m$.

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

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

$\mathsf{Rand}(\mathsf{pk}, c)$ : choose $r \stackrel{\$}{\leftarrow} \mathbb{Z}_n^*$, 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. 3: Setup and Encryption Schemes in $\mathbb{Z}_n$ (repeated from page 14)

**Theorem 21.** *$\mathbb{Z}_n - \mathsf{ESP}_\times^+$ is computationally zero-knowledge.*

*$\mathbb{Z}_n-\mathsf{ESP}$ is Zero-Knowledge for Alice.* Let us first exhibit the simulator, we then prove that the behavior of the simulator is indistinguishable from Alice's behavior.

$\mathscr{S}im$ first receives as input two strong RSA moduli $n$ and $N$ with $N > (2 + 2^{\kappa+1})n^2$. It picks $g, g_1, g_2, g_3 \stackrel{\$}{\leftarrow} \mathbb{J}_n$ and $\chi \stackrel{\$}{\leftarrow} \mathbb{Z}_n^* \backslash \mathbb{J}_n$. It then sets $\mathsf{pk} \leftarrow (n, g, \chi, g_1, g_2, g_3, N)$. It also picks $(d_B, v_B, t_{pB}, t_{qB}, s_B, s_{2B}, D_B) \stackrel{\$}{\leftarrow} \mathbb{Z}_{n^2/2} \times \mathbb{Z}_n \times \mathbb{Z}_{n/2}^3 \times \mathbb{Z}_{n/4} \times \mathbb{Z}_{N^2/2}$ and sends them to the adversarial Bob, together with $\mathsf{pk}$.

Each time $\mathscr{S}im$ is asked to participate to an instance of a $\mathbb{Z}_n-\mathsf{ESP}$ on an input $C$, it additionally receives a target ciphertext $\bar{C}$, and the expected output is a randomization of $\bar{C}$:

- If the direction is from additive to multiplicative,
  - $\mathscr{S}im$ runs the simulation of $\mathsf{EZT}$ on $(C, C' \stackrel{\$}{\leftarrow} \mathbb{Z}_{n^2}^*)$;
  - $\mathscr{S}im$ randomly chooses $C_{1+} \stackrel{\$}{\leftarrow} \mathbb{Z}_{n^2}^*$, and sends it to Bob;
  - $\mathscr{S}im$ parses $\bar{C} = (\bar{C}_{1\times}, \bar{C}_{2\times}, \bar{C}_{3\times})$, and runs the simulation of the $\mathsf{ESP}$ protocol on the input $(C_{1+}, \bar{C}_{1\times})$;
  - $\mathscr{S}im$ randomly picks $k, R_2 \stackrel{\$}{\leftarrow} \mathbb{J}_n$ and generates $c \leftarrow \mathsf{Com}(k)$ and $C_2' \leftarrow R_2^{-1} \bullet \bar{C}_{2\times}$, as well as $D_2 \leftarrow (1 + R_2 \cdot n)/C_2^{d_B}$. It also randomly picks $C_2, C_3 \stackrel{\$}{\leftarrow} \mathbb{Z}_{n^2}^*$, $C_3', D_3 \stackrel{\$}{\leftarrow} \mathbb{J}_n$, and sends $(c, C_2, C_2', D_2, C_3, C_3', D_3)$ to Bob;

**2-Party $\mathsf{ESP}_{+}^{\times}$ from $C = \mathscr{E}_{\oplus}(m)$ into $(C_{1\times}, C_{2\times}, C_{3\times}) = \mathscr{E}_{\otimes}^{0}(m)$**

| | | |
|---|---|---|
| Alice gets $C_{\mathsf{EZT}} = \mathscr{E}_{\oplus}(b)$ | $\xleftrightarrow{C_{\mathsf{EZT}} \leftarrow \mathsf{EZT}(C)}$ | Bob gets nothing |
| $C_{1+} \leftarrow C \boxplus C_{\mathsf{EZT}}$ | $\xrightarrow{C_{1+}}$ | |
| Alice gets $C_{1\times}$ | $\xleftrightarrow{C_{1\times} \leftarrow \mathbb{Z}_n^*\text{-}\mathsf{ESP}(C_{1+})}$ | Bob gets $C_{1\times}$ |
| $T, T', R_2, R_3, k \xleftarrow{\$} \mathsf{QR}_n$ | | |
| $c \leftarrow \mathsf{com}(k)$ | | |
| $C_{2+} \leftarrow \mathscr{E}_{\oplus}(1) \boxplus (T-1) \bullet C_{\mathsf{EZT}}$ | | |
| $C_{3+} \leftarrow \mathscr{E}_{\oplus}(1) \boxplus (T'-1) \bullet C_{\mathsf{EZT}}$ | | |
| $C_2 \leftarrow \mathbb{Z}_n\text{-}\mathsf{P.Rand}(R_2 \bullet C_{2+})$ | | |
| $C_2' \leftarrow \mathsf{QR}_n\text{-}\mathsf{EG.Enc}(R_2^{-1})$ | | |
| $C_3 \leftarrow \mathbb{Z}_n\text{-}\mathsf{P.Rand}(k \cdot R_3 \bullet C_{3+})$ | | |
| $C_3' \leftarrow \mathsf{QR}_n\text{-}\mathsf{EG'.Enc}(R_3^{-1})$ | | |
| $D_2 \leftarrow C_2'^{d_A} \bmod n^2$ | | |
| $D_3 \leftarrow C_3'^{d_A} \bmod n^2$ | $\xrightarrow{c, C_2, C_2', D_2, C_3, C_3', D_3}$ | $x_2 \leftarrow ([C_2^{d_B} D_2 \bmod n^2] - 1)/n$ |
| | | $x_3 \leftarrow ([C_3^{d_B} D_3 \bmod n^2] - 1)/n$ |
| | | $C_{2\times} \leftarrow \mathsf{QR}_n\text{-}\mathsf{EG.Rand}(x_2 \bullet C_2')$ |
| $C_{3\times}'' \leftarrow k^{-1} \bullet C_{3\times}'$ | $\xleftarrow{C_{2\times}, C_{3\times}'}$ | $C_{3\times}' \leftarrow \mathsf{QR}_n\text{-}\mathsf{EG'.Rand}(x_3 \bullet C_3')$ |
| $k' \xleftarrow{\$} \mathbb{Z}_{n/4}$ | | |
| $C_{3\times} \leftarrow \mathsf{QR}_n\text{-}\mathsf{EG'.Rand}({C_{3\times}''}^{k'})$ | $\xrightarrow{C_{3\times}}$ | |

**2-Party $\mathsf{ESP}_{\times}^{+}$ from $(C_{1\times}, C_{2\times}, C_{3\times}) = \mathscr{E}_{\otimes}^{0}(m)$ into $C' = \mathscr{E}_{\oplus}(m)$**

| | | |
|---|---|---|
| Alice gets $C_{1+}$ | $\xleftrightarrow{C_{1+} \leftarrow \mathbb{Z}_n^*\text{-}\mathsf{ESP}(C_{1\times})}$ | Bob gets $C_{1+}$ |
| $R_2 \xleftarrow{\$} \mathsf{QR}_n,$ | | |
| $k' \xleftarrow{\$} \mathbb{Z}_{n/4}$ | | |
| $C_2 \leftarrow \mathsf{QR}_n\text{-}\mathsf{EG.Rand}(R_2 \bullet C_{2\times}^{k'})$ | | |
| $= (c_0, c_1)$ | | |
| $C_2' \leftarrow \mathscr{E}_{\oplus}(R_2^{-1})$ | | |
| $d_1 \leftarrow c_0^{s_{2A}}$ | $\xrightarrow{C_2, C_2', d_1}$ | $x_2 \leftarrow c_1/d_1 c_0^{s_{2B}}$ |
| | | $k \xleftarrow{\$} \mathbb{Z}_n^*$ |
| | $\xleftarrow{C_{2+}'}$ | $C_{2+}' \leftarrow k \bullet (x_2 \bullet C_2' \boxminus \mathscr{E}_{\oplus}(1))$ |
| Alice gets nothing | $\xleftrightarrow{C_{\mathsf{EZT}} \leftarrow \mathsf{EZT}(C_{2+}')}$ | Bob gets $C_{\mathsf{EZT}} = \mathscr{E}_{\oplus}(b')$ |
| | | $(\rho_0, \rho_1) \xleftarrow{\$} \mathbb{Z}_n^2$ |
| | | $B_1 \leftarrow C_{1+} \boxplus \mathscr{E}_{\oplus}(\rho_0)$ |
| | | $B_2 \leftarrow \mathscr{E}_{\oplus}(\rho_1) \boxplus C_{\mathsf{EZT}}$ |
| | | $B_3 \leftarrow \rho_0 \bullet C_{\mathsf{EZT}}$ |
| | | $B_4 \leftarrow \rho_1 \bullet C_{1+}$ |
| | | $B_5 \leftarrow \mathscr{E}_{\oplus}(\rho_0 \rho_1)$ |
| | | $B_6 \leftarrow B_3 \boxplus B_4 \boxplus B_5$ |
| | | $A_1' \leftarrow B_1^{d_B} \bmod n^2$ |
| $A_1 \leftarrow ([B_1^{d_A} A_1' \bmod n^2] - 1)/n$ | $\xleftarrow{B_1, B_2, A_1', A_2', B_6}$ | $A_2' \leftarrow B_2^{d_B} \bmod n^2$ |
| $A_2 \leftarrow ([B_2^{d_A} A_2' \bmod n^2] - 1)/n$ | | |
| $C' \leftarrow \mathscr{E}_{\oplus}(A_1 \cdot A_2) \boxminus B_6$ | $\xrightarrow{C'}$ | |

Fig. 5: Interactive Encryption Switching in $\mathbb{Z}_n$ (repeated from page 16)

  - $\mathscr{S}im$ eventually sends $\bar{C}_{3\times}$ to Bob;
- If the direction is from multiplicative to additive,
  - $\mathscr{S}im$ randomly picks $C_{1+} \stackrel{\$}{\leftarrow} \mathbb{Z}_{n^2}^*$, and runs the simulation of the $\mathsf{ESP}$ protocol on the inputs $(C_{1\times}, C_{1+})$;
  - $\mathscr{S}im$ randomly picks $C_2 \stackrel{\$}{\leftarrow} \mathbb{J}_n^2$, $C_2' \stackrel{\$}{\leftarrow} \mathbb{Z}_{n^2}^*$, $d_1 \stackrel{\$}{\leftarrow} \mathbb{J}_n$, and sends the tuple $(C_2, C_2', d_1)$ to Bob;
  - $\mathscr{S}im$ runs the simulation of $\mathsf{EZT}$ on $(C_{2+}', C' \stackrel{\$}{\leftarrow} \mathbb{Z}_{n^2}^*)$;
  - $\mathscr{S}im$ eventually sends $\bar{C}$.

With the following sequence of games, we show that the simulator $\mathscr{S}im$ has the same behavior as Alice, even for many instances. The simulators will be constructed step by step, starting from a simulator that knows all the secrets, and then behaves exactly as Alice.

**Game $\mathbf{G}_0$:** This is the real game.

**Input.** In this game, $\mathscr{S}im$ knows the factorization of both $n$ and $N$, with two generators $g, h$ of $\mathbb{J}_n$.

**Setup.** $\mathscr{S}im$ generates all the secret keys for both Alice and Bob in an honest way, gives his secret key to Bob.

**Switches.** Each time it is asked to participate to an instance of the switching protocol on an input $(C, \bar{C})$, $\mathscr{S}im$ plays honestly the real game with the secret key of Alice.

**Game $\mathbf{G}_1$:** This simulator uses the trapdoor $s_3$.

**Input and Setup.** As in the previous game, excepted that $\mathscr{S}im$ keeps the secret $s_3$.

**Switches.** For each $\mathsf{ESP}_\times^+(C, \bar{C})$ or $\mathsf{ESP}_+^\times(C, \bar{C})$, either $C$ or $\bar{C}$ is $\mathscr{E}_\otimes^0(m)$, and then it uses $s_3$ to extract $b$ from the third part: if the plaintext is 1, then $b$ is set to 0, otherwise it is set it 1. Since no modification is done, this game is perfectly indistinguishable from the previous one.

**Game $\mathbf{G}_2$:** We start to modify the simulation in the **multiplicative to additive direction**, by altering $C_2$ and $C_2'$.

**Input and Setup.** As in the previous game.

**Switches.** For each $\mathsf{ESP}_\times^+(C, \bar{C})$, $\mathscr{S}im$ chooses two random squares $R, R' \stackrel{\$}{\leftarrow} \mathsf{QR}_n$, and then generates two fresh ciphertexts $C_2 \leftarrow \mathsf{QR}_n\text{-}\mathsf{EG.Enc}(R')$, and $C_2' \leftarrow \mathscr{E}_\oplus(R^b/R')$, with the extracted $b$. $C_2$ contains $R_2 \cdot R^b$, for $R = T^{k'}$, and $C_2'$ contains $R_2^{-1}$. Since $n$ is a strong RSA modulus, the order of $\mathsf{QR}_n$ is $\lambda/2 = p'q'$, and so any square is likely a generator. Hence a random $R$ follows a distribution that is statistically close to the distribution of $T^{k'}$. By setting $R_2 \leftarrow R'/R^b$, we are exactly as in the previous game: This game is statistically indistinguishable from the previous one.

**Game $\mathbf{G}_3$:** We now alter the decryption of $C_2$.

**Input and Setup.** As in the previous game.

**Switches.** For each $\mathsf{ESP}_\times^+(C, \bar{C})$, $\mathscr{S}im$ sends $d_1 \leftarrow c_1 c_0^{-s_{2B}}/R'$.

Since $C_2$ encrypts $R'$, this should be the value of $x_2 = c_1 c_0^{-s_{2B}}/d_1$: This game is perfectly indistinguishable from the previous one.

**Game $\mathbf{G}_4$:** We can now modify the simulation of the $\mathsf{EZT}$.

**Input and Setup.** As in the previous game.

**Switches.** For each $\mathsf{ESP}_\times^+(C, \bar{C})$, $\mathscr{S}im$ simulates the $\mathsf{EZT}$ protocol on the input $(C_{2+}', \mathscr{E}_\oplus(1-b))$, from the extracted $b$.

From the previous games, $C_{2+}'$ contains a Paillier encryption of $R^b - 1 \bmod n$: if $b = 0$, this is 0, if $b = 1$, this is non-zero. So the $\mathsf{EZT}$ output should be a ciphertext of $b' = 1-b$. Under the (statistical) zero-knowledge property of the $\mathsf{EZT}$ protocol, this game is (statistically) indistinguishable from the previous one.

**Game $\mathbf{G}_5$:** We end by modifying the final flow.

**Input and Setup.** As in the previous game.

**Switches.** For each $\mathsf{ESP}_\times^+(C, \bar{C})$, in the last flow, $\mathscr{S}im$ sends $C' \leftarrow \bar{C}$.

When $(C, \bar{C})$ are twin ciphertexts, this flow is indistinguishable from the previous game since $C'$ should look like a fresh encryption of $m$, because of the fresh encryption of $A_1 \cdot A_2$: This game is statistically indistinguishable from the previous one.

**Game $\mathbf{G}_6$:** We now modify the simulation in the **additive to multiplicative direction**.

**Input and Setup.** As in the previous game.

**Switches.** For each $\mathsf{ESP}_{+}^{\times}(C, \bar{C})$, $\mathscr{S}im$ chooses a random $k \xleftarrow{\$} \mathsf{QR}_n$, but sends a commitment $c$ on a random value.

Under the hiding property of the commitment scheme, this game is indistinguishable from the previous one. However, now, $C_3$ and $C'_3$ contains independent values.

**Game $\mathbf{G}_7$:** We now modify the simulation of the $\mathsf{EZT}$.

**Input and Setup.** As in the previous game.

**Switches.** For each $\mathsf{ESP}_{+}^{\times}(C, \bar{C})$, $\mathscr{S}im$ simulates the $\mathsf{EZT}$ protocol on the input $(C, \mathscr{E}_{\oplus}(b))$, from the extracted $b$.

When $(C, \bar{C})$ are twin ciphertexts, the third part of $\bar{C}$ contains a $\mathsf{QR}_n$-$\mathsf{EG}'$ encryption of $\bar{T'}^{b}$, where $b = 1$ if the message is 0, and $b = 0$ otherwise. This is the bit $b$ we've extracted above, and that should be encrypted in the result of the honest $\mathsf{EZT}$ protocol on $C$. Under the (statistical) zero-knowledge property of the $\mathsf{EZT}$ protocol, this game is (statistically) indistinguishable from the previous one.

**Game $\mathbf{G}_8$:** We now alter the ciphertexts $C_{1+}$, $C_{2+}$, and $C_{3+}$.

**Input and Setup.** As in the previous game.

**Switches.** For each $\mathsf{ESP}_{+}^{\times}(C, \bar{C})$, $\mathscr{S}im$ simulates $C_{1+} \leftarrow C \boxplus \mathscr{E}_{\oplus}(b)$, $C_{2+} \leftarrow \mathscr{E}_{\oplus}(T^b)$, and $C_{3+} \leftarrow \mathscr{E}_{\oplus}(T'^b)$.

Since $C_{\mathsf{EZT}} = \mathscr{E}_{\oplus}(b)$, this game is perfectly indistinguishable from the previous one.

**Game $\mathbf{G}_9$:** We continue by altering $C_2$, $C'_2$ and $C_3$, $C'_3$.

**Input and Setup.** As in the previous game.

**Switches.** For each $\mathsf{ESP}_{+}^{\times}(C, \bar{C})$, $\mathscr{S}im$ chooses two random squares $R_2, R_3 \xleftarrow{\$} \mathsf{QR}_n$, and then generates $C_2 \leftarrow \mathscr{E}_{\oplus}(R_2)$ and $C'_2 \leftarrow {R_2}^{-1} \bullet \bar{C}_{2\times}$, as well as $C_3 \leftarrow \mathscr{E}_{\oplus}(k \cdot R_3)$ and $C'_3 \leftarrow (k \cdot R_3)^{-1} \bullet \bar{C}_{3\times}$.

When $(C, \bar{C})$ are twin ciphertexts, $\bar{C}_{2\times}$ should contain a $\mathsf{QR}_n$-$\mathsf{EG}$ encryption of $\bar{T}^b$, and $\bar{C}_{3\times}$ should contain a $\mathsf{QR}_n$-$\mathsf{EG}'$ encryption of $\bar{T'}^{b}$, hence this game is perfectly indistinguishable from the previous one.

**Game $\mathbf{G}_{10}$:** We now alter the decryption of $C_2$ and $C_3$.

**Input and Setup.** As in the previous game.

**Switches.** For each $\mathsf{ESP}_{+}^{\times}(C, \bar{C})$, $\mathscr{S}im$ sends $D_2 \leftarrow (1 + R_2 n)/C_2^{d_B}$ and $D_3 \leftarrow (1 + k \cdot R_3 n)/C_3^{d_B}$.

Since $C_2$ encrypts $R_2$, this should be the value of $x_2 = R_2 = ([C_2^{d_B} D_2 \bmod n^2] - 1)/n$, and similarly for $C_3$, with $x_3 = k \cdot R_3$: This game is perfectly indistinguishable from the previous one.

**Game $\mathbf{G}_{11}$:** We then modify the final flow.

**Input and Setup.** As in the previous game.

**Switches.** For each $\mathsf{ESP}_{+}^{\times}(C, \bar{C})$, $\mathscr{S}im$ sends $C_{3\times} \leftarrow \bar{C}_3$.

When $(C, \bar{C})$ are twin ciphertexts, this flow is indistinguishable from the previous game since $C_{3\times}$ should look like a fresh encryption $T'^{b}$ for some random square $T'$, as explained in $\mathbf{G}_2$: This game is statistically indistinguishable from the previous one.

**Game $\mathbf{G}_{12}$:** We continue by simulating the $\mathbb{Z}_n^*-\mathsf{ESP}$ all in parallel in both directions.

**Input.** $\mathscr{S}im$ does not know the factorization of $n$, nor of $N$.

**Setup.** $\mathscr{S}im$ randomly generates the public keys without knowing the secret keys, excepted $s_3$.

**Switches.** For each instance of the switching protocol, we run the simulators for the $\mathbb{Z}_n^*-\mathsf{ESP}$:

- from additive to multiplicative, on an input $(C, \bar{C})$, where one can parse $\bar{C} = (\bar{C}_{1\times}, \bar{C}_{2\times}, \bar{C}_{3\times})$, $\mathscr{S}im$ simulates the $\mathbb{Z}_n^*-\mathsf{ESP}$ protocol on the inputs $(C_{1+}, \bar{C}_{1\times})$.

  When $(C, \bar{C})$ are twin ciphertexts, in $\bar{C} = (\bar{C}_{1\times}, \bar{C}_{2\times}, \bar{C}_{3\times})$, $\bar{C}_{1\times}$ should be $\mathscr{E}_{\otimes}(m + b)$.
- from multiplicative to additive, on an input $(C, \bar{C})$, $\mathscr{S}im$ simulates the $\mathbb{Z}_n^*-\mathsf{ESP}$ protocol on the inputs $(C_{1\times}, \bar{C} \boxplus \mathscr{E}_{\oplus}(b))$.

  When $(C, \bar{C})$ are twin ciphertexts, $\bar{C} = \mathscr{E}_{\oplus}(m)$, so $\bar{C} \boxplus \mathscr{E}_{\oplus}(b)$ contains $m + b$.

Under the zero-knowledge property of the $\mathsf{ESP}$ protocols, thanks to the $\mathsf{DDH}$ assumption in $\mathsf{QR}_n$, the $\mathsf{QR}$ assumption in $\mathbb{Z}_n^*$, the $\mathsf{DCR}$ assumption over $\mathbb{Z}_n^*$, and the $\mathsf{DCR}$ assumption over $\mathbb{Z}_N^*$, this game is indistinguishable from the previous one. This corresponds to inline the games from the proof of Theorem 10. One can note that our simulator does not need to know any secret excepted $s_3$.

**Game $\mathbf{G}_{13}$:** We now start to remove the need of $b$.

**Input and Setup.** As in the previous game (no more secret excepted $s_3$).

**Switches.** For each $\mathsf{EZT}$ protocol, a random ciphertext $C' \xleftarrow{\$} \mathbb{Z}_{n^2}^*$ is given as second input.

Under the $\mathsf{IND\text{-}CPA}$ security of the Paillier encryption scheme modulo $n$ (thanks to the $\mathsf{DCR}$ assumption over $\mathbb{Z}_n$), this game is indistinguishable from the previous one.

**Game $\mathbf{G}_{14}$:** We simulate the $\mathbb{Z}_n^*-\mathsf{ESP}$ from additive to multiplicative at random.

**Input and Setup.** As in the previous game.

**Switches.** For each $\mathbb{Z}_n^*-\mathsf{ESP}$ from additive to multiplicative, $\mathscr{S}im$ simulates it on the inputs $(C_{1\times}, C_{1+})$, for a random $C_{1+} \xleftarrow{\$} \mathbb{Z}_{n^2}^*$.

Under the $\mathsf{IND\text{-}CPA}$ security of the Paillier encryption scheme modulo $n$ (thanks to the $\mathsf{DCR}$ assumption over $\mathbb{Z}_n$), this game is indistinguishable from the previous one.

**Game $\mathbf{G}_{15}$:** We alter again the ciphertexts $C_2$, $C_2'$ in the **multiplicative to additive direction**.

**Input and Setup.** As in the previous game.

**Switches.** For each $\mathsf{ESP}_{\times}^{+}(C, \bar{C})$, $\mathscr{S}im$ simulates $C_2$ and $C_2'$ at random.

Under the $\mathsf{IND\text{-}CPA}$ security of the Paillier encryption scheme modulo $n$ (thanks to the $\mathsf{DCR}$ assumption over $\mathbb{Z}_n$) and the $\mathsf{IND\text{-}CPA}$ security of $\mathsf{QR}_n$-$\mathsf{EG}$ (thanks to the $\mathsf{DDH}$ assumption in $\mathsf{QR}_n$), this game is indistinguishable from the previous one.

**Game $\mathbf{G}_{16}$:** We alter again the decryption of $C_3$ in the **additive to multiplicative direction**.

**Input and Setup.** As in the previous game.

**Switches.** For each $\mathsf{ESP}_{+}^{\times}(C, \bar{C})$, $\mathscr{S}im$ sends $D_3 \xleftarrow{\$} \mathbb{Z}_{n^2}^*$.

Under the $\mathsf{IND\text{-}CPA}$ security of the Paillier encryption scheme modulo $n$ (thanks to the $\mathsf{DCR}$ assumption over $\mathbb{Z}_n$), this game is indistinguishable from the previous one.

**Game $\mathbf{G}_{17}$:** We also alter the ciphertext $C_3'$.

**Input and Setup.** As in the previous game.

**Switches.** For each $\mathsf{ESP}_{+}^{\times}(C, \bar{C})$, $\mathscr{S}im$ simulates $C_3' \xleftarrow{\$} \mathsf{QR}_n^2$.

$C_3'$ should contain $\bar{T'}^b/R$ for a random square $R = k \cdot R_3$, that is never revealed anymore: whatever the value of $b$ is, this is a random square: This game is perfectly indistinguishable from the previous one. Note that we do not need to extract $b$ anymore.

**Game $\mathbf{G}_{18}$:** We can now choose random partial secret keys.

**Input.** $\mathscr{S}im$ does not know the factorization of $n$, nor of $N$.

**Setup.** $\mathscr{S}im$ randomly chooses $\chi \leftarrow \mathbb{Z}_n^* \setminus \mathbb{J}_n$, and this virtually defines $p$ such that $\chi$ is a square modulo $p$. $\mathscr{S}im$ chooses $(d_B, v_B, t_{pB}, t_{qB}, s_B, D_B) \xleftarrow{\$} \mathbb{Z}_{n^2/2} \times \mathbb{Z}_n \times \mathbb{Z}_{n/2}^3 \times \mathbb{Z}_{N^2/2}$. $\mathscr{S}im$ picks at random $g_0 \xleftarrow{\$} \mathbb{Z}_n^*$ to set $g \leftarrow -g_0^2 \bmod n$, a generator of $\mathbb{J}_n$. It also picks $g_1 \xleftarrow{\$} \mathbb{J}_n$, as well as $g_2, g_3 \xleftarrow{\$} \mathsf{QR}_n$. It then sets $\mathsf{pk} \leftarrow (n, g, \chi, g_1, g_2, g_3, N)$. It picks $(d_B, v_B, t_{pB}, t_{qB}, s_B, s_{2B}, D_B) \xleftarrow{\$} \mathbb{Z}_{n^2/2} \times \mathbb{Z}_n \times \mathbb{Z}_{n/2}^3 \times \mathbb{Z}_{n/4} \times \mathbb{Z}_{N^2/2}$ and sends them to the adversarial Bob, together with $\mathsf{pk}$.

Since the distributions of the partial keys are statistically close to the original ones, this game is statistically indistinguishable from the previous one.

**Game $\mathbf{G}_{19}$:** We can now choose random public parameters in $\mathbb{J}_n$.

**Input.** $\mathscr{S}im$ receives two strong RSA moduli $n$ and $N$ with $N > 2n^2$.

**Setup.** $\mathscr{S}im$ randomly chooses $\chi \leftarrow \mathbb{Z}_n^* \setminus \mathbb{J}_n$, as well as $g, g_1, g_2, g_3 \xleftarrow{\$} \mathbb{J}_n$. The rest of the game is unchanged.

Under the $\mathsf{QR}$ assumption in $\mathbb{Z}_n^*$, this game is indistinguishable from the previous one.

**Game $\mathbf{G}_{20}$:** We can now choose random ciphertexts in $\mathbb{J}_n$.

**Input and Setup.** As in the previous game. **Switches.** All the ciphertexts with random plaintexts in $\mathsf{QR}_n$ are replaced with random plaintexts in $\mathbb{J}_n$.

Under the $\mathsf{QR}$ assumption in $\mathbb{Z}_n^*$, this game is indistinguishable from the previous one.

*$\mathbb{Z}_n-$ESP is Zero-Knowledge for Bob.* Let us first exhibit the simulator, we then prove that the behavior of the simulator is indistinguishable from Bob's behavior.

$\mathscr{S}im$ first receives as input two strong RSA moduli $n$ and $N$ with $N > (2 + 2^{\kappa+1})n^2$. It picks $g, g_1, g_2, g_3 \xleftarrow{\$} \mathbb{J}_n$ and $\chi \xleftarrow{\$} \mathbb{Z}_n^* \backslash \mathbb{J}_n$. It then sets $\mathsf{pk} \leftarrow (n, g, \chi, g_1, g_2, g_3, N)$. It also picks $(d_A, v_A, t_{pA}, t_{qA}, s_A, s_{2A}, D_A) \xleftarrow{\$} \mathbb{Z}_{n^2/2} \times \mathbb{Z}_n \times \mathbb{Z}_{n/2}^3 \times \mathbb{Z}_{n/4} \times \mathbb{Z}_{N^2/2}$ and sends them to the adversarial Alice, together with $\mathsf{pk}$.

Each time $\mathscr{S}im$ is asked to participate to an instance of a $\mathbb{Z}_n-\mathsf{ESP}$ on an input $C$, it additionally receives a target ciphertext $\bar{C}$, and the expected output is a randomization of $\bar{C}$:

- If the direction is from additive to multiplicative,
  - $\mathscr{S}im$ runs the simulation of $\mathsf{EZT}$ on $(C, C' \xleftarrow{\$} \mathbb{Z}_{n^2}^*)$;
  - $\mathscr{S}im$ parses $\bar{C} = (\bar{C}_{1\times}, \bar{C}_{2\times}, \bar{C}_{3\times})$, and runs the simulation of the $\mathsf{ESP}$ protocol on the input $(C_{1+}, \bar{C}_{1\times})$;
  - $\mathscr{S}im$ extracts $k$ from the commitment $c$ and sends $C_{2\times} \leftarrow \bar{C}_{2\times}$ and $C_{3\times} \leftarrow k \bullet \bar{C}_{3\times}$ to Alice.
- If the direction is from multiplicative to additive,
  - $\mathscr{S}im$ randomly picks $C_{1+} \xleftarrow{\$} \mathbb{Z}_{n^2}^*$, and runs the simulation of the $\mathsf{ESP}$ protocol on the inputs $(C_{1\times}, C_{1+})$;
  - $\mathscr{S}im$ sends $C'_{2+} \xleftarrow{\$} \mathbb{Z}_{n^2}^*$ to Alice;
  - $\mathscr{S}im$ runs the simulation of $\mathsf{EZT}$ on $(C'_{2+}, C' \xleftarrow{\$} \mathbb{Z}_{n^2}^*)$;
  - $\mathscr{S}im$ randomly picks $A_1, A_2 \xleftarrow{\$} \mathbb{Z}_n$, $B_1, B_2 \xleftarrow{\$} \mathbb{Z}_{n^2}^*$, sets $A'_1 \leftarrow (1 + A_1 n)/B_1^{d_A} \bmod n^2$, $A'_2 \leftarrow (1 + A_2 n)/B_2^{d_A} \bmod n^2$, and $B_6 \leftarrow \bar{C} \boxminus \mathscr{E}_{\oplus}(A_1 \cdot A_2)$. It sends the tuple $(B_1, B_2, A'_1, A'_2, B_6)$ to Alice.

With the following sequence of games, we show that the simulator $\mathscr{S}im$ has the same behavior as Bob, even for many instances. The simulators will be constructed step by step, starting from a simulator that knows all the secrets, and then behaves exactly as Bob.

**Game $\mathbf{G}_0$:** This is the real game.

**Input.** In this game, $\mathscr{S}im$ knows the factorization of both $n$ and $N$, with two generators $g, h$ of $\mathbb{J}_n$, as well as the extraction key for the commitement scheme.

**Setup.** $\mathscr{S}im$ generates all the secret keys for both Alice and Bob in an honest way, gives his secret key to Alice.

**Switches.** Each time it is asked to participate to an instance of the switching protocol on an input $(C, \bar{C})$, $\mathscr{S}im$ plays honestly the real game with the secret key of Bob.

**Game $\mathbf{G}_1$:** This simulator uses the trapdoor $s_3$.

**Input and Setup.** As in the previous game, excepted that $\mathscr{S}im$ keeps the secret $s_3$.

**Switches.** For each $\mathsf{ESP}_{\times}^{+}(C, \bar{C})$ or $\mathsf{ESP}_{+}^{\times}(C, \bar{C})$, either $C$ or $\bar{C}$ is $\mathscr{E}_{\otimes}^{0}(m)$, and then it uses $s_3$ to extract $b$ from the third part: if the plaintext is 1, then $b$ is set to 0, otherwise it is set it 1. Since no modification is done, this game is perfectly indistinguishable from the previous one.

**Game $\mathbf{G}_2$:** We start to modify the simulation in the **multiplicative to additive direction**, by altering $C'_{2+}$.

**Input and Setup.** As in the previous game.

**Switches.** For each $\mathsf{ESP}_{\times}^{+}(C, \bar{C})$, $\mathscr{S}im$ chooses a random element $R \xleftarrow{\$} \mathbb{Z}_n^*$, and generates a fresh ciphertext $C'_{2+} \leftarrow \mathscr{E}_{\oplus}(b \cdot R)$, with the extracted $b$.

$C_{2\times}$ contains $T^b$, so $C_2$ contains $x_2 = R_2 \cdot T^{bk'}$, while $C'_2$ contains $R_2^{-1}$: $C'_{2\times}$ contains $k(T^{bk'} - 1) \bmod n$. As a consequence, if $b = 0$ this is 0, otherwise $T^{bk'} \neq 1 \bmod n$, and so $C'_{2\times}$ likely contains a random element in $\mathbb{Z}_n^*$: This game is statistically indistinguishable from the previous one.

**Game $\mathbf{G}_3$:** We can now modify the simulation of the $\mathsf{EZT}$.

**Input and Setup.** As in the previous game.

**Switches.** For each $\mathsf{ESP}_{\times}^{+}(C, \bar{C})$, $\mathscr{S}im$ simulates the $\mathsf{EZT}$ protocol on the input $(C'_{2+}, \mathscr{E}_{\oplus}(1 - b))$, from the extracted $b$.

From the previous games, $C'_{2+}$ contains a Paillier encryption of $b \times R$ for a random $R \xleftarrow{\$} \mathbb{Z}_n^*$. So the $\mathsf{EZT}$ output should be a ciphertext of $b' = 1 - b$. Under the (statistical) zero-knowledge property of the $\mathsf{EZT}$ protocol, this game is (statistically) indistinguishable from the previous one.

**Game $\mathbf{G}_4$:** We now alter the generation of $B_6$.

**Input and Setup.** As in the previous game.

**Switches.** For each $\mathsf{ESP}_{\times}^{+}(C, \bar{C})$, $\mathscr{S}im$ generates $B_6 \leftarrow \bar{C} \boxminus \mathscr{E}_{\oplus}(A_1 \cdot A_2)$, where $A_1 \leftarrow m + b + \rho_0 \bmod n$ and $A_2 \leftarrow \rho + 1 - b \bmod n$, with the extracted $b$ and the extracted message $m$.

By construction, $C_{1\times}$ contains an encryption of $m + b$, and so do is for $C_{1+}$, then $B_1$ contains $A_1 = m + b + \rho_0$, $B_2$ contains an encryption of $\rho + 1 - b$. In addition, when $(C, \bar{C})$ are twin ciphertexts, $C'$ should be equivalent to $\bar{C}$, hence this game is perfectly indistinguishable from the previous one.

**Game $\mathbf{G}_5$:** We now alter the generation of $B_1, B_2$ and $A'_1$, $A'_2$.

**Input and Setup.** As in the previous game.

**Switches.** For each $\mathsf{ESP}^{+}_{\times}(C, \bar{C})$, $\mathscr{S}im$ generates fresh ciphertexts $B_1 \xleftarrow{\$} \mathscr{E}_\oplus(A_1)$ and $B_2 \xleftarrow{\$} \mathscr{E}_\oplus(A_2)$ for random $A_1, A_2 \xleftarrow{\$} \mathbb{Z}_n$, as well as $A'_1 \leftarrow (1 + A_1 n)/B_1^{d_A}$ and $A'_2 \leftarrow (1 + A_2 n)/B_2^{d_A}$.

This corresponds to choose $\rho_0 \leftarrow A_1 - m - b$ and $\rho_1 \leftarrow A_2 + b - 1$: This game is perfectly indistinguishable from the previous one.

**Game $\mathbf{G}_6$:** We now modify the simulation in the **additive to multiplicative direction**, with the $\mathsf{EZT}$.

**Input and Setup.** As in the previous game.

**Switches.** For each $\mathsf{ESP}^{\times}_{+}(C, \bar{C})$, $\mathscr{S}im$ simulates the $\mathsf{EZT}$ protocol on the input $(C, \mathscr{E}_\oplus(b))$, from the extracted $b$.

When $(C, \bar{C})$ are twin ciphertexts, the third part of $\bar{C}$ contains a $\mathsf{QR}_n$-$\mathsf{EG}'$ encryption of $\bar{T}'^b$, where $b = 1$ if the message is 0, and $b = 0$ otherwise. This is the bit $b$ we've extracted above, and that should be encrypted in the result of the honest $\mathsf{EZT}$ protocol on $C$. Under the (statistical) zero-knowledge property of the $\mathsf{EZT}$ protocol, this game is (statistically) indistinguishable from the previous one.

**Game $\mathbf{G}_7$:** We now alter the ciphertext $C_{2\times}$.

**Input and Setup.** As in the previous game.

**Switches.** For each $\mathsf{ESP}^{\times}_{+}(C, \bar{C})$, $\mathscr{S}im$ sends $C_{2\times} \leftarrow \bar{C}_2$.

When $(C, \bar{C})$ are twin ciphertexts, this flow is indistinguishable from the previous game since $C_{2\times}$ should look like a fresh encryption of $T^b$ for some random square $T$: This game is statistically indistinguishable from the previous one.

**Game $\mathbf{G}_8$:** We now alter the ciphertext $C'_{3\times}$.

**Input and Setup.** As in the previous game.

**Switches.** For each $\mathsf{ESP}^{\times}_{+}(C, \bar{C})$, $\mathscr{S}im$ first extracts $k$ from the commitment $c$, and sends $C'_{3\times} \leftarrow k \bullet \bar{C}_3$.

When $(C, \bar{C})$ are twin ciphertexts, this flow is indistinguishable from the previous game since $C_{3\times}$ should look like a fresh encryption of $k{\cdot}T'^b$ for some random square $T'$, because of the multiplication by $k$ in $C_3$: This game is statistically indistinguishable from the previous one.

**Game $\mathbf{G}_9$:** We continue by simulating the $\mathbb{Z}_n^*{-}\mathsf{ESP}$ all in parallel in both directions.

**Input.** $\mathscr{S}im$ does not know the factorization of $n$, nor of $N$.

**Setup.** $\mathscr{S}im$ randomly generates the public keys without knowing the secret keys, excepted $s_3$.

**Switches.** For each instance of the switching protocol, we run the simulators for the $\mathbb{Z}_n^*{-}\mathsf{ESP}$:

- from additive to multiplicative, on an input $(C, \bar{C})$, where one can parse $\bar{C} = (\bar{C}_{1\times}, \bar{C}_{2\times}, \bar{C}_{3\times})$, $\mathscr{S}im$ simulates the $\mathbb{Z}_n^*{-}\mathsf{ESP}$ protocol on the inputs $(C_{1+}, \bar{C}_{1\times})$.
  When $(C, \bar{C})$ are twin ciphertexts, in $\bar{C} = (\bar{C}_{1\times}, \bar{C}_{2\times}, \bar{C}_{3\times})$, $\bar{C}_{1\times}$ should be $\mathscr{E}_\otimes(m + b)$.
- from multiplicative to additive, on an input $(C, \bar{C})$, $\mathscr{S}im$ simulates the $\mathbb{Z}_n^*{-}\mathsf{ESP}$ protocol on the inputs $(C_{1\times}, \bar{C} \boxplus \mathscr{E}_\oplus(b))$.
  When $(C, \bar{C})$ are twin ciphertexts, $\bar{C} = \mathscr{E}_\oplus(m)$, so $\bar{C} \boxplus \mathscr{E}_\oplus(b)$ contains $m + b$.

Under the zero-knowledge property of the $\mathsf{ESP}$ protocols, thanks to the $\mathsf{DDH}$ assumption in $\mathsf{QR}_n$, the $\mathsf{QR}$ assumption in $\mathbb{Z}_n^*$, the $\mathsf{DCR}$ assumption over $\mathbb{Z}_n^*$, and the $\mathsf{DCR}$ assumption over $\mathbb{Z}_N^*$, this game is indistinguishable from the previous one. This corresponds to inline the games from the proof of Theorem 10. One can note that our simulator does not need to know any secret excepted $s_3$.

**Game $\mathbf{G}_{10}$:** We now start to remove the need of $b$.

**Input and Setup.** As in the previous game (no more secret excepted $s_3$).

**Switches.** For each $\mathsf{EZT}$ protocol, a random ciphertext $C' \xleftarrow{\$} \mathbb{Z}_{n^2}^*$ is given as second input.

Under the $\mathsf{IND}$-$\mathsf{CPA}$ security of the Paillier encryption scheme modulo $n$ (thanks to the $\mathsf{DCR}$ assumption over $\mathbb{Z}_n$), this game is indistinguishable from the previous one.

**Game $\mathbf{G}_{11}$:** We simulate the $\mathbb{Z}_n^*{-}\mathsf{ESP}$ from additive to multiplicative at random.

**Input and Setup.** As in the previous game.

**Switches.** For each $\mathbb{Z}_n^*{-}\mathsf{ESP}$ from additive to multiplicative, $\mathscr{S}im$ simulates it on the inputs $(C_{1\times}, C_{1+})$, for a random $C_{1+} \xleftarrow{\$} \mathbb{Z}_{n^2}^*$.

Under the $\mathsf{IND\text{-}CPA}$ security of the Paillier encryption scheme modulo $n$ (thanks to the $\mathsf{DCR}$ assumption over $\mathbb{Z}_n$), this game is indistinguishable from the previous one.

**Game $\mathbf{G}_{12}$:** We alter again the ciphertext $C'_{2+}$ in the **multiplicative to additive direction**.

**Input and Setup.** As in the previous game.

**Switches.** For each $\mathsf{ESP}^{+}_{\times}(C, \bar{C})$, $\mathscr{S}im$ simulates $C'_{2+} \xleftarrow{\$} \mathbb{Z}^*_{n^2}$.

Under the $\mathsf{IND\text{-}CPA}$ security of the Paillier encryption scheme modulo $n$ (thanks to the $\mathsf{DCR}$ assumption over $\mathbb{Z}_n$), this game is indistinguishable from the previous one. Note that we do not need to extract $b$ anymore.

**Game $\mathbf{G}_{13}$:** We alter again the ciphertexts $B_1$ and $B_2$.

**Input and Setup.** As in the previous game.

**Switches.** For each $\mathsf{ESP}^{+}_{\times}(C, \bar{C})$, $\mathscr{S}im$ simulates $B_1, B_2 \xleftarrow{\$} \mathbb{Z}^*_{n^2}$.

Under the $\mathsf{IND\text{-}CPA}$ security of the Paillier encryption scheme modulo $n$ (thanks to the $\mathsf{DCR}$ assumption over $\mathbb{Z}_n$), this game is indistinguishable from the previous one.

**Game $\mathbf{G}_{14}$:** We can now choose random partial secret keys.

**Input.** $\mathscr{S}im$ does not know the factorization of $n$, nor of $N$.

**Setup.** $\mathscr{S}im$ randomly chooses $\chi \leftarrow \mathbb{Z}^*_n \setminus \mathbb{J}_n$, and this virtually defines $p$ such that $\chi$ is a square modulo $p$. $\mathscr{S}im$ chooses $(d_B, v_B, t_{pB}, t_{qB}, s_B, D_B) \xleftarrow{\$} \mathbb{Z}_{n^2/2} \times \mathbb{Z}_n \times \mathbb{Z}^3_{n/2} \times \mathbb{Z}_{N^2/2}$. $\mathscr{S}im$ picks at random $g_0 \xleftarrow{\$} \mathbb{Z}^*_n$ to set $g \leftarrow -g_0^2 \bmod n$, a generator of $\mathbb{J}_n$. It also picks $g_1 \xleftarrow{\$} \mathbb{J}_n$, as well as $g_2, g_3 \xleftarrow{\$} \mathsf{QR}_n$. It then sets $\mathsf{pk} \leftarrow (n, g, \chi, g_1, g_2, g_3, N)$. It picks $(d_A, v_A, t_{pA}, t_{qA}, s_A, s_{2A}, D_A) \xleftarrow{\$} \mathbb{Z}_{n^2/2} \times \mathbb{Z}_n \times \mathbb{Z}^3_{n/2} \times \mathbb{Z}_{n/4} \times \mathbb{Z}_{N^2/2}$ and sends them to the adversarial Alice, together with $\mathsf{pk}$.

Since the distributions of the partial keys are statistically close to the original ones, this game is statistically indistinguishable from the previous one.

**Game $\mathbf{G}_{15}$:** We can now choose random public parameters in $\mathbb{J}_n$.

**Input.** $\mathscr{S}im$ receives two strong RSA moduli $n$ and $N$ with $N > 2n^2$.

**Setup.** $\mathscr{S}im$ randomly chooses $\chi \leftarrow \mathbb{Z}^*_n \setminus \mathbb{J}_n$, as well as $g, g_1, g_2, g_3 \xleftarrow{\$} \mathbb{J}_n$. The rest of the game is unchanged.

Under the $\mathsf{QR}$ assumption in $\mathbb{Z}^*_n$, this game is indistinguishable from the previous one.

## D Details on our Twin-Ciphertext Techniques

### D.1 Security Analysis of TCP's

In Section 6, we proposed a zero-knowledge proof that two ciphertexts from two different cryptosystems do indeed encrypt the same value. The $\mathsf{TCP}$ described in Section 6 is only honest-verifier zero-knowledge: this is a $\Sigma$-protocol, with a typical 3-flow structure. There are two classical ways to enhance the security of such protocols against malicious verifiers, in a sequential or parallel use:

1. By adding a first flow in which the verifier commits to its challenge using an $\mathsf{IND\text{-}CCA}$ extractable commitment scheme (such constructions are called $\Omega$-protocols), or
2. By letting the prover use an equivocal commitment scheme (see [?]) in its first flow.

We will focus on the second technique in our security proof. Note that 3-flow concurrent zero-knowledge cannot exist, hence if one wants concurrent zero-knowledge, the first method shall be used instead. In the setting of $\mathsf{ESP}$, however, it offers a sufficient security: in all the $\mathsf{TCP}$ involved, Alice will always act as the prover and Bob as the verifier. (Even if their roles were to be exchanged, though, one could still use two equivocal schemes, with two different keys, one held by Alice and one by Bob. Then, even if one of the keys could be compromised when a player sees simulated proofs because simulation-soundness is not guaranteed, this key would provide no advantage in proving false statements later).

**Equivocal Commitment Scheme.** The simplest way to commit to all the pairs of ciphertexts in an equivocal way is to use the following construction:

**Setup:** Pick a hash function $H : \{0,1\}^* \mapsto \{0,1\}^\kappa$ and $g_e \xleftarrow{\$} \mathbb{G}$, in a group of public order $n$. Pick $x_e \xleftarrow{\$} \mathbb{Z}_n$; set $h_e \leftarrow g_e^{x_e}$. $(H, g_e, h_e)$ is the public key; $x_e$ is the trapdoor.

**Commit:** $\mathsf{com}_e : (m; r) \mapsto h_e^r g_e^{H(m)}$.

**Equivocate:** Return $(m', r + x_e^{-1}(H(m) - H(m')) \bmod n)$.

**Enhanced $\mathsf{TCP}$.** 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 picks $\rho$. She generates $\kappa$ twin-ciphertext pairs $(C_i, C'_i)_i = (\mathscr{E}_\oplus(\mu_i; r_i), \mathscr{E}_\otimes(\mu_i; s_i))_i$, and sends $c_e \leftarrow \mathsf{com}_e((C_i, C'_i)_i; \rho)$.
- Bob sends a challenge $c = c_1 \cdots c_\kappa \xleftarrow{\$} \{0, 1\}^\kappa$;
- Alice opens $c_e$, and for each $i \leq \kappa$, she sends
  - the plaintext $\mu_i$ 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 from the relations (2), and $\sigma_i \leftarrow (R_i \cdot s_i) \odot ((-1) \cdot s)$ according to the multiplicative case, otherwise;
- Bob checks the openings of commitments and
  - either checks the validity of $(C_i, C'_i)$ with $\mu_i$ 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.

The security of the enhanced $\mathsf{TCP}$ directly deals with malicious verifiers.

**Security.** The **correctness** is straightforward and follows directly from a careful analysis of the protocol.

The **zero-knowledge** relies on the equivocality on the commitment scheme used in the first flow: the simulator (who knows the trapdoor) use Equivocate to open the commitment to a vector of ciphertext pairs such that for each $i \leq \kappa$, the pair is

- a twin-ciphertext pair if $c_i = 0$,
- a ciphertext pair colinear to $(C, C')$ otherwise.

The **soundness** relies on the binding property of the commitment scheme used in the first flow: let us suppose that $(C, C')$ is not a twin-ciphertext pair. For each $i \leq \kappa$, let us show that an adversary, who does not know the trapdoor, cannot answer both the challenge 0 and 1:

- If the adversary opens his commitment to the same pair $(C_i, C'_i)$ for both challenge, then we can check simultaneously check that $(C_i, C'_i)$ is a twin-ciphertext pair and that it is colinear to $(C, C')$. But as $(C, C')$ is not a twin-ciphertext pair, it is impossible: the probability of this event is 0.
- Else, if the adversary open his commitment in two different ways, he breaks the biding property of the Pedersen commitment scheme, which is equivalent to the discrete logarithm problem.

This ensures that if the discrete logarithm is a hard problem, no polynomially bounded adversary can answer more than one challenge, either 0 or 1. Hence, for each $i$, the probability for any polynomially bounded adversary to answer correctly to a randomly chosen challenge $c_i$ is at best $1/2$. Thereby, The probability for such an adversary to answer correctly $\kappa$ challenges is bounded by $1/2^\kappa$.

Even if the $\mathsf{TCP}s$ are crucial to show the security of our $\mathsf{ESP}$s against malicious adversaries their use is very expensive as it relies on cut-and-choose, which implies a multiplicative dependence in the security parameter $\kappa$. It seems unlikely that there exists a simple way to avoid this parameter, due to the non-algebraic structure of our statement. If a given protocol contains $\ell$ switches from one encryption scheme to the other one in total, then the communication complexity of the protocol could naturally be dominated by the large factor $O(\ell\kappa)$ for the $\mathsf{TCP}s$. Fortunately, we mentioned in Section 6 that the heaviest part of the computation can be done in a pre-processing phase. Moreover, a single $\mathsf{TCP}$ can be used to generate several other fresh $\mathsf{TCP}s$ resulting in a thrifty *batch* technique, itself relying on the next argument.

### D.2 New Argument for Multi-Exponentiation with Encrypted Basis

Our batch $\mathsf{TCP}$ relies on a new argument, the *multi-exponentiation with encrypted basis* ($\mathsf{MEB}$) argument.

**Setup.** We describe the setup for our new zero-knowledge arguments, the $\mathsf{MEB}$ argument and the batch $\mathsf{TCP}$. An $\mathsf{MEB}$ argument between a prover $\mathscr{P}$ and a verifier $\mathscr{V}$ has the following characteristics: for any integer $\ell$ polynomially bounded by the security parameter $\kappa$ and a strong RSA modulus $n$ for the Paillier encryption,

**Word:** $(\lambda_i)_{i \leq \kappa} \in (\{0,1\}^\kappa)^\ell$, and $\ell + 1$ Paillier ciphertexts $C, (c_i)_{i \leq \ell}$.
**Statement:** There are some values $r$ and $(m_i, r_i)_{i \leq \ell}$ such that for all $i \leq \ell$, $c_i = \mathscr{E}_\oplus(m_i, r_i)$, and $C = \mathscr{E}_\oplus(\prod_{i=1}^{\ell} m_i^{\lambda_i}; r)$.
**Witness:** $r, (m_i, r_i)_{i \leq \ell}$

The argument assumes the generation of $\mathbb{G}$, a subgroup of order $n$ of a group of prime order $2kn+1$ for some reasonably small $k$; we can expect $k = O(\log(n))$. For each $i \leq \ell$, $\mathsf{com}_\oplus^{(i)} : (x, \rho) \mapsto g_i^x h_t^\rho$ is a perfectly hiding and additively homomorphic commitment scheme as described Section 6. (note that all the $\mathsf{com}_\oplus^{(i)}$ use the same $h_t$). We denote by $\mathsf{com}_G : (x_1, \cdots, x_\ell; \rho) \mapsto \prod_{i=1}^{\ell} g_i^{x_i} \cdot h_t^\rho$ a generalized Pedersen commitment scheme (see [Ped92, BG12]) over size-$\ell$ vectors of elements of $\mathbb{G}$.

**Subroutines.** we overview some zero-knowledge arguments contained in the description of our $\mathsf{MEB}$ argument.

*Argument of Equality of a Committed Value and a Plaintext.* For $i \leq \ell$, let $\gamma$ be an element of $\mathbb{G}$, and let $\varGamma$ be a Paillier ciphertext. In the following, we will call an *equality argument* a zero-knowledge argument proving knowledge of $x \in \mathbb{Z}_n$ such that $\gamma = g_i^x h_t^{r_x}$ and $\varGamma = (1+n)^x \rho_x^n$. Such an argument is folklore, and can be found for example in [?].

*Bayer and Groth Product Arguments.* In [BG12], Bayer and Groth described two zero-knowledge arguments, namely, the Hadamard product argument and the single value product argument. Let $A$ be a matrix of size $\kappa \times \ell$, and $b$ be some element of $\mathbb{Z}_n$. On input $\kappa$ generalized Pedersen commitments, each one committing to a row of $A$ (which is a size-$\ell$ vector), and a commitment $c_b$ to $b$, the combination of the two arguments, the full product argument, allows the prover to show that $b$ is the product of all the entries of $A$. The full argument has a size $O(\ell + \kappa)$ and seven rounds of interaction. It is honest-verifier zero-knowledge, and can be made zero-knowledge against malicious adversaries at the cost of a small additive overhead.

- The *Hadamard product argument* is a batch argument which allows to prove that $O(\kappa)$ proofs of the form "this generalized Pedersen commitment commits to the term-by-term product of two other generalized Pedersen commitments" are simultaneously true; the argument has a size $O(\ell + \kappa)$.
- The *single value product argument* shows that some committed value commits to the product of all components of a vector committed in a given generalized Pedersen commitment.

**Description.** We now proceed with the description of our $\mathsf{MEB}$ argument. It relies on the equality argument and Bayer and Groth products arguments [BG12].

1. $\mathscr{P}$ picks $\rho, (\rho_i)_{i \leq \ell}$ and sends for all $i \leq \ell$, $a_i \leftarrow \mathsf{com}_\oplus^{(i)}(m_i, \rho_i)$ and $a \leftarrow \mathsf{com}_\oplus^{(1)}(\prod_i m_i^{\lambda_i}; \rho)$ (the choice of $\mathsf{com}_\oplus^{(1)}$ is arbitrary). Using equalirt arguments, she proves for each $i$ her knowledge of $(m_i, r_i, \rho_i)$ such that $c_i = (1+n)^{m_i} r_i^n$ and $a_i = g_i^{m_i} h_t^{\rho_i}$ and her knowledge of $(\rho, r)$ such that $C = (1+n)^{\prod_i m_i^{\lambda_i}} r^n$ and $a = g_1^{\prod_i m_i^{\lambda_i}} h_t^\rho$.
2. For each $i$, let $(\lambda_{ij})_{j \leq \kappa}$ be the bit decomposition of $\lambda_i$. Both players locally compute for all $j \in \{0, \cdots, \kappa - 1\}$
$$c_{Gj} \leftarrow \mathsf{com}_G\left((m_i^{\lambda_{ij}})_{i \leq \ell}; \sum \lambda_{ij} \rho_i\right)$$
3. For all $j \in \{0, \cdots, \kappa - 1\}$, $\mathscr{P}$ picks $\rho'_j$ and sends
$$c'_{Gj} \leftarrow \mathsf{com}_G\left((m_i^{\sum_{k=j}^{\kappa-1} 2^{k-j} \lambda_{ik}})_{i \leq \ell}; \rho'_j\right)$$

4. For all $j \in \{1, \cdots, \kappa - 1\}$, $\mathscr{P}$ picks $\rho''_j$ and sends
$$c''_{Gj} \leftarrow \mathsf{com}_G\left((m_i^{\sum_{k=j}^{\kappa-1} 2^{k-j+1}\lambda_{ik}})_{i\leq\ell}; \rho'_j\right)$$
5. $\mathscr{P}$ and $\mathscr{V}$ engage in a Hadamard product argument to prove that for all $j \in \{0, \cdots, \kappa - 1\}$, $c''_{Gj}$ commits to the term-by-term square of the vector committed in $c'_{Gj}$ (which is a Hadamard product of the vector with itself).
6. Let $c''_{G\kappa} \leftarrow \mathsf{com}_G(1, \cdots, 1; 0)$. $\mathscr{P}$ and $\mathscr{V}$ engage in a Hadamard product argument to prove that for all $j \in \{0, \cdots, \kappa - 1\}$, $c'_{Gj}$ commits to the Hadamard product of the vectors committed in $c''_{G(j+1)}$ and $c_{Gj}$.
7. $\mathscr{P}$ and $\mathscr{V}$ engage in a single value product argument to prove that $c$ commits to the product of the components of the vector committed in $c'_{G0}$.

**Security.** About the **correctness**, note that the two Hadamard product arguments and the single value product argument ensure that the $c_{Gi}$, $c'_{Gi}$ and $c''_{Gi}$ commit to the intermediate values at each step of the following square-and-multiply algorithm:

```
function SQUARE-AND-MULTIPLY($(m_i)_i \in \mathbb{Z}_n^{*\ell}, (\lambda_i)_{i\leq\kappa} \in (\{0,1\}^\kappa)^\ell$)
    Let $\bullet_\ell$ denote the Hadamard product between size-$\ell$ vectors
    For each $i$ let $(\lambda_{ij})_j$ be the bit-decomposition of $\lambda_i$
    $\boldsymbol{M} = (M_1, \cdots, M_\ell) \leftarrow (1, \cdots, 1)$
    for $j = \kappa - 1$ to 0 do
        $\boldsymbol{M} \leftarrow \boldsymbol{M} \bullet_\ell \boldsymbol{M} \bullet_\ell (m_i^{\lambda_{ij}})_{i\leq\ell}$
    $m \leftarrow \prod_i M_i$
    return $m$
```

Hence the correctness follows from the correctness of this algorithm. About the **soundness**, it follows from the soundness of classical zero-knowledge proofs (in particular, a simulator can extract the witness $((m_i, r_i)_i, r)$ from $\mathscr{P}$ by calling $\ell + 1$ times the simulator extracting the witness from an equality argument) and the soundness of Bayer and Groth's product arguments. **Computational zero-knowledge** follows directly from the (perfect) hiding property of the commitment schemes and the (computational) zero-knowledge property of the underlying arguments (a simulator can emulate $\mathscr{P}$ by sending random group elements instead of the correct commitments and faking the underlying arguments).

**A Note on the Argument.** It is important to note that the use of $\kappa$ distinct Pedersen commitment scheme with the same $h_t$ is at the core of our argument. Indeed, if the prover had to commit to all the $c_{Gj}$ for $j \leq \kappa$ and prove that each of them commit to the vector $(m_i^{\lambda_{ij}})_i$, we would not be able to avoid a $O(\ell\kappa)$ cost. instead, each $m_i$ is committed separately, and only $\ell$ proofs are required for the correctness of these commitments; then, each player can compute *locally* and in a *verifiable way* the commitments $c_{Gj}$. Then, all the other commitments are proven correct by proving the correctness of their relation to the $c_{Gj}$.

### D.3 Batch TCP and its Security

As a pre-processing and batching technique, we explained in Section 6 that we can pre-process a TCP: Alice and Bob execute a TCP on a twin-ciphertext pair $(C_r, C'_r) = (\mathscr{E}_\oplus(R; r), \mathscr{E}_\otimes(R; r'))$ *randomly chosen* by Alice before the inputs are revealed. Then, in the on-line phase, Alice performs a cheap colinearity proof between the random pair and her target pair $(C, C') = (\mathscr{E}_\oplus(m; \rho), \mathscr{E}_\otimes(m; \rho'))$, by revealing some relations between $m, R$ and the random coins. As $R$ is random, $m$ remains completely hidden (however, a preprocessed twin-ciphertext pair can obviously not be used twice). This means that Alice can preprocess as many TCP as she wants, simultaneously.

In this section, we show how Alice can perform a batch zero-knowledge argument, allowing her to prove that $\ell$ ciphertext pairs are twin-ciphertext pairs using a *single* TCP and an additional argument of size $O(\ell + \kappa)$. In the following section, we always assume that the plaintext are elements of $\mathbb{Z}_n^*$.

Let us construct a batch argument for $\mathsf{TCP}$: Alice wants to prove to Bob that $\ell$ ciphertext pairs $(c_i, c'_i) = (\mathscr{E}_\oplus(m_i; r_i), \mathscr{E}_\otimes(m'_i; r'_i))$ are twin-ciphertext pairs, *i.e.* that for all $i \leq \ell$ $m_i = m'_i$. This argument relies on our $\mathsf{MEB}$ argument and has the same setup. Additionally, the CRS contains the description of a pseudo-random generator (PRG).

1. Bob sends $\lambda \xleftarrow{\$} \{0,1\}^\kappa$. Both players generate $(\lambda_i)_{i\leq\ell}$, using $\lambda$ as a seed for the PRG.
2. Alice picks $\rho$ and sends $C \leftarrow \mathscr{E}_\oplus(\prod_i m_i^{\lambda_i}; \rho)$. Both players compute $C' \leftarrow \prod_i \mathscr{E}_\otimes(m'_i; r'_i)^{\lambda_i} = \mathscr{E}_\otimes(\prod_i {m'}_i^{\lambda_i}; \sum_i r_i\lambda_i)$.
3. Alice and Bob perform an $\mathsf{MEB}$ argument on $((\lambda_i)_{i\leq\ell}, C, (c_i)_{i\leq\ell})$.
4. Alice and Bob perform a $\mathsf{TCP}$ on $(C, C')$.

**Reloading the Pool of Twin-Ciphertext Pairs.** Note that in the fourth phase of our batch $\mathsf{TCP}$, the $\mathsf{TCP}$ on $(C, C')$ can be performed with a colinearity proof between $(C, C')$ and a random twin-ciphertext pair previously proven. This means that once a single $\mathsf{TCP}$ was performed, the pool of twin-ciphertext pairs can by dynamically recharged by "consuming" a pair to prove $\ell$ new pairs, without ever having to perfom a full $\mathsf{TCP}$ again; hence, the original overhead implied by the cut-and-choose in the first $\mathsf{TCP}$ vanishes. As we shown in Section 3 how twin-ciphertext pairs can be used to perform efficiently several zero-knowledge proofs (and not only $\mathsf{TCP}$), this means that two players who have set up and proven a pool of twin-ciphertext pairs can later efficiently recharge the pool to perform new protocols, without having to agree in advance on the protocols they might wish to perform in the future.

**Security.** The $\mathsf{MEB}$ argument shows that $C$ does indeed encrypt $\prod_i m_i^{\lambda_i}$, the $m_i$ being the plaintexts of the $c_i$. $C'$ can be computed publicly and encrypts $\prod_i {m'}_i^{\lambda_i}$. Hence, the final $\mathsf{TCP}$ shows that $\prod_i m_i^{\lambda_i} = \prod_i {m'}_i^{\lambda_i}$. As $C, (c_i)_{i\leq\ell}$ were generated before Bob sent $\lambda$ to Alice, this shows that with overwhelming probability, for each $i \leq \ell$ $m_i = m'_i$. The soundness and the zero-knowledge directly follow from that of the underlying $\mathsf{MEB}$ and $\mathsf{TCP}$.