Efficient Designated-Verifier Non-Interactive Zero-Knowledge Proofs of Knowledge

Pyrros Chaidos[⋆] and Geoffroy Couteau[⋆⋆]

Abstract. We propose a framework for constructing efficient designated-verifier non-interactive zero-knowledge proofs (DVNIZK) for a wide class of algebraic languages over abelian groups, under standard assumptions. The proofs obtained via our framework are proofs of knowledge, enjoy statistical, and unbounded soundness (the soundness holds even when the prover receives arbitrary feedbacks on previous proofs). Previously, no efficient DVNIZK system satisfying *any* of those three properties was known. Our framework allows proving arbitrary relations between cryptographic primitives such as Pedersen commitments, ElGamal encryptions, or Paillier encryptions, in an efficient way. For the latter, we further exhibit the first non-interactive zero-knowledge proof system in the standard model that is more efficient than proofs obtained via the Fiat-Shamir transform, with still-meaningful security guarantees and under standard assumptions. Our framework has numerous applications, in particular for the design of efficient privacy-preserving non-interactive authentication.

Keywords. Zero-knowledge proofs, Non-interactive proofs.

1 Introduction

Zero-knowledge proof systems allow a prover to convince someone of the truth of a statement, without revealing anything beyond the fact that the statement is true. After their introduction in the seminal work of Goldwasser, Micali, and Rackoff [GMR89], they have proven to be a fundamental primitive in cryptography. Among them, *non-interactive zero-knowledge proofs* (NIZK proofs), where the proof consists of a single flow from the prover to the verifier, are of particular interest, in part due to their tremendous number of applications in cryptographic primitives and protocols, and in part due to the theoretical and technical challenges that they represent.

For almost two decades after their introduction in [BFM88], NIZKs coexisted in two types: inefficient NIZKs secure under standard assumptions (such as doubly enhanced trapdoor permutations [FLS90]) in the common reference string model, [1] and practically efficient NIZKs built from the Fiat-Shamir heuristic [FS87, PS96], which are secure in the random oracle model [BR93] (hence only heuristically secure in the standard model). This state of affairs changed with the arrival of pairing-based cryptography, from which a fruitful line of work (starting with the work of Groth, Ostrovsky, and Sahai [GOS06a, GOS06b]) introduced increasingly more efficient NIZK proof systems in the standard model. That line of work culminated with the framework of Groth-Sahai proofs [GS08], which provided an efficient framework of pairing-based NIZKs for a large class of useful languages. Yet, one decade later, pairing-based NIZKs from the Groth-Sahai framework remain the only known efficient NIZK proof system in the standard model. Building efficient NIZKs in the standard model, without pairing-based assumptions, is a major open problem, and research in this direction has proven elusive.

[⋆] National & Kapodistrian University of Athens, Greece

[⋆⋆] Karsruhe Institute of Technology, Germany. Part of this work was made while the second author was at École Normale Supérieure de Paris, France

[1] The common reference string model assumes that a string is drawn from a known distribution in a setup phase by a trusted dealer. This model is the most common model for non-interactive zero-knowledge, as NIZKs in the plain model exist only for trivial languages [Ore87].

1.1 Designated-Verifier Non-Interactive Zero-Knowledge

Parallel to the research on NIZKs, an alternative promising line of research has focused on *designated-verifier* non-interactive zero-knowledge proof systems (DVNIZKs). A DVNIZK retains most of the security properties of a NIZK, but is not publicly verifiable: only the owner of some secret information (the designated verifier) can check the proof. Nevertheless, DVNIZKs can replace publicly verifiable NIZKs in a variety of applications. In addition, unlike their publicly-verifiable counterpart, it is known that efficient DVNIZKs secure in the standard model for rich classes of languages can be constructed without pairing-based assumptions [DFN06, VV09, CG15, Lip17]. However, to date, research in DVNIZKs has attracted less attention than NIZKs, the previously listed papers being (to our knowledge) the only existing works on this topic, and several important questions have been left open. We list the main open questions below.

Proofs Versus Arguments. A non-interactive zero-knowledge argument system is a NIZK in which the soundness property is only required to hold against computationally bounded adversaries. In a NIZK *proof* system, however, soundness is required to hold even against computationally unbounded adversaries.

Currently, while several DVNIZK argument systems have been designed in the standard model without pairing-based assumptions, efficient DVNIZK proof systems without pairings remain an open question. In fact, to our knowledge, the only known constructions of (possibly inefficient) DVNIZK proofs rely on publicly-verifiable NIZK proofs.

Soundness Versus Knowledge Extraction. A non-interactive zero-knowledge proof (or argument) system is a NIZK *of knowledge* if it guarantees that, when the prover succeeds in convincing the verifier, he must *know* a witness for the truth of the statement. This is in constrast with the standard soundness notion, which only guarantees that the statement is true. Formally, this is ensured by requiring the existence of an efficient simulator that can extract a witness from the proof.

Non-interactive zero-knowledge proofs of knowledge are more powerful than standard NIZKs, and the knowledge-extractability property is crucial in many applications. In particular, they are necessary for the very common task of proving relations between values committed with a perfectly hiding commitment scheme, and they are a core component in privacy-preserving authentication mechanisms [BCKL08]. Currently, all known DVNIZK argument systems are not arguments of knowledge. Designing efficient DVNIZKs of knowledge without pairing-based assumptions remains an open question.

Bounded Soundness Versus Unbounded Soundness. The classical soundness security notion for non-interactive zero-knowledge proof systems states that if the statement is not true, no malicious prover can possibly convince the verifier of the truth of the statement with non-negligible probability. While this security notion is sufficient for publicly-verifiable NIZKs, it turns out to be insufficient when considering designated-verifier NIZKs, and corresponds only to a *passive* type of security notion. Indeed, the verification of a DVNIZK involves a secret value, known to the verifier. The fact that a DVNIZK satisfies the standard soundness notion does not preclude the possibility for a malicious prover to learn this secret value, e.g. by submitting a large number of proofs and receiving feedback on whether the proof was accepted or not. Intuitively, this is the same type of issue as for encryption schemes indistinguishable against chosen-plaintext attacks, which can be broken if the adversary is given access to a decryption oracle, or for signature schemes secure against key-only or known-message attacks, which can be broken if the adversary is given access to a signing oracle. Here, an adversary could possibly break the soundness of a DVNIZK if it is given access to a verification oracle.

In practice, this means that as soon as a proof system with bounded soundness is used for more than a logarithmic number of proofs, the soundness property is no longer guaranteed to hold. This calls for a stronger notion of soundness, *unbounded soundness*, which guarantees security even against adversaries that are given arbitrary access to a verification oracle.

Designing a DVNIZK with unbounded soundness has proven to be highly non-trivial. In fact, apart from publicly-verifiable NIZKs (which can be seen as particular types of DVNIZKs where the secret key of the verifier is the empty string), the only known construction of DVNIZK claiming to satisfy unbounded soundness is the construction of [DFN06], where the claim is supported by a proof of security in an idealized model. However, we found this claim to be flawed: there is an explicit attack against the unbounded soundness of any protocol obtained using the compiler of [DFN06], which operates by using slightly malformed proofs to extract the verification key. In Appendix A, we describe our attack, and identify the flaw in the proof of Theorem 5 in [DFN06, Appendix A]. We have notified the authors of our finding and will update future versions of this work with their reply. To our knowledge, constructing designated-verifier zero-knowledge proof systems whose soundness is maintained after polynomially many interactions with the prover remains an open question. In all current constructions, the common reference string and the public key must be refreshed after a logarithmic number of proofs.

1.2 Our Contribution

In this work, we first introduce a framework for designated-verifier NIZKs on group-dependent languages, in the spirit of the Groth-Sahai framework for NIZKs on languages related to pairing-friendly elliptic curves. Our framework only requires that the underlying abelian group on which it is instantiated has order M, where $\mathbb{Z}_M$ is the plaintext-space of an homomorphic cryptosystem with specific properties, and allows to prove a wide variety statements formulated in terms of the operation associated to this abelian group. In particular, we do not need to rely on groups equipped with a pairing. The DVNIZKs obtained with our framework are efficient, as they only require a few group elements and ciphertexts.

The zero-knowledge property of our schemes reduces to the IND-CPA security of the underlying encryption scheme. Additionally, our DVNIZKs enjoy the following properties: they are (adaptively) *knowledge-extractable*; their knowledge-extractability holds *statistically*; their knowledge-extractability is *unbounded*. We stress that previously, no efficient construction of DVNIZK in the standard model satisfying *any* of the above properties was known. The third property, unbounded soundness, was only claimed to hold for the construction of [DFN06], and this claim was formalized with a proof in an idealized model, but as previously mentioned, we found this claim to be flawed. We also point out that in the Groth-Sahai framework, witness extraction is limited either to statements about group elements, or to statements about exponents committed in a bit-by-bit fashion (making the proof highly inefficient). In contrast, our proof system allows to efficiently extract large exponents, without harming the efficiency of the proof. In addition to the above properties, our DVNIZKs satisfy some other useful properties: they are multi-theorem [FLS90], randomizable [BCC$^+$09], and same-string zero-knowledge [DDO$^+$01] (*i.e.*, the common reference string used by the prover and the simulator are the same).

Second, our framework comes with a dual variant, where the role of the encryption scheme and the abelian group are reversed, to prove statements, not about elements of the abelian group, but about the underlying homomorphic encryption scheme. This dual variant leads to DVNIZKs satisfying adaptive statistical unbounded soundness, but not knowledge-extractability (i.e. the dual variant does not give proofs of knowledge).

Third, we show that if one is willing to give up unbounded soundness for efficiency, our techniques can be used to construct extremely efficient DVNIZKs with bounded-soundness. The DVNIZKs that we obtain this way are more efficient than *any* previously known construction of non-interactive zero-knowledge proofs, even when considering NIZKs in the random oracle model using the Fiat-Shamir transform: the proofs we obtain are shorter than the proofs obtained via the Fiat-Shamir transform by almost a factor two. To our knowledge, this is the first example of a NIZK construction in the standard model which (conditionally) improves on the Fiat-Shamir paradigm.

Instantiating the Encryption Scheme. Informally, the security properties we require from the underlying scheme are the following: it must be additively homomorphic, with plaintext space $\mathbb{Z}_M$, random source $\mathbb{Z}_R$, and $\gcd(M, R) = 1$, and it must be *decodable*, which means that a plaintext m can be efficiently recovered from an encryption of m with random coin 0. A natural candidate for the above scheme is the Paillier encryption scheme [Pai99] (and its variants, such as Damgård-Jurik [DJ01]). This gives rise to efficient DVNIZK proofs of knowledge over abelian groups of composite order (e.g. subgroups of $\mathbb{F}_p^*$, with order a prime $p = k \cdot n + 1$ for a small k and an RSA modulus n, or composite-order elliptic curves), as well as efficient DVNIZKs for proving relations between Paillier ciphertexts (using the dual variant of our framework). Alternatively, the scheme can also be instantiated with the more recent Castagnos-Laguillaumie encryption scheme [CL15] to get DVNIZKs over prime-order abelian groups.

Our framework captures many useful zero-knowledge proofs of knowledge that are commonly used in cryptography. This includes DVNIZK proofs of knowledge of a discrete logarithm, of correctness of a Diffie-Hellman tuple, of multiplicative relationships between Pedersen commitments or ElGamal ciphertexts (or variants thereof), among many others. Our results show that, in the settings where a designated-verifier is sufficient, one can build efficient non-interactive zero-knowledge proofs of knowledge for most statements of interest, under well-known assumptions and with strong security properties, without having to rely on pairing-friendly groups.

1.3 Our Method

It is known that linear relations (*i.e.*, membership in linear subspaces) can be non-interactively verified, using the homomorphic properties of cryptographic primitives over abelian groups. Indeed, DVNIZK proofs for linear languages can be constructed, e.g., from hash proof systems [KW15, GHKW16]. As made explicit in the seminal paper of Groth and Sahai [GS08], pairings provide exactly the additional structure that allows to evaluate up to degree-two relations, which can be easily generalized to arbitrary relations. However, this requires to use pairing-friendly elliptic curves, and pairing-based assumptions.

An alternative road was taken in [DFN06] and subsequent works, to obtain non-interactive zero-knowledge proofs for a wide variety of relations, in the designated-verifier setting. To illustrate, let us consider a prover interacting with a verifier, with a common input $(g_1, g_2, h_1, h_2) \in \mathbb{G}^4$ in some group $\mathbb{G}$ of order p, where p is a λ-bit prime. The prover wants to show that (h_1, h_2) have the same discrete logarithm in the basis (g_1, g_2), *i.e.*, there exists x such that $(h_1, h_2) = (g_1^x, g_2^x)$. The standard interactive zero-knowledge proof for this statement proceeds as follows:[2]

1. The prover picks $r \xleftarrow{\$} \{0,1\}^{3\lambda}$, and sends $(a_1, a_2) \leftarrow (g_1^r, g_2^r)$.
2. The verifier picks and sends a uniformly random challenge $e \xleftarrow{\$} \mathbb{Z}_p$.
3. The prover computes and sends $d \leftarrow e \cdot x + r$. The verifier accepts the proof if and only if $(g_1^d, g_2^d) = (h_1^e a_1, h_2^e a_2)$.

The idea of [DFN06] is to squash this interactive protocol into a (designated-verifier) non-interactive proof, by giving the challenge to the prover in advance. As knowing the challenge before sending the first flow gives the prover the ability to cheat, the challenge is encrypted with an additively homomorphic encryption scheme. That way, the prover cannot see the challenge; yet, he can still compute an encryption of the value d homomorphically, using the encryption of e. The verifier, who is given the secret verification key, can decrypt the last flow and perform the above check. Thus, the proof is a tuple (a_1, a_2, c_d), where c_d is an encryption of d homomorphically computed from (x, r) and an encryption c_e of the challenge e.

Although natural, this intuitive approach has proven quite tough to analyze. In [DFN06], the authors had to rely on a new complexity-leveraging-type assumption tailored to their scheme, which (informally) states that the simulator cannot break the security of the encryption scheme, even if he is powerful enough to break the problem underlying the protocol (in the above example, the discrete logarithm problem over $\mathbb{G}$). Even in the bounded setting, analyzing the soundness guarantees of the protocols obtained by this compilation technique (and its variants) is non-trivial, and it has been the subject of several subsequent works [VV09, CG15, Lip17]. Additionally, in the unbounded setting, where we must give an efficient simulator that can successfully answer to the proofs submitted by any malicious prover, this compilation technique breaks down. Furthermore, for DVNIZKs constructed with this method, soundness holds only computationally, and security does not guarantee that the simulator can extract a witness for the statement.

Our core idea to overcome all of the above issues is to implement the same strategy in a slightly different way: rather than encrypting the challenge e as the *plaintext* of an homomorphic encryption scheme, we encrypt it as the *random coin* of an encryption scheme which is also homomorphic over the coins. To understand how this allows us to improve over all previous constructions, suppose that we have an encryption scheme Enc which is homomorphic over both the plaintext and the random coins, with plaintext space $\mathbb{Z}_M$ and random source $\mathbb{Z}_R$, and that M is coprime to R. Consider the previously described protocol for proving equality of two discrete logarithms. Given an encryption $\mathsf{Enc}(0; e)$ of 0, where the challenge is the random coin, a prover holding (x, r) can compute and send $\mathsf{Enc}(x; \rho)$ and $\mathsf{Enc}(r; -e\rho)$, for some random ρ. This allows the verifier, who knows e, to compute $\mathsf{Enc}(x \cdot e + r; 0)$, from which she can extract $d = x \cdot e + r \bmod M$ (note that the verifier only needs to know e; unlike in previous work, she does not need to know the decryption key of Enc). Observe that the extracted value depends only on e *modulo* M. At the same time, however, the ciphertext $E(0; e)$ only leaks e *modulo* R, even to an unbounded adversary. By picking e to be sufficiently large ($e > MR$), as M is coprime to R, the verifier can ensure that this leaks *no information* (statistically) about $e \bmod M$. Therefore, we can use a statistical argument to show that the prover cannot cheat when the verification using d succeeds. To allow for efficient simulation of the verifier, we simply give to the simulator the secret key of the scheme, which will allow him to extract all encrypted values, and to

[2] More formally, this proof only satisfies zero-knowledge against honest verifiers, but this property is sufficient for the construction of [DFN06].

check the validity of the equations, without knowing $e \bmod M$. As the simulator is able to extract the values encrypted with Enc, the scheme can be proven to be (statistically) knowledge-extractable.

Note that in previous constructions of $\mathsf{DVNIZKs}$, the secret key of the underlying encryption scheme was the verification key, which is used by the verifier in the real game. Here, we use a different approach: the verifier does not know the secret key of the underlying scheme (knowing it would break the zero-knowledge property), but only the value of a specific random coin. The secret key is given only to the simulator, who uses it to extract information in the simulated game.

Example: DVNIZK Proof of Knowledge of a Discrete Logarithm. We illustrate our method with the classical example of proving knowledge of a discrete logarithm. For concreteness, we describe an explicit protocol using the Paillier encryption scheme; therefore, this section assumes some basic knowledge of the Paillier encryption scheme. All necessary preliminaries can be found in Section 2. Let $\mathbb{G}$ be a group of order n, where $n = p \cdot q$ is an RSA modulus (*i.e.*, a product of two strong primes). Let g be a generator of $\mathbb{G}$, and let T be a group element. A prover P wishes to prove to a verifier V that he knows a value $t \in \mathbb{Z}_n$ such that $g^t = T$.

Let $h \leftarrow u^n \bmod n^2$, where u denotes an arbitrary generator of $\mathbb{J}_n$, the subgroup of elements of $\mathbb{Z}_n^*$ with Jacobi symbol 1. The Paillier encryption of a message $m \in \mathbb{Z}_n$ with randomness $r \in \mathbb{Z}_{\varphi(n)/2}$ is $\mathsf{Enc}(m; r) = (1+n)^m h^r \bmod n^2$. The public key of the DVNIZK is $E = h^e \in \mathbb{Z}_{n^2}^*$, for a random $e \gg n \cdot \varphi(n)/2$; observe that this is exactly $\mathsf{Enc}(0; e)$. The secret key is e. The DVNIZK proceeds as follows:

The prover P picks $x \xleftarrow{\$} \mathbb{Z}_n$ and a Paillier random coin r, and computes $X \leftarrow g^x$, $T' \leftarrow (1+n)^t h^r \bmod n^2$, and $X' \leftarrow (1+n)^x E^{-r} \bmod n^2$. The verifier V computes $D \leftarrow T^e X \bmod n^2$ and $D' \leftarrow (T')^e X' \bmod n^2$. Then, she checks that D' is of the form $(1+n)^d \bmod n^2$. If so, V computes $d \bmod n$ from D', and checks that $D = g^d$. V accepts iff both checks succeeded.

Let us provide an intuition of the security of this scheme. Correctness follows easily by inspection. Zero-knowledge comes from the fact that T' hides t, under the $\mathsf{IND\text{-}CPA}$ security of Paillier. For statistical knowledge extractability, note E only reveals $e \bmod \varphi(n)$ to an unbounded adversary, which leaks (statistically) no information on $e \bmod n$ as $\varphi(n)$ is coprime to n. This ensures the value t' encrypted in T' must be equal to t, otherwise the verification equations would uniquely define $e \bmod n$, which is statistically unknown to the prover [3]. The simulator knows $\varphi(n)$ (but not $e \bmod n$) and gets t by decrypting T'.

1.4 Applications

A natural application of non-interactive zero-knowledge proofs of knowledge is the design of privacy-preserving non-interactive authentication schemes. This includes classical authentication protocols, but also P-signatures [BCKL08] and their many applications, such as anonymous credentials [BCKL08], group signatures [Cv91], electronic cash [CFN90], or anonymous authentication [TFS04]. Our framework can lead to a variety of efficient new constructions of designated-verifier variants for the above applications without pairings, whereas all previous constructions either had to rely on the random oracle model, or

[3] In fact, we can only prove that the verification equations uniquely define e modulo at least one of the prime factors of n, but as n has only large factors, this is sufficient to complete the proof.

use pairing-based cryptography.[4] In many scenarios of non-interactive authentication, the designated-verifier property is not an issue.

In addition, the aforementioned applications build upon the Groth-Sahai framework for NIZKs. However, Groth-Sahai NIZKs only satisfy a restricted notion of extractability, called f-extractability in [BCKL08]. As a result, constructions of privacy-preserving authentication mechanisms from Groth-Sahai NIZKs require a careful security analysis. Our framework leads to fully extractable zero-knowledge proofs, which could potentially simplify the design of some of the above applications. We note that our DVNIZKs are additionally randomizable, which has applications for delegatable anonymous credential schemes [BCC+09].

Other potential applications of our framework include round-efficient two-party computation protocols secure against malicious adversaries, electronic voting (see e.g. [CG15]), as well as designated-verifier variants of standard cryptographic primitives, such as verifiable encryption [CD00], or verifiable pseudorandom-functions [BCKL09]. Potential applications to the construction of adaptive oblivious transfers can also be envisioned: in [GH08], the authors mention that an adaptive oblivious transfer protocol can be designed by replacing the interactive zero-knowledge proofs of the protocol of [CNs07] by non-interactive one. They raise two issues to this approach, namely, that Groth-Sahai proofs are only witness-indistinguishable for the required class of statements, and that they only satisfy a weak form of extractability. None of these restrictions apply to our DVNIZK constructions.

1.5 Related Work

Non-interactive zero-knowledge proofs were first introduced in [BFM88]. Efficient publicly-verifiable non-interactive zero-knowledge proofs can be constructed in the random oracle model [FS87, PS96, Fis05], or in the non-programmable random oracle model [Lin15] (using a common reference string in addition). The latter construction was improved in [CPSV16]. In the standard model, the main construction of efficient publicly-verifiable NIZKs is the Groth-Sahai framework [GS08].

Designated-verifier non-interactive zero-knowledge arguments where first introduced in [PsV06], where it was shown that the existence of semantically secure encryption implies the existence of DVNIZK arguments with bounded soundness; however, the construction is highly inefficient and therefore only of theoretical interest. Furthermore, even putting aside efficiency consideration, the construction is inherently limited to arguments (as opposed to proofs) with bounded soundness (as opposed to unbounded soundness).

Designated-verifier NIZKs for linear languages can be constructed from hash proof systems [CS02, KW15, GHKW16]. Such NIZKs are perfectly zero-knowledge and statistically adaptively sound, but are not proofs of knowledge and are restricted to very specific statements, captured by linear equations.

Efficient designated-verifier NIZKs for more general statements were first described in [DFN06]. The authors describe a general compiler that converts any three-round (honest-verifier) zero-knowledge protocol satisfying some (mild) requirements into a DVNIZK. However, the construction has several drawbacks: the soundness only holds under a very specific complexity-leveraging assumption, and only against adversaries making at most $O(\log \lambda)$ proofs (as already mentioned, the paper claims that the construction enjoy unbounded soundness as well, but this claim is flawed, see Appendix A. In addition, the proofs obtained with this compiler are not proofs of knowledge.

[4] These applications typically require a proof-friendly signature scheme, but designated-verifier variants of such scheme can easily be constructed (without pairings) from algebraic MACs [CMZ14, KPW15], by committing to the secret key of the MAC and proving knowledge of the committed value with a DVNIZK; such statements are naturally handled by our framework.

In subsequent works [VV09, CG15], variations of the compilation technique of [DFN06] are described, where the complexity-leveraging assumption was replaced by more standard assumptions (although achieving a more restricted type of soundness) by relying on encryption schemes with additional properties. Eventually, [Lip17] removes some of the constraints of the constructions of [CG15], and provides new protocols that can be compiled using the transformation. However, all the constructions obtained in these papers are only computationally sound, do not enjoy unbounded soundness, and are not proofs of knowledge; this strongly limits their scope, and in particular, prevents them from being used in the previously discussed applications.

1.6 Organization

In Section 2, we introduce our notation, and necessary primitives. Section 2 also describes the notion of a DVNIZK-friendly encryption scheme, which is central to our framework. In Section 3, we introduce our framework for building $\mathsf{DVNIZKs}$ of knowledge over an abelian group, illustrate it with practical examples, and prove its security. In Section 4, we describe the dual variant of our framework for proving statements over plaintexts of a DVNIZK-friendly encryption scheme. In Section 5, we describe optimizations on the efficiency of $\mathsf{DVNIZKs}$ for relations between plaintexts of a DVNIZK-friendly scheme, by eschewing unbounded soundness. Eventually, in Appendix A we describe our attack on the unbounded soundness property of the compiler of [DFN06].

2 Preliminaries

Throughout this paper, λ denotes the security parameter. A probabilistic polynomial time algorithm (PPT, also denoted *efficient* algorithm) runs in time polynomial in the (implicit) security parameter λ. A positive function f is *negligible* if for any polynomial p there exists a bound $B > 0$ such that, for any integer $k \geq B$, $f(k) \leq 1/|p(k)|$. An event depending on λ occurs with *overwhelming probability* when its probability is at least $1 - \mathrm{negl}(\lambda)$ for a negligible function negl. Given a finite set S, the notation $x \overset{\$}{\leftarrow} S$ means a uniformly random assignment of an element of S to the variable x. We represent adversaries as interactive probabilistic Turing machines; the notation $\mathscr{A}^{\mathcal{O}}$ indicates that the machine $\mathscr{A}$ is given oracle access to $\mathcal{O}$. Adversaries will sometime output an arbitrary state st to capture stateful interactions.

Abelian Groups and Modules. We use additive notation for groups for convenience, and write $(\mathbb{G}, \boldsymbol{+})$ for an abelian group of order k. When it is clear from the context, we denote 0 its neutral element (otherwise, we denote it $0_{\mathbb{G}}$). We denote by $\bullet$ the scalar-multiplication algorithm (*i.e.* for any $(x, G) \in \mathbb{Z}_k \times \mathbb{G}$, $x \bullet G = G \boldsymbol{+} G \boldsymbol{+} \ldots \boldsymbol{+} G$, where the sum contains x terms). Observe that we can naturally view $\mathbb{G}$ as a $\mathbb{Z}_k$-module $(\mathbb{G}, \boldsymbol{+}, \bullet)$, for the ring $(\mathbb{Z}_k, +, \cdot)$. For simplicity, we write $\boldsymbol{-}G$ for $(-1) \bullet G$. We use lower case to denote elements of $\mathbb{Z}_k$, upper case to denote elements of $\mathbb{G}$, and bold notations to denote vectors. We extend the notations $(\boldsymbol{+}, \boldsymbol{-})$ to vectors and matrices in the natural way, and write $\boldsymbol{x} \bullet \boldsymbol{G}$ to denote the scalar product $x_1 \bullet G_1 \boldsymbol{+} \ldots \boldsymbol{+} x_t \bullet G_t$ (where $\boldsymbol{x}, \boldsymbol{G}$ are vectors of the same length t). For a vector $\boldsymbol{v}$, we denote by $\boldsymbol{v}^\intercal$ its transpose. By $\mathsf{GGen}(1^\lambda)$, we denote a probabilistic efficient algorithm that, given the security parameter λ, generates an abelian group $\mathbb{G}$ such that the best known algorithm for solving discrete logs in $\mathbb{G}$ takes time 2^λ. In the following, we write $(\mathbb{G}, k) \overset{\$}{\leftarrow} \mathsf{GGen}(1^\lambda)$. Additionally, we denote by $\mathsf{GGen}(1^\lambda, k)$ a group generation algorithm that allows us to select the order k beforehand.

RSA Groups. A *strong prime* is a prime $p = 2p' + 1$ such that p' is also a prime. We call *RSA modulus* a product $n = pq$ of two strong primes. We denote by φ Euler's totient function; it holds that $\varphi(n) = (p-1)(q-1)$. We denote by $\mathbb{J}_n$ the cyclic subgroup of $\mathbb{Z}_n^*$ of elements with Jacobi symbol 1 (the order of this group is $\varphi(n)/2$), and by QR_n the cyclic subroup of squares of $\mathbb{Z}_n^*$ (which is also a subgroup of $\mathbb{J}_n$ and has order $\varphi(n)/4$). By $\mathsf{Gen}(1^\lambda)$, we denote a probabilistic efficient algorithm that, given the security parameter λ, generates a strong RSA modulus n and secret parameters (p, q) where $n = pq$, such that the best known algorithm for factoring n takes time 2^λ. In the following, we write $(n, (p, q)) \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$.

2.1 Commitment Schemes

A commitment scheme allows a committer holding a secret value s to send a *commitment* c of s to a verifier, and later on to *open* this commitment to reveal the value s. Such a commitment should *hide* the committed value s to the verifier, but *binds* the committer in opening only s. More formally,

Definition 1. *(Commitment Scheme) A commitment scheme C with message space $\mathcal{M}$, commitment space $\mathcal{C}$, opening space $\mathcal{D}$, and random source $\mathcal{R}$, is a triple of PPT algorithms* $(C.\mathsf{Setup}, C.\mathsf{Com}, C.\mathsf{Verify})$*, such that*

- $C.\mathsf{Setup}(1^\lambda)$ *generates the public parameters* pp *of the scheme,*
- $C.\mathsf{Com}_{\mathsf{pp}}(m; r)$*, given the message $m \in \mathcal{M}$ and some random coins $r \in \mathcal{R}$, outputs a commitment-opening pair (c, d),*
- $C.\mathsf{Verify}_{\mathsf{pp}}(c, d, m)$*, outputs a bit b whose value depends on the validity of the opening (m, d) with respect to the commitment c,*

which satisfies the correctness, hiding, and binding properties defined below.

Definition 2. *(Correctness of a Commitment Scheme) A commitment scheme C is* correct *if for any* $\mathsf{pp} \xleftarrow{\$} C.\mathsf{Setup}(1^\lambda)$*, any message $m \in \mathcal{M}$, and any random coin $r \in \mathcal{R}$, for $(c, d) \leftarrow C.\mathsf{Com}_{\mathsf{pp}}(m; r)$, it holds that $C.\mathsf{Verify}_{\mathsf{pp}}(c, d, m) = 1$.*

Definition 3. *(Hiding Property of a Commitment Scheme) A commitment scheme C is* hiding *if for any PPT adversary $\mathscr{A}$, it holds that*

$$\Pr\left[\begin{array}{ccc} \mathsf{pp} \xleftarrow{\$} C.\mathsf{Setup}(1^\lambda), & (m_0, m_1, \mathsf{st}) \leftarrow \mathscr{A}(\mathsf{pp}), & \\ b \xleftarrow{\$} \{0,1\}, & r \xleftarrow{\$} \mathcal{R}, & : b' = b \\ b' \leftarrow \mathscr{A}(\mathsf{st}, c), & (c, d) \leftarrow C.\mathsf{Com}(m_b; r) & \end{array}\right] \leq \frac{1}{2} + \mu(\lambda)$$

for some function $\mu(\lambda) = \mathrm{negl}(\lambda)$.

Definition 4. *(Binding Property of a Commitment Scheme) A commitment scheme C is* binding *if for any PPT adversary $\mathscr{A}$, it holds that*

$$\Pr\left[\begin{array}{c} \mathsf{pp} \xleftarrow{\$} C.\mathsf{Setup}(1^\lambda), (c, d, m_0, m_1) \leftarrow \mathscr{A}(\mathsf{pp}) : \\ m_0 \neq m_1 \wedge C.\mathsf{Verify}(c, d, m_0) = C.\mathsf{Verify}(c, d, m_1) = 1 \end{array}\right] \leq \mu(\lambda)$$

for some function $\mu(\lambda) = \mathrm{negl}(\lambda)$.

Homomorphic Commitment Scheme. A commitment scheme can also be *homomorphic*, if for a group law $*$ on the message space $\mathcal{M}$, from $(c_0, d_0) \leftarrow \mathsf{Com}(m_0; r_0)$ and $(c_1, d_1) \leftarrow \mathsf{Com}(m_1; r_1)$, one can generate (c, d) from (c_0, c_1) so that $\mathsf{Verify}(c, d, m_0 * m_1) = 1$.

Example: the Pedersen Commitment. We recall below the Pedersen commitment scheme, which commits to integers in $\mathbb{Z}_p$ over a group $\mathbb{G}$ of order p. This scheme is perfectly hiding, and binding under the discrete logarithm assumption over $\mathbb{G}$. In addition, the Pedersen scheme is homomorphic over $\mathbb{Z}_p$.

– $C.\mathsf{Setup}(1^\lambda)$: output $\mathsf{pp} = (G, H) \xleftarrow{\$} \mathbb{G}^2$.
– $C.\mathsf{Com}_{\mathsf{pp}}(m; r)$: given the message $m \in \mathbb{Z}_p$ and some random coins $r \in \mathbb{Z}_p$, output $c = m \bullet G \boldsymbol{+} r \bullet H$ and $d = r$.
– $C.\mathsf{Verify}_{\mathsf{pp}}(c, d, m)$: output 1 if $c = m \bullet G \boldsymbol{+} d \bullet H$, and 0 otherwise.

2.2 Encryption Schemes

We recall the definition of a public-key encryption scheme:

Definition 5. *(Public-Key Encryption Scheme) A public-key encryption scheme S is a triple of PPT algorithms* $(S.\mathsf{KeyGen}, S.\mathsf{Enc}, S.\mathsf{Dec})$*, such that*

– $S.\mathsf{KeyGen}(1^\lambda)$*, generates a pair* $(\mathsf{ek}, \mathsf{dk})$*,* ek *is the public encryption key and* ek *is the secret decryption key. We assume that* ek *specifies the ciphertext space $\mathcal{C}$, the message space $\mathcal{M}$, and the random source $\mathcal{R}$;*
– $S.\mathsf{Enc}_{\mathsf{ek}}(m; r)$*, given the message $m \in \mathcal{M}$ and some random coins $r \in \mathcal{R}$, outputs a ciphertext c;*
– $S.\mathsf{Dec}_{\mathsf{dk}}(c)$*, output a message $m \in \mathcal{M}$;*

which satisfies the correctness and $\mathsf{IND\text{-}CPA}$ *security properties defined below.*

We extend in a natural way the algorithm Enc over vectors: for vectors $\boldsymbol{m} = (m_i)_i \in \mathbb{Z}_M^*$ and $\boldsymbol{r} = (r_i)_i \in \mathbb{Z}_R^*$ of the same size, $S.\mathsf{Enc}_{\mathsf{ek}}(\boldsymbol{m}; r)$ denotes the vector $(S.\mathsf{Enc}_{\mathsf{ek}}(m_i, r_i))_i$. We extend the algorithm Dec to vectors of ciphertexts in a similar way.

Definition 6. *(Correctness of an Encryption Scheme) A public-key encryption scheme S is* correct *if for any pair* $(\mathsf{ek}, \mathsf{dk}) \xleftarrow{\$} S.\mathsf{KeyGen}(1^\lambda)$*, any message $m \in \mathcal{M}$, and any random coin $r \in \mathcal{R}$, decryption is the reverse operation of encryption:* $S.\mathsf{Dec}_{\mathsf{dk}}(S.\mathsf{Enc}_{\mathsf{ek}}(m; r)) = m$.

Definition 7. *(*$\mathsf{IND\text{-}CPA}$ *Security Property of a Public-Key Encryption Scheme) A public-key encryption scheme S is* $\mathsf{IND\text{-}CPA}$ secure *if for any PPT adversary $\mathscr{A}$, it holds that*

$$\Pr\left[\begin{array}{l} (\mathsf{ek}, \mathsf{dk}) \xleftarrow{\$} S.\mathsf{KeyGen}(1^\lambda), \\ (m_0, m_1, \mathsf{st}) \xleftarrow{\$} \mathscr{A}(\mathsf{ek}), b \xleftarrow{\$} \{0,1\} \ : \ \mathscr{A}(\mathsf{st}, c) = b \\ r \xleftarrow{\$} \mathcal{R}, c \leftarrow S.\mathsf{Enc}_{\mathsf{ek}}(m_b; r) \end{array}\right] \leq \frac{1}{2} + \mu(\lambda)$$

for some function $\mu(\lambda) = \mathrm{negl}(\lambda)$.

In this work, we will focus on additively homomorphic encryption schemes, which are homomorphic for both the message and the random coin. More formally, we require that the message space $\mathcal{M}$ and the random source $\mathcal{R}$ are integer sets $(\mathbb{Z}_M, \mathbb{Z}_R)$ for some integers (M, R), and that there exists an efficient operation $\oplus$ such that for any $(\mathsf{ek}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda)$, any $(m_1, m_2) \in \mathbb{Z}_M^2$ and $(r_1, r_2) \in \mathbb{Z}_R^2$, denoting $(C_i)_{i \leq 2} \leftarrow (S.\mathsf{Enc}_{\mathsf{ek}}(m_i; r_i))_{i \leq 2}$, it holds that $C_1 \oplus C_2 = S.\mathsf{Enc}_{\mathsf{ek}}(m_1 + m_2 \bmod M; r_1 + r_2 \bmod R)$. We say an encryption scheme is *strongly additive* if it satisfies these requirements. Note that the existence of $\oplus$ implies (via a standard square-and-multiply method) the existence of an algorithm that, on input a ciphertext $C = S.\mathsf{Enc}_{\mathsf{ek}}(m; r)$ and an integer $\rho \in \mathbb{Z}$, outputs a ciphertext

$C' = S.\mathsf{Enc}_{\mathsf{ek}}(\rho m \bmod M; \rho r \bmod R)$. We denote by $\rho \odot C$ the external multiplication of a ciphertext C by an integer ρ, and by $\ominus$ the operation $C \oplus (-1) \odot C'$ for two ciphertexts (C, C'). We will sometimes slightly abuse these notations, and write $C \oplus m$ (resp. $C \ominus m$) for a plaintext m to denote $C \oplus S.\mathsf{Enc}_{\mathsf{ek}}(m; 0)$ (resp. $C \ominus S.\mathsf{Enc}_{\mathsf{ek}}(m; 0)$).

A simple observation on strongly additively homomorphic encryption schemes is that IND-CPA security implies that R must either be equal to $0 \mod M$, or unknown given ek. Otherwise, an IND-CPA adversary would set $(m_0, m_1) = (0, 1)$ and check if $R \odot C$ equals $S.\mathsf{Enc}_{\mathsf{ek}}(0; 0)$ or $S.\mathsf{Enc}_{\mathsf{ek}}(R; 0)$.

The Paillier Encryption Scheme. The Paillier encryption scheme [Pai99] is a well-known additively homomorphic encryption scheme over $\mathbb{Z}_n$ for an RSA modulus n. We describe here a standard variant [DJN10, Lip17], where the random coin is an exponent over $\mathbb{J}_n$ rather than a group element. Note that the exponent space of $\mathbb{J}_n$ is $\mathbb{Z}_{\varphi(n)/2}$, which is a group of unknown order; however, it suffices to draw exponents at random from $\mathbb{Z}_{n/2}$ to get a distribution statistically close from uniform over $\mathbb{Z}_{\varphi(n)/2}$.

- $\mathsf{KeyGen}(1^\lambda)$: run $(n, (p, q)) \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$, pick $g \xleftarrow{\$} \mathbb{J}_n$, set $h \leftarrow g^n \bmod n^2$, and compute $\delta \leftarrow n^{-1} \bmod \varphi(n)$ (n and $\varphi(n)$ are relatively prime). Return $\mathsf{ek} = (n, h)$ and $\mathsf{dk} = \delta$;
- $\mathsf{Enc}(\mathsf{ek}, m; r)$: given $m \in \mathbb{Z}_n$, for a random $r \xleftarrow{\$} \mathbb{Z}_{n/2}$, compute and output $c \leftarrow (1 + n)^m \cdot h^r \bmod n^2$;
- $\mathsf{Dec}(\mathsf{dk}, c)$: compute $x \leftarrow c^{\mathsf{dk}} \bmod n$ and $c_0 \leftarrow [c \cdot x^{-n} \bmod n^2]$. Return $m \leftarrow (c_0 - 1)/n$.

Note that knowing dk is equivalent to knowing the factorization of n. The IND-CPA security of the Paillier encryption scheme reduces to the decisional composite residuosity (DCR) assumption, which states that it is computationally infeasible to distinguish random n'th powers over $\mathbb{Z}^*_{n^2}$ from random elements of $\mathbb{Z}^*_{n^2}$.[5] It is also strongly additive, where the homomorphic addition of ciphertexts is the multiplication over $\mathbb{Z}^*_{n^2}$.

The ElGamal Encryption Scheme. We recall the additive variant of the famous ElGamal cryptosystem [ElG84], over an abelian group $(\mathbb{G}, \boldsymbol{+})$ of order k.

- $\mathsf{KeyGen}(1^\lambda)$: pick $G \xleftarrow{\$} \mathbb{G}$, pick $s \xleftarrow{\$} \mathbb{Z}_k$, set $G \leftarrow s \bullet G$, and return $\mathsf{ek} = (G, H)$ and $\mathsf{dk} = s$;
- $\mathsf{Enc}(\mathsf{ek}, m; r)$: given $m \in \mathbb{Z}_k$, for a random $r \xleftarrow{\$} \mathbb{Z}_k$, output $\boldsymbol{C} \leftarrow (r \bullet G, (m \bullet G) \boldsymbol{+} (r \bullet H))$;
- $\mathsf{Dec}(\mathsf{dk}, \boldsymbol{C})$: parse $\boldsymbol{C}$ as (C_0, C_1), and compute $M \leftarrow C_1 \boldsymbol{-} (\mathsf{dk} \bullet C_0)$. Compute the discrete logarithm m of M in base G, and return m.

The IND-CPA security of the ElGamal encryption scheme reduces to the decisional Diffie-Hellman (DDH) assumption over $\mathbb{G}$, which states that it is computationally infeasible to distinguish tuples of the form $(G, H, x \bullet G, x \bullet H)$ for random x from uniformly random 4-tuples over $\mathbb{G}$. It is also strongly additive (and the homomorphic operation is the vector addition over $\mathbb{G}$). However, the decryption procedure is not efficient in general, as it requires to compute a discrete logarithm. For the decryption process to be efficient, the message m must be restricted to come from a subset of $\mathbb{Z}_k$ of polynomial size.

[5] In the variant we consider here, we must restrict our attention to elements of $\mathbb{Z}^*_{n^2}$ which have Jacobi symbol 1 when reduced modulo n as $g \in \mathbb{J}_n$, but this can be checked in polynomial time anyway.

DVNIZK-Friendly Encryption Scheme. We say that a strongly additive encryption scheme is DVNIZK-*friendly*, when it satisfies the following additional properties:

– Coprimality Property: we require that the size M of the plaintext space and the size R of the random source are coprime[6], *i.e.*, $\gcd(M, R) = 1$;
– Decodable: for any $(\mathsf{ek}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda)$, the function $f_{\mathsf{ek}} : m \mapsto \mathsf{Enc}_{\mathsf{ek}}(m; 0)$ must be efficiently invertible (*i.e.*, there is a PPT algorithm, which is given ek, computing f_{ek}^{-1} on any value from the image of f_{ek}).

One can observe that the Paillier cryptosystem is DVNIZK-friendly ($\gcd(n, \varphi(n)) = 1$, and any message m can be efficiently recovered from $\mathsf{Enc}_{\mathsf{ek}}(m; 0) = (1 + n)^m \bmod n^2$), while the ElGamal cryptosystem is not (it satisfies none of the above properties). Other DVNIZK-friendly cryptosystems include variants of the Paillier cryptosystem [DJ01, CS02, BCP03, DJ03, DJN10], and the more recent Castagnos-Laguillaumie cryptosystem [CL15], with prime-order plaintext space. For simplicity, we will also assume that all prime factors of the size M of the plaintext space of a DVNIZK-friendly cryptosystem are of superpolynomial size; our results can be extended to cryptosystems with a small plaintext space (or a plaintext space with small prime factors), but at a cost in efficiency. Note that by the homomorphic property, the decodability property implies that a plaintext can always be recovered from a ciphertext if the random coin is known.

2.3 Non-Interactive Zero-Knowledge Proof Systems

In the definitions below, we focus on proof systems for NP-languages that admit an efficient (polynomial-time) prover. For an NP-language $\mathscr{L}$, we denote $R_{\mathscr{L}}$ its associated relation, *i.e.*, a polynomial-time algorithm which satisfies $\mathscr{L} = \{x \mid \exists w, |w| = \mathsf{poly}(|x|) \wedge R_{\mathscr{L}}(x, w) = 1\}$. It is well known that non-interactive proof systems cannot exist for non-trivial languages in the plain model [Ore87]; our constructions will be described in the common reference string model. For conciseness, the common reference string is always implictly given as input to all algorithms. We note that all of our constructions can be readily adapted to work in the registered public-key model as well, a relaxation of the common reference string model introduced by Barak et al in [BCNP04].

While languages are naturally associated to *statements of membership*, the constructions of this paper will mainly consider *statements of knowledge*. We write $\mathsf{St}(x) = \text{Ӄ}\{w : R(x, w) = 1\}$ to denote the statement "I know a witness w such that $R(x, w) = 1$" for a word x and a polytime relation R. Similarly, we write $\mathsf{St}(x) = \exists\{w : R(x, w) = 1\}$ to denote the existential statement "there exists a witness w such that $R(x, w) = 1$".

Definition 8. *(Non-Interactive Zero-Knowledge Proof System) A non-interactive zero-knowledge (*NIZK*) proof system* Π *between for a family of languages* $\mathscr{L} = \{\mathscr{L}_{\mathsf{crs}}\}_{\mathsf{crs}}$ *is a quadruple of probabilistic polynomial-time algorithms* $(\Pi.\mathsf{Setup}, \Pi.\mathsf{KeyGen}, \Pi.\mathsf{Prove}, \Pi.\mathsf{Verify})$ *such that*

– $\Pi.\mathsf{Setup}(1^\lambda)$*, outputs a common reference string* crs *(which specifies the language* $\mathscr{L}_{\mathsf{crs}}$*),*
– $\Pi.\mathsf{KeyGen}(1^\lambda)$*, outputs a public key* pk *and a verification key* vk*,*
– $\Pi.\mathsf{Prove}(\mathsf{pk}, x, w)$*, on input the public key* pk*, a word* $x \in \mathscr{L}_{\mathsf{crs}}$*, and a witness* w*, outputs a proof* π*,*
– $\Pi.\mathsf{Verify}(\mathsf{pk}, \mathsf{vk}, x, \pi)$*, on input the public key* pk*, the verification key* vk*, a word* x*, and a proof* π*, outputs* $b \in \{0, 1\}$*,*

[6] In view of our previous observation on $\mathsf{IND}\text{-}\mathsf{CPA}$ security for strongly additive cryptosystems, this implies that R is secret.

which satisfies the completeness, zero-knowledge, and soundness properties defined below.

We assume for simplicity that once it is generated, the common reference string crs is implicitly passed as an argument to the algorithms $(\Pi.\mathsf{KeyGen}, \Pi.\mathsf{Prove}, \Pi.\mathsf{Verify})$. In the above definition of NIZK proof systems, we let the key generation algorithm generate a verification key vk which is used by the verifier to check the proofs. We call *publicly verifiable non-interactive zero-knowledge proof system* a NIZK proof system in which vk is set to the empty string (or, equivalently, in which vk is made part of the public key). Otherwise, we call it a *designated-verifier non-interactive zero-knowledge proof system.*

Definition 9. *(Completeness) A* NIZK *proof system* $\Pi = (\Pi.\mathsf{Setup}, \Pi.\mathsf{KeyGen}, \Pi.\mathsf{Prove}, \Pi.\mathsf{Verify})$ *for a family of languages* $\mathscr{L} = \{\mathscr{L}_{\mathsf{crs}}\}_{\mathsf{crs}}$ *with relations* R_{crs} *satisfies the (perfect,statistical) completeness property if for* $\mathsf{crs} \overset{\$}{\leftarrow} \Pi.\mathsf{Setup}(1^\lambda)$*, for every* $x \in \mathscr{L}_{\mathsf{crs}}$ *and every witness* w *such that* $R_{\mathsf{crs}}(x, w) = 1$*,*

$$\Pr\begin{bmatrix}(\mathsf{pk}, \mathsf{vk}) \overset{\$}{\leftarrow} \Pi.\mathsf{KeyGen}(1^\lambda), & \\ \pi \leftarrow \Pi.\mathsf{Prove}(\mathsf{pk}, x, w) & \end{bmatrix} : \Pi.\mathsf{Verify}(\mathsf{pk}, \mathsf{vk}, x, \pi) = 1 \Bigg] = 1 - \mu(\lambda)$$

where $\mu(\lambda) = 0$ *for perfect completeness, and* $\mu(\lambda) = \mathrm{negl}(\lambda)$ *for statistical completeness.*

We now define the zero-knowledge property.

Definition 10. *(Composable Zero-Knowledge) A* NIZK *proof system* $\Pi = (\Pi.\mathsf{Setup}, \Pi.\mathsf{KeyGen}, \Pi.\mathsf{Prove}, \Pi.\mathsf{Verify})$ *for a family of languages* $\mathscr{L} = \{\mathscr{L}_{\mathsf{crs}}\}_{\mathsf{crs}}$ *with relations* R_{crs} *satisfies the (perfect, statistical) composable zero-knowledge property if for any* $\mathsf{crs} \overset{\$}{\leftarrow} \Pi.\mathsf{Setup}(1^\lambda)$*, there exists a probabilistic polynomial-time simulator* Sim *such that for any stateful adversary* $\mathscr{A}$*,*

$$\left| \Pr\begin{bmatrix}(\mathsf{pk}, \mathsf{vk}) \overset{\$}{\leftarrow} \Pi.\mathsf{KeyGen}(1^\lambda), & \\ (x, w) \leftarrow \mathscr{A}(\mathsf{pk}, \mathsf{vk}), & : \ (R_{\mathsf{crs}}(x, w) = 1) \wedge (\mathscr{A}(\pi) = 1) \\ \pi \leftarrow \Pi.\mathsf{Prove}(\mathsf{pk}, x, w) & \end{bmatrix} - \Pr\begin{bmatrix}(\mathsf{pk}, \mathsf{vk}) \overset{\$}{\leftarrow} \Pi.\mathsf{KeyGen}(1^\lambda), & \\ (x, w) \leftarrow \mathscr{A}(\mathsf{pk}, \mathsf{vk}), & : \ (R_{\mathsf{crs}}(x, w) = 1) \wedge (\mathscr{A}(\pi) = 1) \\ \pi \leftarrow \mathsf{Sim}(\mathsf{pk}, \mathsf{vk}, x) & \end{bmatrix} \right| \leq \mu(\lambda)$$

where $\mu(\lambda) = 0$ *for perfect composable zero-knowledge, and* $\mu(\lambda) = \mathrm{negl}(\lambda)$ *for statistical composable zero-knowledge. If the composable zero-knowledge property holds against efficient (PPT) verifiers, the proof system satisfies computational composable zero-knowledge.*

The composable zero-knowledge property was first introduced in [Gro06]. It strenghtens the standard zero-knowledge definition, in that it explicitly states that the trapdoor of the simulator is exactly the verification key vk of the verifier. This strong security property guarantees that the same common reference string can be used for many different proofs, as the same trapdoor is used for simulating all proofs, which enhances the proof system with composability properties. We note that [Gro06] additionally required indistinguishability between real and simulated common reference string; in our constructions, this will be trivially satisfied, as the simulated crs will be exactly the real one. We define below the notion of (bounded) adaptive soundness, which allows the input to be adversarially picked after the public key is fixed.

Definition 11. *(Bounded Adaptive Soundness) A* NIZK *proof system* $\Pi = (\Pi.\mathsf{Setup}, \Pi.\mathsf{KeyGen}, \Pi.\mathsf{Prove}, \Pi.\mathsf{Verify})$ *for a family of languages* $\mathscr{L} = \{\mathscr{L}_{\mathsf{crs}}\}_{\mathsf{crs}}$ *with relations* R_{crs}

satisfies the bounded adaptive soundness property if for $\mathsf{crs} \xleftarrow{\$} \mathsf{Setup}(1^\lambda)$*, for every adversary* $\mathscr{A}$*,*

$$\Pr\left[\begin{matrix}(\mathsf{pk},\mathsf{vk}) \xleftarrow{\$} \Pi.\mathsf{KeyGen}(1^\lambda),\\ (\pi, x) \leftarrow \mathscr{A}(\mathsf{pk})\end{matrix} : x \notin \mathscr{L}_{\mathsf{crs}} \wedge \Pi.\mathsf{Verify}(\mathsf{pk},\mathsf{vk},x,\pi)\right] = \mathrm{negl}(\lambda).$$

Definition 11 is formulated with respect to arbitrary adversaries $\mathscr{A}$, which leads to a statistical notion of soundness. A natural relaxation of this requirement is to consider only *efficient* (PPT) adversarial provers. We denote by *computational soundness* this relaxed notion of soundness. Computationally sound proof systems are called *argument systems.*

Unbounded Soundness. Definition 11 corresponds to a *bounded* notion of soundness, in the sense that soundness is only guaranteed to hold when the prover tries to forge a single proof of a wrong statement, right after the setup phase. However, if the prover is allowed to interact polynomially many times with the verifier before trying to forge a proof, sending proofs and receiving feedback on whether the proof was accepted, the previous definition provides no security guarantees.

Intuitively, in this situation, the distinction between bounded and unbounded soundness is comparable to the distinction between security against chosen plaintext attacks and security against chosen ciphertext attacks for cryptosystems. We define unbounded soundness in a similar fashion, by giving the prover access to a verification oracle $\mathcal{O}_{\mathsf{vk}}[\mathsf{pk}]$ (with crs implicitly given as parameter) which, on input (x, π), returns $b \leftarrow \mathsf{Verify}(\mathsf{pk},\mathsf{vk},x,\pi)$.

Definition 12. *(Q-bounded Adaptive Soundness) A* NIZK *proof system* $\Pi = (\Pi.\mathsf{Setup}, \Pi.\mathsf{KeyGen}, \Pi.\mathsf{Prove}, \Pi.\mathsf{Verify})$ *for a family of languages* $\mathscr{L} = \{\mathscr{L}_{\mathsf{crs}}\}_{\mathsf{crs}}$ *with relations* R_{crs} *satisfies the Q-bounded adaptive soundness property if for* $\mathsf{crs} \xleftarrow{\$} \Pi.\mathsf{Setup}(1^\lambda)$*, and every adversary* $\mathscr{A}$ *making at most Q queries to* $\mathcal{O}_{\mathsf{vk}}[\mathsf{pk}]$*, it holds that*

$$\Pr\left[\begin{matrix}(\mathsf{pk},\mathsf{vk}) \xleftarrow{\$} \Pi.\mathsf{KeyGen}(1^\lambda),\\ (\pi, x) \leftarrow \mathscr{A}^{\mathcal{O}_{\mathsf{vk}}[\mathsf{pk}]}(\mathsf{pk})\end{matrix} : x \notin \mathscr{L}_{\mathsf{crs}} \wedge \Pi.\mathsf{Verify}(\mathsf{pk},\mathsf{vk},x,\pi)\right] = \mathrm{negl}(\lambda).$$

Alternatively, the above definition can be formulated with respect to polynomial-time adversarial provers, leading to computational Q-bounded adaptive soundness. Note that the answers of the oracle are bits; therefore, if a NIZK proof system satisfies the bounded adaptive soundness property of Definition 11, it also satisfies the above Q-bounded adaptive soundness property for any $Q = O(\log \lambda)$. Indeed, if Q is logarithmic, one can always guess in advance the answers of the verification oracle with non-negligible (inverse polynomial) probability. We say that a NIZK proof system which is Q-bounded adaptively sound for any $Q = \mathsf{poly}(\lambda)$ satisfies *unbounded adaptive soundness.*

Eventually, we define (unbounded) *knowledge-extractability*, a strenghtening of the soundness property which guarantees that if the prover produces an accepting proof, then the simulator can actually *extract* a witness for the statement. To this aim, we extend the syntax of the Setup algorithm to also output a trapdoor τ, used by the extractor. The knowledge-extractibility guarantee is stronger than soundness, in that the proof guarantees not only that there exists a witness, but also that the prover must know that witness. A NIZK satisfying knowledge-extractability is called a NIZK proof of knowledge.

Definition 13. *(Q-bounded Knowledge-Extractability) A* NIZK *proof system* $\Pi = (\Pi.\mathsf{Setup}, \Pi.\mathsf{KeyGen}, \Pi.\mathsf{Prove}, \Pi.\mathsf{Verify})$ *for a family of languages* $\mathscr{L} = \{\mathscr{L}_{\mathsf{crs}}\}_{\mathsf{crs}}$ *with relations* R_{crs} *satisfies the Q-bounded knowledge-extractability property if for* $(\mathsf{crs}, \tau) \xleftarrow{\$} \Pi.\mathsf{Setup}(1^\lambda)$*, and* *every adversary $\mathscr{A}$ making at most Q queries to $\mathcal{O}_{\mathsf{vk}}[\mathsf{pk}]$, there is an efficient extractor* Ext *such that*

$$\Pr\begin{bmatrix}(\mathsf{pk},\mathsf{vk}) \overset{\$}{\leftarrow} \Pi.\mathsf{KeyGen}(1^\lambda), & \\ (\pi, x) \leftarrow \mathscr{A}^{\mathcal{O}_{\mathsf{vk}}[\mathsf{pk}]}(\mathsf{pk}), & : R_{\mathsf{crs}}(x,w) \textit{ iff } \Pi.\mathsf{Verify}(\mathsf{pk},\mathsf{vk},x,\pi) \\ w \leftarrow \mathsf{Ext}(\pi,x,\tau), & \end{bmatrix} \approx 1.$$

3 A Framework for Designated-Verifier Non-Interactive Zero-Knowledge Proofs of Knowledge

In this section, we let k be an integer, $(\mathbb{G}, +)$ be an abelian group of order k, and (α, β, γ) be three integers. We will describe a framework for proving statements of knowledge over a wide variety of algebraic relations over $\mathbb{G}$, in the spirit of the Groth-Sahai framework for NIZK proofs over bilinear groups. To describe the relations handled by our framework, we describe languages of algebraic relations via linear maps. Such descriptions were previously used for the languages handled by various cryptographic primitives, such as smooth projective hash functions [BBC+13], non-interactive zero-knowledge proofs [BP13], and implicit zero-knowledge proof [BCPW15]. While this system was previously used to describe membership statements, we adapt it to statements of knowledge. As previously observed in [BBC+13], this system encompasses a wider class of languages than the Groth-Sahai framework.

3.1 Statements Defined by a Linear Map over $\mathbb{G}$

Let $\boldsymbol{G} \in \mathbb{G}^\alpha$ denote a vector of *public parameters*, and let $\boldsymbol{C} \in \mathbb{G}^\beta$ denote a public *word.* We will consider statements $\mathsf{St}_\Gamma(\boldsymbol{G}, \boldsymbol{C})$ defined by a linear map $\Gamma : (\mathbb{G}^\alpha, \mathbb{G}^\beta) \mapsto \mathbb{G}^{\gamma\times\beta}$ as follows:

$$\mathsf{St}_\Gamma(\boldsymbol{G}, \boldsymbol{C}) = \text{Ꙅ}\{\boldsymbol{x} \in \mathbb{Z}_k^\gamma \mid \boldsymbol{x} \bullet \Gamma(\boldsymbol{G}, \boldsymbol{C}) = \boldsymbol{C}\}$$

That is, the prover knows a witness-vector $\boldsymbol{x} \in \mathbb{Z}_k^\gamma$ such that the equation $\boldsymbol{x} \bullet \Gamma(\boldsymbol{G}, \boldsymbol{C}) = \boldsymbol{C}$ holds. This abstraction captures a wide class of statementsof interest in cryptography. Below, we describe four examples of statements that can be handled by our framework. They aim at clarifying the way the framework can be used, illustrating its power, as well as providing useful concrete instantiations. The examples focus on the most standard primitives (Pedersen commitments, ElGamal ciphertexts), but the reader will easily recognize they can be naturaly generalized to all standard variants of these primitives (e.g., variants of ElGamal secure under t-linear assumptions [BBS04], or under assumptions from the matrix Diffie-Hellman family of assumptions [EHK+13]).

Example 1: Knowledge of Opening to a Pedersen Commitment. We consider statements of knowledge of an opening (m, r) to a Pedersen commitment C.

- Public Parameters: $(G, H) \in \mathbb{G}^2$;
- Word: $C \in \mathbb{G}$;
- Witness: a pair $(m, r) \in \mathbb{Z}_k^2$ such that $C = m \bullet G + r \bullet H$;
- Linear Map: $\Gamma_{\mathsf{Ped}} : (G, H, C) \mapsto (G, H)^\intercal$;
- Statement: $\mathsf{St}_{\Gamma_{\mathsf{Ped}}}(G, H, C) = \text{Ꙅ}\{(m, r) \in \mathbb{Z}_k^2 \mid (m, r) \bullet (G, H)^\intercal = C\}$.

Example 2: Multiplicative Relationship Between ElGamal Ciphertexts. This type of statement is of particular interest, as it can be easily generalized to arbitrary (polynomial) relationships between plaintexts.

- Public Parameters: $(G, H) \in \mathbb{G}^2$;
- Word: $\boldsymbol{C} = ((U_i, V_i)_{0 \le i \le 2}) \in \mathbb{G}^6$;
- Witness: a 5-tuple $\boldsymbol{x} = (m_0, r_0, m_1, r_1, r_2) \in \mathbb{Z}_k^5$ such that $U_i = r_i \bullet G$ and $V_i = m_i \bullet G \boldsymbol{+} r \bullet H$ for $i = 0, 1$, and $U_2 = m_1 \bullet U_0 \boldsymbol{+} r_2 \bullet G$, $V_2 = m_1 \bullet V_0 \boldsymbol{+} r_2 \bullet H$;
- Linear Map:
$$\Gamma_{\mathsf{EM}} : (G, H, \boldsymbol{C}) \mapsto \begin{pmatrix} 0 & G & 0 & 0 & 0 & 0 \\ G & H & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & G & U_0 & V_0 \\ 0 & 0 & G & H & 0 & 0 \\ 0 & 0 & 0 & 0 & G & H \end{pmatrix};$$
- Statement: $\mathsf{St}_{\Gamma_{\mathsf{EM}}}(G, H, \boldsymbol{C}) = \text{\reflectbox{N}}\{\boldsymbol{x} \in \mathbb{Z}_k^5 \mid \boldsymbol{x} \bullet \Gamma_{\mathsf{EM}}(G, H, \boldsymbol{C}) = \boldsymbol{C}\}$.

Example 3: Knowledge of a Discrete Logarithm. We consider statements defined as follows:

- Public Parameters: $G \in \mathbb{G}$;
- Word: $T \in \mathbb{G}$;
- Witness: a value $t \in \mathbb{Z}_k$ such that $T = t \bullet G$;
- Linear Map: $\Gamma_{\mathsf{DL}} : (G, T) \mapsto G$;
- Statement: $\mathsf{St}_{\Gamma_{\mathsf{DL}}}(G, T) = \text{\reflectbox{N}}\{t \in \mathbb{Z}_k \mid t \bullet \Gamma_{\mathsf{DL}}(G, T) = T\}$.

Example 4: ElGamal Encryption of 0. We consider statements of knowledge of a witness for an ElGamal encryption of 0 (or equivalently, for a DDH tuple), defined as follows:

- Public Parameters: $(G, H) \in \mathbb{G}^2$
- Word: $\boldsymbol{C} = (C_0, C_1) \in \mathbb{G}^2$
- Witness: $r \in \mathbb{Z}_k$ such that $C_0 = r \bullet G$ and $C_1 = r \bullet H$;
- Linear Map: $\Gamma_{\mathsf{EZ}} : (G, H, \boldsymbol{C}) \mapsto (G, H)$;
- Statement: $\mathsf{St}_{\Gamma_{\mathsf{EZ}}}(G, H, \boldsymbol{C}) = \text{\reflectbox{N}}\{r \in \mathbb{Z}_k \mid r \bullet \Gamma_{\mathsf{EZ}}(G, H, \boldsymbol{C}) = \boldsymbol{C}\}$.

Conjunction of Statements. The above framework naturally handles conjuctions of statements. Consider two statements $(\mathsf{St}_{\Gamma_0}(\boldsymbol{G_0}, \boldsymbol{C_0}), \mathsf{St}_{\Gamma_1}(\boldsymbol{G_1}, \boldsymbol{C_1}))$, defined by linear maps (Γ_0, Γ_1), with respective public parameters $(\boldsymbol{G_1}, \boldsymbol{G_1})$, over respective words $(\boldsymbol{C_0}, \boldsymbol{C_1})$, and respective witnesses $(\boldsymbol{x_0}, \boldsymbol{x_1})$. Let $\boldsymbol{G} \leftarrow (\boldsymbol{G_1}, \boldsymbol{G_1})$, $\boldsymbol{C} \leftarrow (\boldsymbol{C_0}, \boldsymbol{C_1})$, and $\boldsymbol{x} \leftarrow (\boldsymbol{x_0}, \boldsymbol{x_1})$. We construct the linear map Γ associated to the statement $\mathsf{St}_\Gamma(\boldsymbol{G}, \boldsymbol{C})$ as

$$\Gamma \leftarrow \begin{pmatrix} \Gamma_0 & 0 \\ 0 & \Gamma_1 \end{pmatrix}.$$

One can immediatly observe that $\mathsf{St}_\Gamma(\boldsymbol{G}, \boldsymbol{C}) = \mathsf{St}_{\Gamma_0}(\boldsymbol{G_0}, \boldsymbol{C_0}) \wedge \mathsf{St}_{\Gamma_1}(\boldsymbol{G_1}, \boldsymbol{C_1})$. We will use this observation in Section 4. Note also that the framework handles disjunction of statements as well, as observed in [ABP15].

3.2 A Framework for DVNIZK Proofs of Knowledge

We now introduce our framework for constructing designated-verifier non-interactive zero-knowledge proofs of knowledge for statements defined by a linear map over $\mathbb{G}$. Let $S = (S.\mathsf{KeyGen}, S.\mathsf{Enc}, S.\mathsf{Dec})$ denote a DVNIZK-friendly encryption scheme with plaintext space $\mathbb{Z}_k$. We construct a DVNIZK of knowledge $\Pi_\mathsf{K} = (\Pi_\mathsf{K}.\mathsf{Setup}, \Pi_\mathsf{K}.\mathsf{KeyGen}, \Pi_\mathsf{K}.\mathsf{Prove}, \Pi_\mathsf{K}.\mathsf{Verify})$ for a statement $\mathsf{St}_\Gamma(\boldsymbol{G}, \boldsymbol{C})$ over a word $\boldsymbol{C} \in \mathbb{G}^\beta$, with public parameters $\boldsymbol{G} \in \mathbb{G}^\alpha$, defined by a linear map $\Gamma : (\mathbb{G}^\alpha, \mathbb{G}^\beta) \mapsto \mathbb{G}^{\gamma \times \beta}$. Our construction proceeds as follows:

- $\Pi_{\mathsf{K}}.\mathsf{Setup}(1^\lambda)$: compute $(\mathsf{ek}, \mathsf{dk}) \xleftarrow{\$} S.\mathsf{KeyGen}(1^\lambda)$. Output $\mathsf{crs} \leftarrow \mathsf{ek}$. Note that ek defines a plaintext space $\mathbb{Z}_k$ and a random source $\mathbb{Z}_R$. As the $\mathsf{IND\text{-}CPA}$ and strong additive properties of S require R to be unknown, we assume that a bound B on R is publicly available. We denote $\ell \leftarrow 2^\lambda k B$.
- $\Pi_{\mathsf{K}}.\mathsf{KeyGen}(1^\lambda)$: pick $e \leftarrow \mathbb{Z}_\ell$, set $\mathsf{pk} \leftarrow S.\mathsf{Enc}_{\mathsf{ek}}(0; e)$ and $\mathsf{vk} \leftarrow e$.
- $\Pi_{\mathsf{K}}.\mathsf{Prove}(\mathsf{pk}, \boldsymbol{C}, \boldsymbol{x})$: on a word $\boldsymbol{C} \in \mathbb{Z}_k^\beta$, with witness $\boldsymbol{x}$ for the statement $\mathsf{St}_\Gamma(\boldsymbol{G}, \boldsymbol{C})$, pick $\boldsymbol{x'} \xleftarrow{\$} \mathbb{Z}_k^\gamma$, $\boldsymbol{r} \xleftarrow{\$} \mathbb{Z}_{2^\lambda B}^\gamma$, compute

$$\boldsymbol{X} \leftarrow S.\mathsf{Enc}_{\mathsf{ek}}(\boldsymbol{x}, \boldsymbol{r}), \qquad \boldsymbol{X'} \leftarrow S.\mathsf{Enc}_{\mathsf{ek}}(\boldsymbol{x'}, 0) \ominus (\boldsymbol{r} \odot \mathsf{pk}), \qquad \boldsymbol{C'} \leftarrow \boldsymbol{x'} \bullet \Gamma(\boldsymbol{G}, \boldsymbol{C}),$$

  and output $\boldsymbol{\pi} \leftarrow (\boldsymbol{X}, \boldsymbol{X'}, \boldsymbol{C'})$.
- $\Pi_{\mathsf{K}}.\mathsf{Verify}(\mathsf{pk}, \mathsf{vk}, \boldsymbol{C}, \boldsymbol{\pi})$: parse $\boldsymbol{\pi}$ as $(\boldsymbol{X}, \boldsymbol{X'}, \boldsymbol{C'})$. Check that $e \odot \boldsymbol{X} \oplus \boldsymbol{X'}$ is decodable, and decode it to a vector $\boldsymbol{d} \in \mathbb{Z}_k^\gamma$. Note that the possibility to check that a ciphertext is decodable is implied by the existence of a decoding algorithm. Check that

$$\boldsymbol{d} \bullet \Gamma(\boldsymbol{G}, \boldsymbol{C}) = e \bullet \boldsymbol{C} \boldsymbol{+} \boldsymbol{C'}.$$

  If all checks succeeded, accept. Otherwise, reject.

The proof $\boldsymbol{\pi}$ consists of 2γ ciphertexts of S, and β elements of $\mathbb{G}$. Below, we illustrate our construction of DVNIZK on the examples of statements given in the previous section. For the sake of concreteness, we instantiate the DVNIZK-friendly encryption scheme S with Paillier (hence the operation $\boldsymbol{+}$ is instantiated as the multiplication modulo n^2), so that the message space is $\mathbb{Z}_n$ and the randomizer space is $\mathbb{Z}_{\varphi(n)/2}$ for an RSA modulus n. In the examples, we use a bound $B = n$ and draw Paillier random coins from $\mathbb{Z}_{2^\lambda B}$, following our generic framework. However, observe that in the case of Paillier, we can also draw the coins from $\mathbb{Z}_{n/2}$ to get a distribution statistically close to uniform over $\mathbb{Z}_{\varphi(n)/2}$, which is more efficient.

Example 1: Knowledge of Opening to a Pedersen Commitment.

- $\Pi_{\mathsf{Ped}}.\mathsf{Setup}(1^\lambda)$: Compute $((n, h), \delta) = (\mathsf{ek}, \mathsf{dk}) \xleftarrow{\$} S.\mathsf{KeyGen}(1^\lambda)$. Output $\mathsf{crs} \leftarrow \mathsf{ek}$. Let $\ell \leftarrow 2^\lambda n^2$. Let $\mathbb{G} \xleftarrow{\$} \mathsf{GGen}(1^\lambda, n)$, $(G, H) \xleftarrow{\$} \mathbb{G}^2$.
- $\Pi_{\mathsf{Ped}}.\mathsf{KeyGen}(1^\lambda)$: pick $e \xleftarrow{\$} \mathbb{Z}_\ell$, set $\mathsf{pk} \leftarrow h^e \bmod n^2$ and $\mathsf{vk} \leftarrow e$.
- $\Pi_{\mathsf{Ped}}.\mathsf{Prove}(\mathsf{pk}, C, \boldsymbol{x})$: on a word $C \in \mathbb{G}$, with witness $\boldsymbol{x} = (m, r) \in \mathbb{Z}_n^2$ for the statement $\mathsf{St}_{\Gamma_{\mathsf{Ped}}}(\boldsymbol{G}, \boldsymbol{C})$, pick $\boldsymbol{x'} \xleftarrow{\$} \mathbb{Z}_n^2$, $\boldsymbol{\rho} \xleftarrow{\$} \mathbb{Z}_{2^\lambda B}^2$, compute $\boldsymbol{X} \leftarrow (1+n)^{\boldsymbol{x}} h^{\boldsymbol{\rho}} \bmod n^2, \boldsymbol{X'} \leftarrow (1+n)^{\boldsymbol{x'}} \mathsf{pk}^{-\boldsymbol{\rho}} \bmod n^2, \boldsymbol{C'} \leftarrow \boldsymbol{x'} \bullet (G, H)^\intercal$, and output $\boldsymbol{\pi} \leftarrow (\boldsymbol{X}, \boldsymbol{X'}, \boldsymbol{C'})$.
- $\Pi_{\mathsf{Ped}}.\mathsf{Verify}(\mathsf{pk}, \mathsf{vk}, \boldsymbol{C}, \boldsymbol{\pi})$: parse $\boldsymbol{\pi}$ as $(\boldsymbol{X}, \boldsymbol{X'}, \boldsymbol{C'})$. Check that $\boldsymbol{X}^e \boldsymbol{X'}$ is of the form $(1+n)^{\boldsymbol{d}}$, and recover the vector $\boldsymbol{d} \in \mathbb{Z}_n^2$. Check that $\boldsymbol{d} \bullet (G, H)^\intercal = e \bullet \boldsymbol{C} \boldsymbol{+} \boldsymbol{C'}$.

Example 2: Multiplicative Relationship Between ElGamal Ciphertexts.

- $\Pi_{\mathsf{EM}}.\mathsf{Setup}(1^\lambda)$ as $\Pi_{\mathsf{Ped}}.\mathsf{Setup}(1^\lambda)$.
- $\Pi_{\mathsf{EM}}.\mathsf{KeyGen}(1^\lambda)$ as $\Pi_{\mathsf{Ped}}.\mathsf{KeyGen}(1^\lambda)$.
- $\Pi_{\mathsf{EM}}.\mathsf{Prove}(\mathsf{pk}, \boldsymbol{C}, \boldsymbol{x})$: on a word $\boldsymbol{C} \in \mathbb{G}^6$, with witness $\boldsymbol{x} = (m_0, r_0, m_1, r_1, r_2) \in \mathbb{Z}_n^5$ for the statement $\mathsf{St}_{\Gamma_{\mathsf{EM}}}(\boldsymbol{G}, \boldsymbol{C})$, pick $\boldsymbol{x'} \xleftarrow{\$} \mathbb{Z}_n^5$, $\boldsymbol{\rho} \xleftarrow{\$} \mathbb{Z}_{2^\lambda B}^5$, compute $\boldsymbol{X} \leftarrow (1+n)^{\boldsymbol{x}} h^{\boldsymbol{\rho}} \bmod n^2, \boldsymbol{X'} \leftarrow (1+n)^{\boldsymbol{x}} \mathsf{pk}^{-\boldsymbol{\rho}} \bmod n^2, \boldsymbol{C'} \leftarrow \boldsymbol{x'} \bullet \Gamma_{\mathsf{EM}}(\boldsymbol{G}, \boldsymbol{C})$, and output $\boldsymbol{\pi} \leftarrow (\boldsymbol{X}, \boldsymbol{X'}, \boldsymbol{C'})$.
- $\Pi_{\mathsf{EM}}.\mathsf{Verify}(\mathsf{pk}, \mathsf{vk}, \boldsymbol{C}, \boldsymbol{\pi})$: parse $\boldsymbol{\pi}$ as $(\boldsymbol{X}, \boldsymbol{X'}, \boldsymbol{C'})$. Check that $\boldsymbol{X}^e \boldsymbol{X'}$ is of the form $(1+n)^{\boldsymbol{d}}$, and recover the vector $\boldsymbol{d} \in \mathbb{Z}_n^5$. Check that $\boldsymbol{d} \bullet \Gamma_{\mathsf{EM}}(\boldsymbol{G}, \boldsymbol{C}) = e \bullet \boldsymbol{C} \boldsymbol{+} \boldsymbol{C'}$.

Example 3: Knowledge of a Discrete Logarithm. Π_{DL} allows a prover to show that he knows $t \in \mathbb{Z}_n \mid t \bullet \Gamma_{\mathsf{DL}}(G, T) = t \bullet G = T$ for a publicly known G.

- $\Pi_{\mathsf{DL}}.\mathsf{Setup}(1^\lambda)$: Compute $((n, h), \delta) = (\mathsf{ek}, \mathsf{dk}) \xleftarrow{\$} S.\mathsf{KeyGen}(1^\lambda)$. Output $\mathsf{crs} \leftarrow \mathsf{ek}$. Let $\ell \leftarrow 2^\lambda n^2$. Let $\mathbb{G} \xleftarrow{\$} \mathsf{GGen}(1^\lambda, n)$, and $G \xleftarrow{\$} \mathbb{G}$.
- $\Pi_{\mathsf{DL}}.\mathsf{KeyGen}(1^\lambda)$: as $\Pi_{\mathsf{Ped}}.\mathsf{KeyGen}(1^\lambda)$.
- $\Pi_{\mathsf{DL}}.\mathsf{Prove}(\mathsf{pk}, T, t)$: On a word $T \in \mathbb{G}$ with witness $t \in \mathbb{Z}_n$ pick $t' \xleftarrow{\$} \mathbb{Z}_n$, $r \xleftarrow{\$} \mathbb{Z}_{2^\lambda n}$, compute
$$X \leftarrow (1+n)^t h^r \bmod n^2, \qquad X' \leftarrow (1+n)^{t'} \mathsf{pk}^{-r} \bmod n^2, \qquad T' \leftarrow t' \bullet G,$$
and output $\boldsymbol{\pi} \leftarrow (X, X', T')$.
- $\Pi_{\mathsf{DL}}.\mathsf{Verify}(\mathsf{pk}, \mathsf{vk}, T, \boldsymbol{\pi})$: parse $\boldsymbol{\pi}$ as (X, X', T'). Check that $X^e X'$ is of the form $(1+n)^d \bmod n^2$, and recover $d \in \mathbb{Z}_n$. Check that $d \bullet G = e \bullet T \boldsymbol{+} T'$.

Example 4: ElGamal Encryption of 0.

- $\Pi_{\mathsf{EZ}}.\mathsf{Setup}(1^\lambda)$: as $\Pi_{\mathsf{Ped}}.\mathsf{Setup}(1^\lambda)$.
- $\Pi_{\mathsf{EZ}}.\mathsf{KeyGen}(1^\lambda)$ as $\Pi_{\mathsf{Ped}}.\mathsf{KeyGen}(1^\lambda)$.
- $\Pi_{\mathsf{EZ}}.\mathsf{Prove}(\mathsf{pk}, \boldsymbol{C}, x)$: on a word $\boldsymbol{C} \in \mathbb{G}^2$, with witness $x \in Z_n$ for the statement $\mathsf{St}_{\Gamma_{\mathsf{EZ}}}(\boldsymbol{G}, \boldsymbol{C})$, pick $x' \xleftarrow{\$} \mathbb{Z}_n$, $r \xleftarrow{\$} \mathbb{Z}_{2^\lambda B}$, compute
$$X \leftarrow (1+n)^x h^r \bmod n^2, \quad X' \leftarrow (1+n)^{x'} \mathsf{pk}^{-r} \bmod n^2, \quad \boldsymbol{C'} \leftarrow x' \bullet \Gamma_{\mathsf{EZ}}(\boldsymbol{G}, \boldsymbol{C}),$$
and output $\boldsymbol{\pi} \leftarrow (X, X', \boldsymbol{C'})$.
- $\Pi_{\mathsf{EZ}}.\mathsf{Verify}(\mathsf{pk}, \mathsf{vk}, \boldsymbol{C}, \boldsymbol{\pi})$: parse $\boldsymbol{\pi}$ as $(X, X', \boldsymbol{C'})$. Check that $X^e X'$ is of the form $(1+n)^d$, and recover $d \in \mathbb{Z}_n$. Check that $d \bullet \Gamma_{\mathsf{EZ}}(\boldsymbol{G}, \boldsymbol{C}) = e \bullet \boldsymbol{C} \boldsymbol{+} \boldsymbol{C'}$.

3.3 Security Proof

We now prove the generic DVNIZK construction from Section 3.2 is secure.

Perfect Completeness. It follows from straighforward calculations: $e \odot \boldsymbol{X} \oplus \boldsymbol{X'} = S.\mathsf{Enc}_{\mathsf{ek}}(e \cdot \boldsymbol{x} + \boldsymbol{x'}; e \cdot \boldsymbol{r} - e \cdot \boldsymbol{r}) = S.\mathsf{Enc}_{\mathsf{ek}}(e \cdot \boldsymbol{x} + \boldsymbol{x'}; 0)$ is decodable and decodes to $\boldsymbol{d} = e \cdot \boldsymbol{x} + \boldsymbol{x'} \bmod k$. Then, $\boldsymbol{d} \bullet \Gamma(\boldsymbol{G}, \boldsymbol{C}) = e \bullet (\boldsymbol{x} \bullet \Gamma(\boldsymbol{G}, \boldsymbol{C})) \boldsymbol{+} \boldsymbol{x'} \bullet \Gamma(\boldsymbol{G}, \boldsymbol{C}) = e \bullet \boldsymbol{C} \boldsymbol{+} \boldsymbol{C'}$ by the correctness of the statement $(\boldsymbol{x} \bullet \Gamma(\boldsymbol{G}, \boldsymbol{C}) = \boldsymbol{C})$ and by construction of $\boldsymbol{C'}$.

Composable Zero-Knowledge. We prove the following theorem:

Theorem 14 (Zero-Knowledge of Π_{K}). *If the encryption scheme S is* IND-CPA *secure, then the* DVNIZK *scheme Π_{K} satisfies composable zero-knowledge.*

We describe a simulator $\mathsf{Sim}(\boldsymbol{C}, \mathsf{pk}, \mathsf{vk})$ producing proofs computationally indistinguishable from those produced by an honest prover on true statements. The simulator operates as follows: let $\boldsymbol{d} \xleftarrow{\$} \mathbb{Z}_k^\gamma$, and $\boldsymbol{C'} \leftarrow \boldsymbol{d} \bullet \Gamma(\boldsymbol{G}, \boldsymbol{C}) \boldsymbol{-} e \bullet \boldsymbol{C}$. Sample $\boldsymbol{x} \xleftarrow{\$} \mathbb{Z}_k^\gamma$, $\boldsymbol{r} \xleftarrow{\$} \mathbb{Z}_{2^\lambda B}^\gamma$, and compute $\boldsymbol{X} \leftarrow S.\mathsf{Enc}_{\mathsf{ek}}(\boldsymbol{x}, \boldsymbol{r}), \boldsymbol{X'} \leftarrow S.\mathsf{Enc}_{\mathsf{ek}}(\boldsymbol{d} - \boldsymbol{e} \cdot \boldsymbol{x}, -\boldsymbol{e} \cdot \boldsymbol{r})$. Output $\pi_s = (\boldsymbol{X}, \boldsymbol{X'}, \boldsymbol{C'})$.

Let $\mathscr{A}$ be an adversary that can distinguish Sim from Prove. We will build a reduction against the IND-CPA security of S. The reduction obtains $\boldsymbol{C}, \boldsymbol{x}$ from $\mathscr{A}$, samples $\tilde{\boldsymbol{x}} \leftarrow \mathbb{Z}_k^\gamma$, sends $(\boldsymbol{x}, \tilde{\boldsymbol{x}})$ to the IND-CPA game and sets $\boldsymbol{X}$ to be the challenge from the IND-CPA game. Now, the reduction samples $\boldsymbol{d} \leftarrow \mathbb{Z}_k^\gamma$ and sets $\boldsymbol{X'} := S.\mathsf{Enc}_{\mathsf{ek}}(\boldsymbol{d}; 0) \ominus \boldsymbol{X} \odot e$. Finally, the reduction sets $\boldsymbol{C'} := \boldsymbol{d} \bullet \Gamma(\boldsymbol{G}, \boldsymbol{C}) \boldsymbol{-} e \bullet \boldsymbol{C}$. Send $\pi^* = (\boldsymbol{X}, \boldsymbol{X'}, \boldsymbol{C})$ to $\mathscr{A}$.

Direct calculation shows that if the IND-CPA game outputs an encryption of $\tilde{\boldsymbol{X}}$, then $\boldsymbol{X}, \boldsymbol{X'}, \boldsymbol{C}$ are distributed as those produced by Sim, whereas when it outputs an encryption of $\boldsymbol{X}$ then π^* is distributed identical to a real proof. Thus, whatever advantage $\mathscr{A}$ has in distinguishing Sim from Prove is also achieved by the reduction against IND-CPA. Note that for simplicity, our proof assume that the IND-CPA game is directly played over vectors, but standard methods allow to reduce this to the classical IND-CPA game with a single challenge ciphertext.

Adaptive Unbounded Knowledge-Extractability. We start by showing that Π_{K} satisfies statistical adaptive unbounded knowledge-extractability. More precisely, we prove the following theorem:

Theorem 15 (Soundness of Π_{K}). *There is an efficient simulator* Sim *such that for any (possibly unbounded) adversary $\mathscr{A}$ that outputs an accepting proof $\boldsymbol{\pi}$ with probability ε on an arbitrary word $\boldsymbol{C}$ after making at most Q queries to the oracle $\mathcal{O}_{\mathsf{vk}}[\mathsf{pk}]$,* Sim *extracts a valid witness for the statement $\mathsf{St}_\Gamma(\boldsymbol{G}, \boldsymbol{C})$ with probability at least $\varepsilon - (Q+1)\beta/p_k$, where p_k is the smallest prime factor of k.*

The proof describes an efficient simulator Sim that correctly emulates the verifier, without knowing $\mathsf{vk} \bmod k$. The simulation is done as follows:

- $\mathsf{Sim.Setup}(1^\lambda)$: compute $(\mathsf{ek}, \mathsf{dk}) \xleftarrow{\$} S.\mathsf{KeyGen}(1^\lambda)$. Output $\mathsf{crs} \leftarrow \mathsf{ek}$. The encryption key ek defines a plaintext space $\mathbb{Z}_k$ and a random source $\mathbb{Z}_R$ with bound B. Let $\ell \leftarrow 2^\lambda kB$.
- $\mathsf{Sim.KeyGen}(1^\lambda)$: compute $(\mathsf{pk}, \mathsf{vk}) \xleftarrow{\$} \Pi_{\mathsf{K}}.\mathsf{KeyGen}(1^\lambda)$, output pk, store $e_R \leftarrow \mathsf{vk} \bmod R$, and erase vk.
- $\mathsf{Sim.Verify}(\mathsf{pk}, \mathsf{dk}, e_R, \boldsymbol{C}, \boldsymbol{\pi})$: parse $\boldsymbol{\pi}$ as $(\boldsymbol{X}, \boldsymbol{X'}, \boldsymbol{C'})$. Using the secret key dk of S, decrypt $\boldsymbol{X}$ to a vector $\boldsymbol{x}$, and $\boldsymbol{X'}$ to a vector $\boldsymbol{x'}$. Check that $(-e_R) \odot (\boldsymbol{X} \ominus \boldsymbol{x}) = \boldsymbol{X'} \ominus \boldsymbol{x'}$. Check that $\boldsymbol{x} \bullet \Gamma(\boldsymbol{G}, \boldsymbol{C}) = \boldsymbol{C}$, and that $\boldsymbol{x'} \bullet \Gamma(\boldsymbol{G}, \boldsymbol{C}) = \boldsymbol{C'}$. If all checks succeeded, accept. Otherwise, reject.

The simulator Sim first calls $\mathsf{Sim.Setup}(1^\lambda)$ to generate the common reference string (note that our simulator generates the common reference string honestly, hence the simulation of Setup cannot be distinguished from an honest run of Setup), and stores dk. Each time the adversary $\mathscr{A}$ sends a query $(\boldsymbol{C}, \boldsymbol{\pi})$ to the oracle $\mathcal{O}_{\mathsf{vk}}[\mathsf{pk}]$, Sim simulates $\mathcal{O}_{\mathsf{vk}}[\mathsf{pk}]$ (without knowing $\mathsf{vk} \bmod k$) by running $\mathsf{Sim.Verify}(\mathsf{pk}, \mathsf{dk}, e_R, \boldsymbol{C}, \boldsymbol{\pi})$, and accepts or rejects accordingly. When $\mathscr{A}$ outputs a final answer $(\boldsymbol{C}, \boldsymbol{\pi})$, Sim computes a witness $\boldsymbol{x}$ for $\mathsf{St}_\Gamma(\boldsymbol{G}, \boldsymbol{C})$ by decrypting $\boldsymbol{C}$ with dk.

Observe that the distribution $\{(\mathsf{pk}, \mathsf{vk}) \xleftarrow{\$} \Pi_{\mathsf{K}}.\mathsf{KeyGen}(1^\lambda), e_k \leftarrow \mathsf{vk} \bmod k : (\mathsf{pk}, e_k)\}$ is statistically indistinguishable from the distribution $\{(\mathsf{pk}, \mathsf{vk}) \xleftarrow{\$} \Pi_{\mathsf{K}}.\mathsf{KeyGen}(1^\lambda), e_k \xleftarrow{\$} \mathbb{Z}_k : (\mathsf{pk}, e_k)\}$. Put otherwise, the distribution of $\mathsf{vk} \bmod k$ is statistically indistinguishable from random, even given pk. Indeed, as S is a DVNIZK-friendly encryption scheme, it holds by definition that $\gcd(k, R) = 1$. As $\ell = 2^\lambda Bk \geq 2^\lambda Rk$, the distribution $\{e \xleftarrow{\$} \mathbb{Z}_\ell, e_k \leftarrow e \bmod k, e_R \leftarrow e \bmod R : (e_k, e_R)\}$ is statistically indistinguishable from the uniform distribution over $\mathbb{Z}_k \times \mathbb{Z}_R$, and the value pk only leaks e_R, even to an unbounded adversary (as $S.\mathsf{Enc}_{\mathsf{ek}}(0; e) = S.\mathsf{Enc}_{\mathsf{ek}}(0; e \bmod R)$). We now prove the following claim:

Claim. For any public parameters $\boldsymbol{G}$ and word $\boldsymbol{C}$, it holds that

$$\Pr\left[\begin{array}{l} (\mathsf{pk}, \mathsf{vk}) \xleftarrow{\$} \Pi_{\mathsf{K}}.\mathsf{KeyGen}(1^\lambda), \\ b \leftarrow \mathsf{Sim.Verify}(\mathsf{pk}, \mathsf{dk}, \boldsymbol{C}, \boldsymbol{\pi}), : b' = b \\ b' \leftarrow \Pi_{\mathsf{K}}.\mathsf{Verify}(\mathsf{pk}, \mathsf{vk}, \boldsymbol{C}, \boldsymbol{\pi}) \end{array}\right] \geq 1 - \beta/p_k,$$

where p_k is one of the prime factors of k.

Proof. First, we show that if $b = 1$, then $b' = 1$. Indeed, let us denote $(\boldsymbol{x}, \boldsymbol{x'})$ the plaintexts associated to $(\boldsymbol{X}, \boldsymbol{X'})$. Let $(\boldsymbol{r}, \boldsymbol{r'})$ be the random coins of the ciphertexts $(\boldsymbol{X}, \boldsymbol{X'})$. Observe that, by the homomorphic properties of S, the equation $(-e_R) \odot (\boldsymbol{X} \ominus \boldsymbol{x}) = \boldsymbol{X'} \ominus \boldsymbol{x'}$ is equivalent to $S.\mathsf{Enc}_{\mathsf{ek}}(0; -e_R \cdot \boldsymbol{r}) = S.\mathsf{Enc}_{\mathsf{ek}}(0; \boldsymbol{r'})$, which is equivalent to $e \odot \boldsymbol{X} \oplus \boldsymbol{X'} = S.\mathsf{Enc}(e \cdot \boldsymbol{x} + \boldsymbol{x'} \bmod k; e \cdot \boldsymbol{r} + \boldsymbol{r'} \bmod R) = S.\mathsf{Enc}(e \cdot \boldsymbol{x} + \boldsymbol{x'} \bmod k; 0)$ as $e = e_R \bmod R$. Therefore, the verifier's check that $e \odot \boldsymbol{X} \oplus \boldsymbol{X'}$ is decodable succeeds if and only if Sim's first check succeeds, and the decoded value $\boldsymbol{d} \in \mathbb{Z}_k^\gamma$ satisfies $\boldsymbol{d} = e \cdot \boldsymbol{x} + \boldsymbol{x'} \bmod k$. Moreover, if the equations $\boldsymbol{x} \bullet \Gamma(\boldsymbol{G}, \boldsymbol{C}) = \boldsymbol{C}$ and $\boldsymbol{x'} \bullet \Gamma(\boldsymbol{G}, \boldsymbol{C}) = \boldsymbol{C'}$ are both satisfied (*i.e.* Sim's other checks succeed), then it necessarily holds that $\boldsymbol{d} \bullet \Gamma(\boldsymbol{G}, \boldsymbol{C}) = (e \cdot \boldsymbol{x} + \boldsymbol{x'}) \bullet \Gamma(\boldsymbol{G}, \boldsymbol{C}) = e \bullet (\boldsymbol{x} \bullet \Gamma(\boldsymbol{G}, \boldsymbol{C})) \boldsymbol{+} \boldsymbol{x'} \bullet \Gamma(\boldsymbol{G}, \boldsymbol{C}) = e \bullet \boldsymbol{C} \boldsymbol{+} \boldsymbol{C'}$. This concludes the proof that, conditioned on Sim's checks succeeding, the verifier's checks necessarily succeed.

Now, assume for the sake of contradiction that the converse is not true: suppose that Sim rejected the proof, while the verifier accepted. We already showed that the equation $(-e_R) \odot (\boldsymbol{X} \ominus \boldsymbol{x}) = \boldsymbol{X'} \ominus \boldsymbol{x'}$ is equivalent to the equation $e \odot \boldsymbol{X} \oplus \boldsymbol{X'} = S.\mathsf{Enc}(e \cdot \boldsymbol{x} + \boldsymbol{x'} \bmod k; 0)$; therefore, if $e \odot \boldsymbol{X} \oplus \boldsymbol{X'}$ is decodable (it has random coin 0), then Sim's check that $(-e_R) \odot (\boldsymbol{X} \ominus \boldsymbol{x}) = \boldsymbol{X'} \ominus \boldsymbol{x'}$ succeeds. As we assumed that Sim rejects the proof, this means that at least one of Sim's last checks must fail: either $\boldsymbol{x} \bullet \Gamma(\boldsymbol{G}, \boldsymbol{C}) \neq \boldsymbol{C}$, or $\boldsymbol{x'} \bullet \Gamma(\boldsymbol{G}, \boldsymbol{C}) \neq \boldsymbol{C'}$. By the first check of the verifier, it holds that $e \odot \boldsymbol{X} \oplus \boldsymbol{X'}$ is decodable; denoting $(\boldsymbol{x}, \boldsymbol{x'})$ the plaintexts associated to $(\boldsymbol{X}, \boldsymbol{X'})$, it therefore decodes to $\boldsymbol{d} = e \cdot \boldsymbol{x} + \boldsymbol{x'} \bmod k$. By the second check of the verifier, it holds that $\boldsymbol{d} \bullet \Gamma(\boldsymbol{G}, \boldsymbol{C}) = e \bullet \boldsymbol{C} \boldsymbol{+} \boldsymbol{C'}$, which implies $e \bullet (\boldsymbol{x} \bullet \Gamma(\boldsymbol{G}, \boldsymbol{C})) \boldsymbol{+} \boldsymbol{x'} \bullet \Gamma(\boldsymbol{G}, \boldsymbol{C}) = e \bullet \boldsymbol{C} \boldsymbol{+} \boldsymbol{C'}$. This last equation rewrites to

$$e \bullet (\boldsymbol{x} \bullet \Gamma(\boldsymbol{G}, \boldsymbol{C}) \boldsymbol{-} \boldsymbol{C}) = \boldsymbol{C'} \boldsymbol{-} \boldsymbol{x'} \bullet \Gamma(\boldsymbol{G}, \boldsymbol{C}) \tag{1}$$

Now, recall that by assumption, either $\boldsymbol{x} \bullet \Gamma(\boldsymbol{G}, \boldsymbol{C}) \neq \boldsymbol{C}$, or $\boldsymbol{x'} \bullet \Gamma(\boldsymbol{G}, \boldsymbol{C}) \neq \boldsymbol{C'}$. Observe that Equation 1 further implies, as $e \neq 0$ (with overwhelming probability), that $\boldsymbol{x'} \bullet \Gamma(\boldsymbol{G}, \boldsymbol{C}) \boldsymbol{-} \boldsymbol{C'} \neq 0$ if and only if $\boldsymbol{x} \bullet \Gamma(\boldsymbol{G}, \boldsymbol{C}) \boldsymbol{-} \boldsymbol{C} \neq 0$. Therefore, conditioned on Sim rejecting the proof, it necessarily holds that $\boldsymbol{x} \bullet \Gamma(\boldsymbol{G}, \boldsymbol{C}) \boldsymbol{-} \boldsymbol{C} \neq 0$ and $\boldsymbol{x'} \bullet \Gamma(\boldsymbol{G}, \boldsymbol{C}) \boldsymbol{-} \boldsymbol{C'} \neq 0$. Let (μ_i, ν_i) be two non-zero entries of the vectors $(\boldsymbol{x} \bullet \Gamma(\boldsymbol{G}, \boldsymbol{C}) \boldsymbol{-} \boldsymbol{C}, \boldsymbol{C'} \boldsymbol{-} \boldsymbol{x'} \bullet \Gamma(\boldsymbol{G}, \boldsymbol{C}))$ at the same position $i \leq \beta$; by Equation 1, it holds that $e = \nu_i \cdot \mu_i^{-1} \bmod p$ for at least one of the prime factors p of k. However, recall that the value $e \bmod k$ is *statistically hidden* to the prover (and therefore, so is the value $e \bmod p$), hence the probability of this event happening can be upper-bounded by $\beta/p \leq \beta/p_k$. This concludes the proof of the claim. □

Now, consider an adversary $\mathscr{A}$ that outputs an accepting proof $(\boldsymbol{C}, \boldsymbol{\pi})$ with probability at least ε after a polynomial number Q of interactions with the oracle $\mathcal{O}_{\mathsf{vk}}[\mathsf{pk}]$. By the above claim and a union bound, it necessarily holds that $\mathscr{A}$ outputs an accepting proof $(\boldsymbol{C}, \boldsymbol{\pi})$ with probability at least $\varepsilon - Q\beta/p_k$ after interacting Q times with $\mathsf{Sim.Verify}(\mathsf{pk}, \mathsf{dk}, e_R, \cdot, \cdot)$; moreover, with probability at least $1 - \beta p_k$, this proof is also accepted by Sim's verification algorithm. Overall, Sim obtains a proof accepted by his verification algorithm with probability at least $\varepsilon - (Q + 1)\beta/p_k$. In particular, this implies that the vector $\boldsymbol{x}$ extracted by Sim from $\boldsymbol{\pi}$ satisfies $\boldsymbol{x} \bullet \Gamma(\boldsymbol{G}, \boldsymbol{C}) = \boldsymbol{C}$ with probability at least $\varepsilon - (Q + 1)\beta/p_k$. Therefore, Sim extracts a valid witness for the knowledge statement $\mathsf{St}_\Gamma(\boldsymbol{G}, \boldsymbol{C})$ with probability at least $\varepsilon - (Q+1)\beta/p_k$. As the size k of a DVNIZK-friendly cryptosystem has only superpolynomially large prime-factors, it holds that p_k is superpolynomially large. As $(Q + 1)\beta$ is polynomial, we conclude that if $\mathscr{A}$ outputs an accepting proof with non-negligible probability, then Sim extracts a valid witness with non-negligible probability.

4 Dual Variant of the Framework

In the previous section, we described a framework for constructing efficient $\mathsf{DVNIZKs}$ of knowledge for relations between words defined over an abelian group $(\mathbb{G}, \boldsymbol{+})$, using a cryptosystem with specific properties as the underlying commitment scheme for the proof system. In this section, we show that the framework can also be used in a dual way, by considering languages of relations between the plaintexts of the underlying encryption scheme – we call this variant 'dual variant' of the framework, as the roles of the underlying encryption scheme (which is used as a commitment scheme for the proof) and of the abelian group (which contains the words on which the proof is made) are partially exchanged. This allows for example to handle languages of relations between Paillier ciphertexts. To instantiate the framework, it suffices to have any perfectly binding commitment scheme defined over $\mathbb{G}$. This dual variant leads to efficient DVNIZK proofs for relations between, e.g., Paillier ciphertexts, whose zero-knowledge property reduces to the binding property of the commitment scheme over $\mathbb{G}$ (e.g. the DDH assumption, or its variants), and with statistical (unbounded, adaptive) soundness.

4.1 Perfectly Binding Commitment over $\mathbb{G}$

Suppose that we are given a perfectly binding homomorphic commitment $C = (C.\mathsf{Setup}, C.\mathsf{Com}, C.\mathsf{Verify})$, where $C.\mathsf{Com} : \mathbb{Z}_k \times \mathbb{Z}_k \mapsto \mathbb{G}^*$. Assume further that $C.\mathsf{Setup}$ generates a public vector of parameters $\boldsymbol{G} \in \mathbb{G}^*$, and that there is a linear map Γ_C associated to this commitment such that for all $(m, r) \in \mathbb{Z}_k^2$, $C.\mathsf{Com}(m, r) = (m, r) \bullet \Gamma_C(\boldsymbol{G})$. Note this implies the commitment scheme is homomorphic over $\mathbb{G}$. ElGamal (Sect. 2.2), can be used as a commitment scheme satisfying these properties, is hiding under the DDH assumption and perfectly binding. We do so by using $\mathsf{KeyGen}(1^\lambda)$ in place of $\mathsf{Setup}(1^\lambda)$ to generate group elements (G, H) (the public key of the encryption scheme), and commit (i.e encrypt) via $\Gamma_C(G, H) = ((0, G)^\intercal, (G, H)^\intercal)$. We generalize this to commitments to length-t vectors as follow: we let $\Gamma_{C,t}$ denote the extended matrix such that $C.\mathsf{Com}(\boldsymbol{m}, \boldsymbol{r}) = (\boldsymbol{m}, \boldsymbol{r}) \bullet \Gamma_{C,t}(\boldsymbol{G})$, where $(\boldsymbol{m}, \boldsymbol{r})$ are vectors of length t ($\Gamma_{C,t}$ is simply the block-diagonal matrix whose t blocks are all equal to Γ_C). Consider now the following statement, where the word is a vector $\boldsymbol{C}$ of commitments:

$$\begin{aligned}\mathsf{St}_{\Gamma_{C,t}}(\boldsymbol{G}, \boldsymbol{C}) &= \text{Ʞ}\{(\boldsymbol{m}, \boldsymbol{r}) \mid (\boldsymbol{m}, \boldsymbol{r}) \bullet \Gamma_{C,t}(\boldsymbol{G}) = \boldsymbol{C}\} \\ &= \text{Ʞ}\{(\boldsymbol{m}, \boldsymbol{r}) \mid C.\mathsf{Com}(\boldsymbol{m}, \boldsymbol{r}) = \boldsymbol{C}\}.\end{aligned}$$

One can immediatly observe that this statement (which is a proof of knowledge of openings to a vector of commitments with C) is handled by the framework of Section 3.

4.2 Equality of Plaintexts between $\boldsymbol{C}$ and $\boldsymbol{S}$

In this section, we describe a simple method to convert a DVNIZK on the statement $\mathsf{St}_{\Gamma_{C,t}}(\boldsymbol{G}, \boldsymbol{C}) = \text{Ʞ}\{(\boldsymbol{m}, \boldsymbol{r}) \mid C.\mathsf{Com}(\boldsymbol{m}, \boldsymbol{r}) = \boldsymbol{C}\}$ into a DVNIZK on the statement $\mathsf{St}'(\boldsymbol{G}, \boldsymbol{C}, \boldsymbol{X_m}) = \exists\{(\boldsymbol{m}, \boldsymbol{\rho_m}, \boldsymbol{r}) \mid \boldsymbol{X_m} = S.\mathsf{Enc}_{\mathsf{ek}}(\boldsymbol{m}, \boldsymbol{\rho_m}) \wedge \boldsymbol{C} = C.\mathsf{Com}(\boldsymbol{m}, \boldsymbol{r})\}$ for a length-t vector $\boldsymbol{C}$ of commitments with a commitment scheme over $\mathbb{G}$ satisfying the requirements defined in the previous section, and a length-t vector of DVNIZK-friendly ciphertexts $\boldsymbol{X_m}$. Instantiating the framework of Section 3 for the statement $\mathsf{St}_{\Gamma_{C,t}}(\boldsymbol{G}, \boldsymbol{C})$, we get the following DVNIZK Π:

- $\Pi.\mathsf{Setup}(1^\lambda)$: compute $(\mathsf{ek}, \mathsf{dk}) \xleftarrow{\$} S.\mathsf{KeyGen}(1^\lambda)$. Output $\mathsf{crs} \leftarrow \mathsf{ek}$. Note that ek defines the plaintext space $\mathbb{Z}_k$ and the random source $\mathbb{Z}_R$ with bound B. We denote $\ell \leftarrow 2^\lambda k B$.
- $\Pi.\mathsf{KeyGen}(1^\lambda)$: pick $e \leftarrow \mathbb{Z}_\ell$, set $\mathsf{pk} \leftarrow S.\mathsf{Enc}_{\mathsf{ek}}(0; e)$ and $\mathsf{vk} \leftarrow e$.
- $\Pi.\mathsf{Prove}(\mathsf{pk}, \boldsymbol{C}, (\boldsymbol{m}, \boldsymbol{r}))$: on a word $\boldsymbol{C} \in \mathbb{Z}_k^t$, with witness $(\boldsymbol{m}, \boldsymbol{r})$ for the statement $\mathsf{St}_{\Gamma_{C,t}}(\boldsymbol{G}, \boldsymbol{C})$ (where $\boldsymbol{G} \xleftarrow{\$} C.\mathsf{Setup}(1^\lambda)$), pick random $(\boldsymbol{m'}, \boldsymbol{r'})$, random coins $(\boldsymbol{\rho_m}, \boldsymbol{\rho_r})$

for S, and compute

$$\begin{array}{ll} \boldsymbol{X_m} \leftarrow S.\mathsf{Enc}_{\mathsf{ek}}(\boldsymbol{m}, \boldsymbol{\rho_m}), & \boldsymbol{X_r} \leftarrow S.\mathsf{Enc}_{\mathsf{ek}}(\boldsymbol{r}, \boldsymbol{\rho_r}), \\ \boldsymbol{X'_m} \leftarrow S.\mathsf{Enc}_{\mathsf{ek}}(\boldsymbol{m'}, 0) \ominus (\boldsymbol{\rho_m} \odot \mathsf{pk}), & \boldsymbol{X'_r} \leftarrow S.\mathsf{Enc}_{\mathsf{ek}}(\boldsymbol{r'}, 0) \ominus (\boldsymbol{\rho_r} \odot \mathsf{pk}), \\ \boldsymbol{C'} \leftarrow (\boldsymbol{m'}, \boldsymbol{r'}) \bullet \Gamma_{C,t}(\boldsymbol{G}, \boldsymbol{C}), & \end{array}$$

and output $\boldsymbol{\pi} \leftarrow (\boldsymbol{X_m}, \boldsymbol{X'_m}, \boldsymbol{X_r}, \boldsymbol{X'_r}, \boldsymbol{C'})$.

- $\Pi_{\mathsf{K}}.\mathsf{Verify}(\mathsf{pk}, \mathsf{vk}, \boldsymbol{C}, \boldsymbol{\pi})$: parse $\boldsymbol{\pi}$ as $(\boldsymbol{X_m}, \boldsymbol{X'_m}, \boldsymbol{X_r}, \boldsymbol{X'_r}, \boldsymbol{C'})$. Check that $e \odot \boldsymbol{X_m} \oplus \boldsymbol{X'_m}$ and $e \odot \boldsymbol{X_r} \oplus \boldsymbol{X'_r}$ are decodable, and decode them to vectors $(\boldsymbol{d_m}, \boldsymbol{d_r}) \in (\mathbb{Z}_k^t)^2$. Check that $(\boldsymbol{d_m}, \boldsymbol{d_r}) \bullet \Gamma_{C,t}(\boldsymbol{G}, \boldsymbol{C}) = e \bullet \boldsymbol{C} + \boldsymbol{C'}$.

By the result of Section 3, this is an unbounded statistical adaptive knowledge-extractable DVNIZK of knowledge of an opening for C. Suppose now that we modify the above scheme as follow: we let $\boldsymbol{X_m}$ be part of the *word* on which the proof is executed, rather than being computed as part of the *proof* by the algorithm $\Pi.\mathsf{Prove}$. That is, we consider words of the form $(\boldsymbol{C}, \boldsymbol{X_m})$ with witness $(\boldsymbol{m}, \boldsymbol{r}, \boldsymbol{\rho_m})$ such that $(\boldsymbol{C}, \boldsymbol{X_m}) = (C.\mathsf{Com}(\boldsymbol{m}; \boldsymbol{r}), S.\mathsf{Enc}_{\mathsf{ek}}(\boldsymbol{m}, \boldsymbol{\rho_m}))$. Let Π' denote the modified proof, in which $\boldsymbol{X_m}$ is part of the word and $(\boldsymbol{X'_m}, \boldsymbol{X_r}, \boldsymbol{X'_r}, \boldsymbol{C'})$ are computed as in Π. Observe that the proof of security of our framework immediatly implies that Π' is a secure DVNIZK for *plaintext equality* between commitments with C and encryptions with S: our statistical argument shows that a (possibly unbounded) adversary has negligible probability of outputting a word $\boldsymbol{C}$ together with an accepting proof $\boldsymbol{\pi} = (\boldsymbol{X_m}, \boldsymbol{X'_m}, \boldsymbol{X_r}, \boldsymbol{X'_r}, \boldsymbol{C'})$ where the plaintext extracted by the simulator from $\boldsymbol{X_m}$ is not also the plaintext of $\boldsymbol{C}$. Hence, it is trivial that the probability of outputting a word $(\boldsymbol{C}, \boldsymbol{X_m})$ and an accepting proof $\boldsymbol{\pi'} = (\boldsymbol{X'_m}, \boldsymbol{X_r}, \boldsymbol{X'_r}, \boldsymbol{C'})$ where the plaintext extracted by the simulator from $\boldsymbol{X_m}$ is not also the plaintext of $\boldsymbol{C}$ is also negligible. Thus, we get:

Theorem 16. *The proof system Π' is an adaptive unbounded statistically sound proof for equality between plaintexts of C and plaintexts of S, whose composable zero-knowledge property reduces to the* $\mathsf{IND\text{-}CPA}$ *security of S.*

Note that the proof Π' is no longer a proof of knowledge: while the simulator can extract $(\boldsymbol{m}, \boldsymbol{r})$ from the prover, he cannot necessarily extract the random coins $\boldsymbol{\rho_m}$ of $\boldsymbol{X_m}$, which are now part of the witness. Therefore, for the protocol to make sense, it is important that C is a perfectly binding commitment scheme.

4.3 A Framework for Relations between Plaintexts of S

The observations of the above section suggest a very natural way for designing DVNIZKs for relations between plaintexts $\boldsymbol{m} \in \mathbb{Z}_k^*$ of the encryption scheme S, which intuitively operates in two steps: first, we create commitments to the plaintexts $\boldsymbol{m}$ over $\mathbb{G}$ using C and prove them consistent with the encrypted values using the method described in the previous section. Then, we are able to use the framework of Section 3 to demonstrate the desired relation holds between the commited values (this is a statement naturally captured by the framework). More formally, on input a vector of ciphertexts $\boldsymbol{X_m}$ encrypting plaintexts $\boldsymbol{m}$ with random coins $\boldsymbol{\rho_m}$,

- Pick $\boldsymbol{r}$ and compute $\boldsymbol{C} \leftarrow C.\mathsf{Com}(\boldsymbol{m}, \boldsymbol{r})$.
- Construct a DVNIZK for the statement $\mathsf{St}'(\boldsymbol{G}, \boldsymbol{C}, \boldsymbol{X_m})$ with witness $(\boldsymbol{m}, \boldsymbol{\rho_m}, \boldsymbol{r})$, using the method described in Section 4.2.
- Construct a DVNIZK for the statement $\mathsf{St}_\Gamma(\boldsymbol{G}, \boldsymbol{C})$ with witness $(\boldsymbol{m}, \boldsymbol{r})$, using the framework of Section 3.

The correctness of this approach is immediate: the second DVNIZK guarantees that the appropriate relation is satisfied between the plaintexts of the commitments, while the first one guarantees that the ciphertexts indeed encrypt the committed values. This leads to a DVNIZK proof of relation between plaintexts of S, with unbounded adaptive statistical soundness. Regarding zero-knowledge, as the proof starts by committing to $\boldsymbol{m}$ with C, we must in addition assume that the commitment scheme is hiding (we skip the details of the security analysis, noting that it is straightforward).

Theorem 17. *The above proof system is an adaptive unbounded statistically sound proof for relations between plaintexts of S, whose composable zero-knowledge property reduces to the* IND-CPA *security of S and the hiding property of C.*

We note that we can also obtain a variant of Theorem 17, where zero-knowledge only relies on the IND-CPA of S, and hiding of C implies the soundness property, using commitment schemes *a la* Groth-Sahai where the crs can be generated in two indistinguishable ways, one leading to a perfectly hiding scheme, and one leading to a perfectly binding scheme (such commitments are known, e.g., from the DDH assumption).

Example: Multiplicative Relationship Between Paillier Ciphertexts. We focus now on the useful case of multiplicative relationship between plaintexts of Paillier ciphertexts. We instantiate S with the Paillier encryption scheme over an RSA group $\mathbb{Z}_n$, with a public key (n, h) ($h = g^n \bmod n^2$ for a generator g of $\mathbb{J}_n$), and the commitment scheme C with the ElGamal encryption scheme over a group $\mathbb{G}$ of order n, with public key (G, H). Let $(P_0, P_1, P_2) \in (\mathbb{Z}^*_{n^2})^3$ be three Paillier ciphertexts, and let $(m_0, m_1, m_2, \rho_0, \rho_1, \rho_2)$ be such that $m_2 = m_0 m_1 \bmod n$, and $P_0 = (1+n)^{m_0} h^{\rho_0} \bmod n^2, P_1 = (1+n)^{m_1} h^{\rho_1} \bmod n^2, P_2 = (1+n)^{m_2} h^{\rho_2} \bmod n^2$. Let $E = h^e \bmod n^2$ denote the public key of the verifier. The designated-verifier NIZK for proving that P_2 encrypts $m_0 m_1$ proceeds as follows:

- **Committing over** $\mathbb{G}$**:** pick (r_0, r_1, r_2) and send $(U_i, V_i)_{0 \le i \le 2} \leftarrow (r_i \bullet G, r_i \bullet H {+} m_i \bullet G)_{0 \le i \le 2}$ (which are commitments with ElGamal to (m_0, m_1, m_2) over $\mathbb{G}$).
- **Proof of Plaintext Equality:** pick $(m'_i, r'_i, \rho'_i)_{0 \le i \le 2} \xleftarrow{\$} (\mathbb{Z}_n \times \mathbb{Z}_n \times \mathbb{Z}_{n/2})^3$, and send for $i = 0$ to 2, $X_i \leftarrow (1+n)^{r_i} h^{\rho'_i} \bmod n^2, X'_i \leftarrow (1+n)^{r'_i} E^{-\rho'_i} \bmod n^2, P'_i \leftarrow (1+n)^{m'_i} E^{-\rho_i} \bmod n^2$, and $(U'_i, V'_i) \leftarrow (r'_i \bullet G, r'_i \bullet H {+} m'_i \bullet G)$.
- **Proof of Multiplicative Relationship Between the Committed Values:** apply the proof system of Example 2 from Section 3 to the word $(U_i, V_i)_{0 \le i \le 2}$, with public parameters (G, H), and the witness $\boldsymbol{x} = (m_0, r_0, m_1, r_1, r_2 - r_0 m_1)$ which satisfies $(U_0, V_0) = (r_0 \bullet G, r_0 \bullet H {+} m_0 \bullet G), (U_1, V_1) = (r_1 \bullet G, r_1 \bullet H {+} m_1 \bullet G)$, and $(U_2, V_2) = ((r_2 - r_0 m_1) \bullet G {+} m_1 \bullet U_0, (r_2 - r_0 m_1) \bullet H {+} m_1 \bullet V_0)$.
- **Proof Verification:** upon receving $(U_i, V_i, X_i, X'_i, P'_i, U'_i, V'_i)_{0 \le i \le 2}$ together with the proof of multiplicative relationship between the values committed with $(U_i, V_i)_i$, the verifier with verification key $\mathsf{vk} = e$ checks that $e \odot P_i \oplus P'_i$ and $e \odot X_i \oplus X'_i$ successfully decode (respectively) to values p_i, x_i, and that $e \bullet U_i {+} U'_i = x_i \bullet G$ and $e \bullet V_i {+} V'_i = x_i \bullet H {+} p_i \bullet G$, for $i = 0$ to 2. The verifier additionally checks the multiplicative proof, as in Example 4 from Section 3. She accepts iff all checks succeed.

The proof for the multiplicative statement involves 10 Paillier ciphertexts and 3 ElGamal ciphertexts. Overall, the total proof involves 20 Paillier ciphertexts, and 9 ElGamal ciphertexts. However, this size is obtained by applying the framework naively; in this situation, it introduces a lot of redudancy. For instance, instead of computing Paillier encryptions of (m_0, r_0, m_1, r_1) in the third phase, one can simply reuse the word (P_0, P_1) and the ciphertexts (X_0, X_1), as well as reusing $(P'_i, X'_i)_i$ for the corresponding masks $(m'_i, r'_i)_i$, saving 8 Paillier ciphertexts; similar savings can be obtained for the ElGamal ciphertexts, leading to a proof of total size 12 Paillier ciphertexts + 7 ElGamal ciphertexts.

Furthermore, if we eschew unbounded soundness and accept bounds on m_i we are able to produce a much shorter proof, comprising only two Paillier ciphertexts, outperforming even Fiat-Shamir. We detail this in the next section.

5 Trading Unbounded Soundness for Efficiency

The dual variant of our framework, developed in Section 4, allows to non-interactively prove relations between the plaintexts of a DVNIZK-friendly cryptosystem, such as the Paillier cryptosystem, with strong security guarantees. However, this comes at the cost of somewhat larger proofs than one could hope for: for instance, proving that a Paillier ciphertext encrypts the products of the plaintexts of two other ciphertexts requires 12 Paillier ciphertexts and 7 ElGamal ciphertexts. In some scenarios, one might be willing to sacrifice some of the strong security guarantees provided by the framework, in exchange for an improved efficiency.

In this section, we explain how our techniques can be used in a more direct way, without relying on an intermediate commitment scheme over an abelian group, to prove relationships between plaintexts of a DVNIZK-friendly scheme. The DVNIZK we obtain is statistically zero-knowledge, computationally sound, but does not enjoy unbounded soundness. It is, however extremely efficient: proving a multiplicative relationship between three ciphertexts requires only two ciphertexts. This leads to proofs that are shorter than *any* known alternative for building non-interactive zero-knowledge proofs, almost twice shorter than the proofs in the random oracle model obtained via the Fiat-Shamir transform.

5.1 Coin-Unextractability

Our construction relies on a DVNIZK-friendly cryptosystem, satisfying an additional security requirement: intuitively, it must be infeasible, given a ciphertext, to extract the random coin used to generate it, even given the decryption key of the scheme. More formally,

Definition 18 ($\mathcal{D}$-Coin-Unextractability). *A DVNIZK-friendly cryptosystem S is $\mathcal{D}$-coin-unextractable, for some family of distributions $\mathcal{D} = (\mathcal{D}_{\mathsf{ek}})_{\mathsf{ek}}$ over the random source of S, if for any pair $(\mathsf{ek}, \mathsf{dk}) \xleftarrow{\$} S.\mathsf{KeyGen}(1^\lambda)$, and any PPT adversary $\mathscr{A}$, it holds that*

$$\Pr[r \xleftarrow{\$} \mathcal{D}_{\mathsf{ek}}, c \leftarrow S.\mathsf{Enc}(0; r), r' \leftarrow \mathscr{A}(\mathsf{ek}, \mathsf{dk}, c) : r' = r] = \mathrm{negl}(\lambda).$$

Example: The Paillier Encryption Scheme. For simplicity, let us first look at the coin-extractability of Paillier for the uniform distribution. Let n be an RSA modulus. As it leads to a somewhat simpler analysis, we will consider a slight variant of Paillier where $h = g^n$ is built as the n'th power of a random element g of QR_n (rather than $\mathbb{J}_n$ previously). The random source of Paillier is $\mathbb{Z}_{\varphi(n)/4}$. The challenger for the uniform coin-unextractability of Paillier picks $r \xleftarrow{\$} \mathbb{Z}_{\varphi(n)/4}$ and returns a challenge $c = h^r \bmod n^2$. Given c, the goal is to find out r; hence, given the factorization of n, the uniform coin-unextractability of Paillier is exactly the discrete logarithm assumption over QR_n. Note that given the secret key of Paillier (*i.e.*, the factorization of $n = pq = (2p' + 1)(2q' + 1)$ for primes (p, q, p', q')), the discrete logarithm assumption over QR_n reduces to the discrete logarithm assumption over both QR_p and QR_q (the cyclic subgroups of squares of $\mathbb{Z}_p$ and $\mathbb{Z}_q$), via the chinese remainder theorem.

In the next section, we will be interested in the coin-unextractability property of the scheme S for a slightly more complex distribution: the distribution of uniformly random coins smaller than some given bound. Therefore, let us show that the coin-unextractability of Paillier for this distribution reduces to the short-exponent discrete logarithm assumption. Let t be an integer; we will consider random coins picked at random over $\mathbb{Z}_{n/2^t}$ (note that this is statistically indistinguishable from coins picked at random over $\mathbb{Z}_{\varphi(n)/2^t}$ as long as $n/2^t$ is superpolynomially large). The coin-unextractability property of Paillier for the uniform distribution over $\mathbb{Z}_{n/2^t}$ is clearly equivalent to the discrete logarithm assumption over QR_n with exponents picked at random over $\mathbb{Z}_{n/2^t}$, a type of assumption usually called *short-exponent discrete logarithm assumption* (sDL). However, it is not immediate that the sDL assumption over QR_n reduces to the sDL assumption over QR_p or QR_q, even given the factorization of n, because the chinese remainder theorem does not preserve the size of the exponent in general. A slightly more involved analysis shows that this can nevertheless be ensured in this case, hence the coin-unextractability of Paillier with bounded random coins can be reduced to the sDL assumption over a (multiplicative subgroup of a) prime order field. We detail the reduction from sDL over QR_n to sDL over a subgroup of QR_p in the next section.

5.2 Composite-to-Prime Reduction for the Short-Exponent Discrete Logarithm Assumption

We show that the short-exponent discrete logarithm assumption on a composite order group can be reduced to the same assumption on a prime order group. We denote QR_n (resp. QR_p) the subgroup of squares of $\mathbb{Z}_n^*$ (resp. of $\mathbb{Z}_p^*$). We prove that, given an integer t, the short-exponent discrete logarithm assumption over QR_p for a bound $p/2^t \ll p'$ implies the short-exponent discrete logarithm assumption over QR_n for a bound $n/2^{t+1} \ll \varphi(n)$, assuming $|p/2^t| = \omega(\log \lambda)$ (*i.e.*, the set $\mathbb{Z}_{\lceil p/2^t \rceil}$ is superpolynomially large). The reduction assumes that the factorization of n is known. Note that the reduction for the standard discrete logarithm assumption from QR_p to QR_n is immediate by the chinese remainder theorem; however, as the CRT decomposition does not preserve the size of the exponents in general, the argument is more involved for the short-exponent discrete logarithm assumption. Let us denote $\mathsf{sDL}_t(\mathbb{G})$ the short-exponent discrete logarithm assumption over a group $\mathbb{G}$, with exponents bounded by $|\mathbb{G}|/2^t$.

Lemma 19. *Assuming* $|p/2^t| = \omega(\log \lambda)$, $\mathsf{sDL}_t(\mathsf{QR}_p) \implies \mathsf{sDL}_{t+1}(\mathsf{QR}_n)$.

Proof. For a modulus $n = pq$, we denote by $\mathtt{Dist}_n, \mathtt{Dist}_p$ the following distribution:

$$\mathtt{Dist}_n = \{g \xleftarrow{\$} \mathsf{QR}_n, x \xleftarrow{\$} \mathbb{Z}_{\lceil p/2^k \rceil}, h \leftarrow g^x \bmod n \ :\ (g, h)\}$$
$$\mathtt{Dist}_p = \{g \xleftarrow{\$} \mathsf{QR}_p, x \xleftarrow{\$} \mathbb{Z}_{\lceil n/2^{k+1} \rceil}, h \leftarrow g^x \bmod p \ :\ (g, h)\}$$

Let Adv be an adversary that, on input $n \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$ and a random sample (g, h) of $\mathtt{Dist}_n$, outputs an integer x such that $g^x = h$ with non-negligible probability ε. We denote by $\mathsf{Embed}_{p,n}$ the following probabilistic algorithm:

$\mathsf{Embed}(p, g_p, h_p)$**:** On input a random strong prime $p = 2p' + 1$ and a sample (g_p, h_p) from $\mathtt{Dist}_{p,k}$, pick a random strong prime $q = 2q' + 1$ of the same size as p, a random generator $g_q \xleftarrow{\$} \mathsf{QR}_q$ and a random exponent $x_q \xleftarrow{\$} \mathbb{Z}_{q'}$. Set $n \leftarrow pq$ and computes $\mu \leftarrow [(q')^{-1} \bmod p']$. Set $f \leftarrow g_p^{q'\mu} g_q^{1-q'\mu} \bmod n$, $g \leftarrow f^\mu \bmod n$, $h \leftarrow h_p^\mu g^{x_q} \bmod n$, and output (n, g, h, μ, x_q).

We construct an adversary Adv' against the sDL assumption over QR_p as follows:

Adv': On input a random strong prime $p = 2p' + 1$ and a sample (g_p, h_p) from $\mathtt{Dist}_{p,k}$, compute $(n, g, h, \mu, x_q) \overset{\$}{\leftarrow} \mathsf{Embed}(p, g_p, h_p)$. If $x - x_q = 0 \bmod q'$, output $(x - x_q)/q'$; else, output $\perp$.

Claim. $\Pr[(g_p, h_p) \overset{\$}{\leftarrow} \mathtt{Dist}_p, x_p \overset{\$}{\leftarrow} \mathsf{Adv}'(p, g_p, h_p) \ : \ g_p^{x_p} = h_p \bmod p] \geq \varepsilon - \text{negl}(\lambda)$

Toward proving the claim, let $\mathtt{Dist}'_p$ denote the output distribution of $\mathsf{Embed}_{n,p}$ restricted to the three first entries (n, g, h) of the output. We show that $\mathtt{Dist}'_p$ and $\mathtt{Dist}_n$ are statistically indistinguishable; the claim follows. It is clear that n is distributed exactly as in Gen. Observe that $g = g_p^\mu \bmod p$ and $g = g_q^\mu \bmod q$. As (g_p, g_q) are uniformly random generators of $(\mathsf{QR}_p, \mathsf{QR}_q)$, so are (g_p^μ, g_q^μ) (as $\mu \neq 0 \bmod p'$ and $\mu \neq 0 \bmod q'$). As g is the unique value (modulo n) satisfying $g = g_p^\mu \bmod p$ and $g = g_q^\mu \bmod q$, by the chinese remainder theorem, g is a uniformly random generator of QR_n. We now look at the distribution of h. A direct calculation shows that $h = g^{q'z + x_q} \bmod n$ where z is the discrete logarithm of h_p in base g_p (modulo p). By definition of $\mathtt{Dist}_p$, z is uniformly distributed over $\mathbb{Z}_{\lceil p/2^k \rceil}$, and x_q is uniformly distributed over $\mathbb{Z}_{q'}$. Therefore, $q'z + x_q$ is uniformly distributed over $\mathbb{Z}_{q'\lceil p/2^k \rceil}$. To conclude the proof of the claim, it remains to show that $\mathbb{Z}_{q'\lceil p/2^k \rceil}$ is statistically close to $\mathbb{Z}_{\lceil n/2^{k+1} \rceil}$. To prove that, it suffices to show that $q'\lceil p/2^k \rceil / \lceil n/2^k \rceil = 1 + \text{negl}(\lambda)$. Let us write $p = 2^k p_0 + p_1$, with $p_1 < 2^k$. We have

$$q'\lceil p/2^k \rceil = q'p_0, \text{ and}$$
$$\lceil n/2^{k+1} \rceil = \left\lceil \frac{(2^k p_0 + p_1)(2q' + 1)}{2^{k+1}} \right\rceil = p_0 q' + \left\lceil \frac{2p_1 q' + 2^k p_0 + p_1}{2^{k+1}} \right\rceil$$

Moreover, as $2p_1 q' + 2^k p_0 + p_1 \leq 2^{k+1} q' + 2^k p_0 + p_1$, we have

$$\lceil n/2^{k+1} \rceil \leq p_0 q' + q' + p_1 + p_0/2$$

which gives

$$1 \leq \frac{\lceil n/2^{k+1} \rceil}{q'\lceil p/2^{k+1} \rceil} \leq 1 + \frac{q' + p_1 + p_0/2}{p_0 q'} \leq 1 + \frac{1}{p_0} + \frac{2^k}{p_0 q'} + \frac{1}{2q'} \leq 1 + \frac{2}{p_0} + \frac{1}{2q'}.$$

As $|p_0| = |p/2^k| = \omega(\log \lambda)$ and q' is exponentially large, we have that

$$\frac{2}{p_0} + \frac{1}{2q'} = \text{negl}(\lambda)$$

Which concludes the proof of the claim. □

5.3 Efficient Proof of Multiplicative Relationship

Recall that, for a DVNIZK-friendly encryption scheme S, the encryption key ek specifies a bound B on the size of the random source. We consider the following families of distributions: $\mathcal{D}_t = (\mathcal{D}_{t,\mathsf{ek}})_{\mathsf{ek}}$, for integers $t = t(\lambda)$ and encryption keys ek, where each $\mathcal{D}_{t,\mathsf{ek}}$ is the uniform distribution over $\mathbb{Z}_{B/2^t}$. Intuitively, this corresponds to the distribution of *small* uniformly random coins, where t parametrizes the size of the space from which the coins are taken. Let S be a $\mathcal{D}_t$-coin-unextractable DVNIZK-friendly encryption scheme, for some integer $t = t(\lambda)$ which will be specified afterward. Consider the following scheme Π_{prod} for proving multiplicative relations between DVNIZK-friendly encryptions of plaintexts of bounded size:

– $\Pi_{\mathsf{prod}}.\mathsf{Setup}(1^\lambda)$: compute $(\mathsf{ek}, \mathsf{dk}) \overset{\$}{\leftarrow} S.\mathsf{KeyGen}(1^\lambda)$. Output $\mathsf{crs} \leftarrow \mathsf{ek}$. Note that ek defines a plaintext space $\mathbb{Z}_k$ and a random source $\mathbb{Z}_R$ with a bound B. Let p_k be the smallest prime factor of k. Let $t = t(\lambda)$ be an integer such that $B/2^t < \min(p_k, R)$, and such that $t' \leftarrow \lceil t - \lambda + \log_2(k/B)\rceil$ satisfies $t' > 0$. Let $\ell \leftarrow \lceil B/2^t \rceil$.
– $\Pi_{\mathsf{prod}}.\mathsf{KeyGen}(1^\lambda)$: pick $e \leftarrow \mathbb{Z}_\ell$, set $\mathsf{pk} \leftarrow S.\mathsf{Enc}_{\mathsf{ek}}(0; e)$ and $\mathsf{vk} \leftarrow e$.
– $\Pi_{\mathsf{prod}}.\mathsf{Prove}(\mathsf{pk}, \boldsymbol{x}, \boldsymbol{w})$: given a word $\boldsymbol{x} = (c_0, c_1, c_2) = (S.\mathsf{Enc}_{\mathsf{ek}}(m_0; r_0), S.\mathsf{Enc}_{\mathsf{ek}}(m_1; r_1), S.\mathsf{Enc}_{\mathsf{ek}}(m_2; r_2))$, with $m_0 \leq 2^{t'}$, and witness $\boldsymbol{w} = (m_i, r_i)_{i\leq 2}$, pick $m' \overset{\$}{\leftarrow} \mathbb{Z}_k$. Compute

$$X_0 \leftarrow -r_0 \odot \mathsf{pk} \oplus S.\mathsf{Enc}_{\mathsf{ek}}(m'; 0), \qquad X_1 \leftarrow (r_1 m_0 - r_2) \odot \mathsf{pk} \oplus m' \bullet c_1$$

and output $\pi \leftarrow (X_0, X_1)$.
– $\Pi_{\mathsf{prod}}.\mathsf{Verify}(\mathsf{pk}, \mathsf{vk}, x, \pi)$: parse π as (X_0, X_1). Compute $F_0 \leftarrow e \odot c_0 \oplus X_0$ and $F_1 \leftarrow e \odot c_2 \oplus X_1$. Check that F_0 decodes to some d; otherwise, reject. Check that $F_1 = d \odot c_1$. If the check fails, reject. Otherwise, accept.

Theorem 20. *Let S be a $\mathcal{D}_t$-coin-unextractable* DVNIZK*-friendly encryption scheme. Let t be an integer, chosen as above, and let t' be defined as above. The scheme Π_{prod} is a secure* DVNIZK *argument system of multiplicative relationship between plaintexts provided that the size of the first plaintext is bounded by $2^{t'}$, which is statistically zero-knowledge and statisfies bounded-soundness under the $\mathcal{D}_t$-coin-unextractability of S.*

Instantiated with the Paillier encryption scheme, this leads to a DVNIZK proof system for multiplicative relations between Paillier ciphertext, where a proof requires only two Paillier ciphertexts, whose bounded adaptive soundness property reduces to the short-exponent discrete logarithm assumption over the subgroup of squares of $\mathbb{Z}_p^*$.

Comparison With Alternative Approaches. Before proving Theorem 20, let us discuss the efficiency of the proof system Π_{prod}. For simplicity, we focus on the most natural instantiation, with the Paillier encryption scheme. The standard way of building an efficient non-interactive zero-knowledge proof system for multiplicative relations between Paillier ciphertexts is to start from the classical public-coin three-move honest-verifier zero-knowledge proof (also called Σ-protocol) for this statement, and to compile it into a NIZK using the Fiat-Shamir transform.

The Σ-protocol for multiplicative relation between Paillier ciphertext was first introduced in [DJ01]. Let n be an RSA modulus. Adapting this Σ-protocol to the Paillier variant we use here (where the random coin is an exponent), the total communication of the protocol involves $7\log(n)$ bits (ignoring the cost of sending the challenge). Therefore, the NIZK obtained by compiling this Σ-protocol with the Fiat-Shamir transform also requires $7\log(n)$ bits. To our knowledge, this is the most efficient currently known method for proving such statements non-interactively: all previous works on DVNIZKs describe homomorphic-encryption-based compilers for Σ-protocols which lead to DVNIZKs less efficient than the NIZK obtained with Fiat-Shamir (the DFN transform on this statement exchanges $10\log(n) + 18\lambda$ bits, and all subsequent works require at least this amount of communication). We note that zero-knowledge proofs with shorter communication than the standard Σ-protocol for this statement were suggested in [LZ14], but these protocols cannot be made non-interactive in the random oracle model (and require the prover to know the secret key of the Paillier scheme anyway).

The protocol Π_{prod} requires only $4\log(n)$ bits of communication in total, making it almost twice as efficient as the NIZK obtained with the Fiat-Shamir transform. To our knowledge, our protocol is the first example of a non-interactive zero-knowledge argument in the standard model which is more efficient than proofs obtained via the Fiat-Shamir transform in the random oracle model.

5.4 Security Analysis

In this section, we formally prove Theorem 20.

Correctness. Plugging $X_0 = -r_0 \odot \mathsf{pk} \oplus S.\mathsf{Enc}_{\mathsf{ek}}(m';0) = S.\mathsf{Enc}_{\mathsf{ek}}(m'; -r_0 \cdot e)$ into $F_0 = e \odot c_0 \oplus X_0$, with $c_0 = S.\mathsf{Enc}_{\mathsf{ek}}(m_0; r_0)$, we get $F_0 = S.\mathsf{Enc}_{\mathsf{ek}}(e \cdot m_0 + m'; 0)$, which successfully decodes to $d = e \cdot m_0 + m' \bmod k$. Observe that as $m_0 \leq 2^{t'}$, it holds that $e \cdot m_0 \leq \ell \cdot 2^{t'} \leq k/2^\lambda$. As the probability that a random m' belongs to $[k \cdot (1 - 2^{-\lambda}), k]$ is $1/2^\lambda$, which is negligible, with overwhelming probability it must hold that $e \cdot m_0 + m' < k$, hence $d = e \cdot m_0 + m'$ (over the integers).

Plugging $X_1 = (r_1 m_0 - r_2) \odot \mathsf{pk} \oplus m' \bullet c_1 = S.\mathsf{Enc}_{\mathsf{ek}}(m' m_1; (m' + e m_0) r_1 - e r_2)$ in $F_1 = e \odot c_2 \oplus X_1$, with $c_2 = S.\mathsf{Enc}_{\mathsf{ek}}(m_2; r_2)$, we get $F_1 = S.\mathsf{Enc}_{\mathsf{ek}}(e \cdot m_2 + m' m_1; (m' + e m_0) r_1)$. As $m_2 = m_0 m_1$, this reduces to $F_1 = (m' + e \cdot m_0) \odot S.\mathsf{Enc}_{\mathsf{ek}}(m_1; r_1) = d \odot c_1$, which concludes the proof of correctness.

Computational Bounded Soundness. Consider an adversary $\mathscr{A}$ producing (with probability at least ε) an accepting proof (X_0, X_1) on a word $\boldsymbol{x} = (c_0, c_1, c_2)$ encrypting values (m_0, m_1, m_2) such that $m_0 m_1 \neq m_2$. We exhibit a simulator Sim that interacts with $\mathscr{A}$ and breaks the coin-unextractability property of S with the same probability.

The simulator proceeds as follows: first, he executes the setup honestly, producing $(\mathsf{ek}, \mathsf{dk}) \xleftarrow{\$} S.\mathsf{KeyGen}(1^\lambda)$, and stores $(\mathsf{ek}, \mathsf{dk})$. To simulate the key generation phase, he interacts with a challenger for the $\mathcal{D}_t$-coin-unextractability property of S, and receives a challenge ciphertext c whose random coin was drawn uniformly from $\mathbb{Z}_\ell$ (with $\ell = \lceil B/2^t \rceil$, for the parameter t and the bound B defined by ek). He outputs $\mathsf{pk} \leftarrow c$. Note that the simulation is trivially perfectly indistinguishable from an honest execution of the setup and key generation algorithms. Observe also that by construction, c uniquely defines the challenge e over the integers, as $\ell < R$.

When receiving a word $\boldsymbol{x} = (c_0, c_1, c_2)$ and a proof (X_0, X_1) from $\mathscr{A}$, Sim uses dk to decrypt $(c_0, c_1, c_2, X_0, X_1)$ to $(m_0, m_1, m_2, x_0, x_1)$, and compute $e' \leftarrow (x_0 m_1 - x_1)(m_2 - m_0 m_1)^{-1} \bmod p$, where p is any prime factor of k such that $(m_2 - m_0 m_1)$ is invertible modulo p (which necessarily exists since $(m_2 - m_0 m_1) \neq 0 \bmod k$). He sends e' to the challenger. We now show that if (X_0, X_1) is indeed an accepting proof, then Sim correctly answers the coin-unextractability challenge. From the verification equations, we get:

- $d = e \cdot m_0 + x_0 \bmod k$ //the first equation uniquely defines the value d modulo k
- $e \cdot m_2 + x_1 = d \cdot m_1 \bmod k$

combining the two equations, we get $e \cdot (m_2 - m_0 m_1) = x_0 m_1 - x_1 \bmod k$. By assumption, $m_2 - m_0 m_1 \neq 0 \bmod k$, therefore, there exists a prime factor p of k such that $m_2 - m_0 m_1$ is invertible modulo p. This gives $e = (x_0 m_1 - x_1)(m_2 - m_0 m_1)^{-1} \bmod p$, which uniquely determines e modulo p. As $e < \ell$ and ℓ is smaller than the smallest prime factor of k, this uniquely determines e over the integers, hence Sim correctly computes the correct answer to the coin-unextractability challenge with ciphertext c.

Statistical Composable Zero-Knowledge. We exhibit a simulator who, given vk, output proofs which are statistically indistinguishable from honest proofs on true statements. The simulator Sim is given $(\mathsf{pk}, \mathsf{vk} = e)$ and a word (c_0, c_1, c_2). Sim picks $d \xleftarrow{\$} \mathbb{Z}_k$ and computes

$$X_0 \leftarrow S.\mathsf{Enc}_{\mathsf{ek}}(d; 0) \ominus (e \odot c_0), \qquad X_1 \leftarrow (d \odot c_1) \ominus (e \odot c_2).$$

First, observe that the value d picked by the simulator follows a distribution statistically close from the value decoded by the verifier when interacting with an honest prover on a true statement: the decoded value with an honest prover is $e \cdot m_0 + m'$, where $e \cdot m_0 \leq \ell \cdot 2^{t'} \leq k/2^\lambda$, and m' is uniform over $\mathbb{Z}_k$. Therefore, the distribution of $e \cdot m_0 + m'$ is statistically indistinguishable from uniform over $\mathbb{Z}_k$. As (X_0, X_1) are exactly the unique pair of ciphertexts satisfying the verification equation with respect to the decoded value d, the proof follows.

Acknowledgements

We thank Jens Groth for insightful discussions and contributions to early versions of this work. The first author was supported by EU Horizon 2020 grant 653497 (project PANORAMIX). The second author was supported by ERC grant 339563 (project Crypto-Cloud) and ERC grant 724307 (project PREP-CRYPTO).

References

ABP15. M. Abdalla, F. Benhamouda, and D. Pointcheval. Disjunctions for hash proof systems: New constructions and applications. LNCS, pages 69–100. Springer, 2015.

BBC⁺13. F. Benhamouda, O. Blazy, C. Chevalier, D. Pointcheval, and D. Vergnaud. New techniques for SPHFs and efficient one-round PAKE protocols. In *CRYPTO 2013, Part I*, *LNCS* 8042, pages 449–475. Springer, August 2013.

BBS04. D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In *CRYPTO 2004*, *LNCS* 3152, pages 41–55. Springer, August 2004.

BCC⁺09. M. Belenkiy, J. Camenisch, M. Chase, M. Kohlweiss, A. Lysyanskaya, and H. Shacham. Randomizable proofs and delegatable anonymous credentials. In *CRYPTO 2009*, *LNCS* 5677, pages 108–125. Springer, August 2009.

BCKL08. M. Belenkiy, M. Chase, M. Kohlweiss, and A. Lysyanskaya. P-signatures and noninteractive anonymous credentials. In *TCC 2008*, *LNCS* 4948, pages 356–374. Springer, March 2008.

BCKL09. M. Belenkiy, M. Chase, M. Kohlweiss, and A. Lysyanskaya. Compact e-cash and simulatable VRFs revisited. In *PAIRING 2009*, *LNCS* 5671, pages 114–131. Springer, August 2009.

BCNP04. B. Barak, R. Canetti, J. B. Nielsen, and R. Pass. Universally composable protocols with relaxed set-up assumptions. In *45th FOCS*, pages 186–195. IEEE Computer Society Press, October 2004.

BCP03. E. Bresson, D. Catalano, and D. Pointcheval. A simple public-key cryptosystem with a double trapdoor decryption mechanism and its applications. In *ASIACRYPT 2003*, *LNCS* 2894, pages 37–54. Springer, November / December 2003.

BCPW15. F. Benhamouda, G. Couteau, D. Pointcheval, and H. Wee. Implicit zero-knowledge arguments and applications to the malicious setting. In *CRYPTO 2015, Part II*, LNCS, pages 107–129. Springer, August 2015.

BFM88. M. Blum, P. Feldman, and S. Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *20th ACM STOC*, pages 103–112. ACM Press, May 1988.

BP13. F. Benhamouda and D. Pointcheval. Trapdoor smooth projective hash functions. Cryptology ePrint Archive, Report 2013/341, 2013. `http://eprint.iacr.org/2013/341`.

BR93. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM CCS 93*, pages 62–73. ACM Press, November 1993.

CD00. J. Camenisch and I. Damgård. Verifiable encryption, group encryption, and their applications to separable group signatures and signature sharing schemes. In *ASIACRYPT 2000*, *LNCS* 1976, pages 331–345. Springer, December 2000.

CFN90. D. Chaum, A. Fiat, and M. Naor. Untraceable electronic cash. In *CRYPTO'88*, *LNCS* 403, pages 319–327. Springer, August 1990.

CG15. P. Chaidos and J. Groth. Making sigma-protocols non-interactive without random oracles. In *PKC 2015*, LNCS, pages 650–670. Springer, 2015.

CL15. G. Castagnos and F. Laguillaumie. Linearly homomorphic encryption from DDH. In *CT-RSA 2015*, LNCS, pages 487–505. Springer, 2015.

CMZ14. M. Chase, S. Meiklejohn, and G. Zaverucha. Algebraic MACs and keyed-verification anonymous credentials. In *ACM CCS 14*, pages 1205–1216. ACM Press, 2014.

CNs07. J. Camenisch, G. Neven, and a. shelat. Simulatable adaptive oblivious transfer. In *EUROCRYPT 2007*, *LNCS* 4515, pages 573–590. Springer, May 2007.

CPSV16. M. Ciampi, G. Persiano, L. Siniscalchi, and I. Visconti. A transform for NIZK almost as efficient and general as the Fiat-Shamir transform without programmable random oracles. LNCS, pages 83–111. Springer, 2016.

CS02. R. Cramer and V. Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In *EUROCRYPT 2002*, *LNCS* 2332, pages 45–64. Springer, April / May 2002.

Cv91. D. Chaum and E. van Heyst. Group signatures. In *EUROCRYPT'91*, *LNCS* 547, pages 257–265. Springer, April 1991.

DDO$^+$01. A. De Santis, G. Di Crescenzo, R. Ostrovsky, G. Persiano, and A. Sahai. Robust non-interactive zero knowledge. In *CRYPTO 2001*, *LNCS* 2139, pages 566–598. Springer, August 2001.

DFN06. I. Damgård, N. Fazio, and A. Nicolosi. Non-interactive zero-knowledge from homomorphic encryption. In *TCC 2006*, *LNCS* 3876, pages 41–59. Springer, March 2006.

DJ01. I. Damgård and M. Jurik. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In *PKC 2001*, *LNCS* 1992, pages 119–136. Springer, February 2001.

DJ03. I. Damgård and M. Jurik. A length-flexible threshold cryptosystem with applications. In *ACISP 03*, *LNCS* 2727, pages 350–364. Springer, July 2003.

DJN10. I. Damgård, M. Jurik, and J. B. Nielsen. A generalization of paillier's public-key system with applications to electronic voting. *International Journal of Information Security*, 9(6):371–385, 2010.

EHK$^+$13. A. Escala, G. Herold, E. Kiltz, C. Ràfols, and J. Villar. An algebraic framework for Diffie-Hellman assumptions. In *CRYPTO 2013, Part II*, *LNCS* 8043, pages 129–147. Springer, August 2013.

ElG84. T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *CRYPTO'84*, *LNCS* 196, pages 10–18. Springer, August 1984.

Fis05. M. Fischlin. Communication-efficient non-interactive proofs of knowledge with online extractors. In *CRYPTO 2005*, *LNCS* 3621, pages 152–168. Springer, August 2005.

FLS90. U. Feige, D. Lapidot, and A. Shamir. Multiple non-interactive zero knowledge proofs based on a single random string (extended abstract). In *31st FOCS*, pages 308–317. IEEE Computer Society Press, October 1990.

FS87. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO'86*, *LNCS* 263, pages 186–194. Springer, August 1987.

GH08. M. Green and S. Hohenberger. Universally composable adaptive oblivious transfer. In *ASIACRYPT 2008*, *LNCS* 5350, pages 179–197. Springer, December 2008.

GHKW16. R. Gay, D. Hofheinz, E. Kiltz, and H. Wee. Tightly CCA-secure encryption without pairings. LNCS, pages 1–27. Springer, 2016.

GMR89. S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.

GOS06a. J. Groth, R. Ostrovsky, and A. Sahai. Non-interactive zaps and new techniques for NIZK. In *CRYPTO 2006*, *LNCS* 4117, pages 97–111. Springer, August 2006.

GOS06b. J. Groth, R. Ostrovsky, and A. Sahai. Perfect non-interactive zero knowledge for NP. In *EUROCRYPT 2006*, *LNCS* 4004, pages 339–358. Springer, May / June 2006.

Gro06. J. Groth. Simulation-sound NIZK proofs for a practical language and constant size group signatures. In *ASIACRYPT 2006*, *LNCS* 4284, pages 444–459. Springer, December 2006.

GS08. J. Groth and A. Sahai. Efficient non-interactive proof systems for bilinear groups. In *EUROCRYPT 2008*, *LNCS* 4965, pages 415–432. Springer, April 2008.

KPW15. E. Kiltz, J. Pan, and H. Wee. Structure-preserving signatures from standard assumptions, revisited. In *CRYPTO 2015, Part II*, LNCS, pages 275–295. Springer, August 2015.

KW15. E. Kiltz and H. Wee. Quasi-adaptive NIZK for linear subspaces revisited. LNCS, pages 101–128. Springer, 2015.

Lin15. Y. Lindell. An efficient transform from sigma protocols to NIZK with a CRS and non-programmable random oracle. LNCS, pages 93–109. Springer, 2015.

Lip17. H. Lipmaa. Optimally sound sigma protocols under DCRA. Cryptology ePrint Archive, Report 2017/703, 2017. `http://eprint.iacr.org/2017/703`.

LZ14. S. Laur and B. Zhang. Lightweight zero-knowledge proofs for crypto-computing protocols. In *ISC 2014*, LNCS, pages 140–157. Springer, 2014.

Ore87. Y. Oren. On the cunning power of cheating verifiers: Some observations about zero knowledge proofs (extended abstract). In *28th FOCS*, pages 462–471. IEEE Computer Society Press, October 1987.

Pai99. P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT'99*, *LNCS* 1592, pages 223–238. Springer, May 1999.
PS96. D. Pointcheval and J. Stern. Security proofs for signature schemes. In *EUROCRYPT'96*, *LNCS* 1070, pages 387–398. Springer, May 1996.
PsV06. R. Pass, a. shelat, and V. Vaikuntanathan. Construction of a non-malleable encryption scheme from any semantically secure one. In *CRYPTO 2006*, *LNCS* 4117, pages 271–289. Springer, August 2006.
TFS04. I. Teranishi, J. Furukawa, and K. Sako. k-Times anonymous authentication (extended abstract). In *ASIACRYPT 2004*, *LNCS* 3329, pages 308–322. Springer, December 2004.
VV09. C. Ventre and I. Visconti. Co-sound zero-knowledge with public keys. In *AFRICACRYPT 09*, *LNCS* 5580, pages 287–304. Springer, June 2009.

A An Attack on the Unbounded Soundness of the DFN Transform

In this section, we describe an attack on the unbounded soundness property of a designated-verifier non-interactive zero-knowledge proof obtained using the compiler of [DFN06]. Our attack contradicts Theorem 5 from [DFN06, Appendix A]; for the sake of completeness, we also identify the flaw in the proof of Theorem 5. We apply our attack to the natural example of proving equality of discrete logarithm (which is the example given in [DFN06]), but our attack applies as well to any protocol obtained with this compiler, showing that this compiler cannot be used to construct DVNIZKs with unbounded soundness. We notified the authors of [DFN06] of our findings.

A.1 The DFN Compiler

The DFN compiler applies to any Σ-protocol (which are three-move public-coin honest-verifier zero-knowledge protocols with a specific "commitment - challenge - response" structure) satisfying the following requirements:

- **Relaxed Special Soundness:** for any input x outside of the language, and any first flow a, there is a unique *good* challenge e, where a challenge e is said to be good if there exist a response z such that the proof (a, e, z) on x is accepted by the verifier.
- **Special Honest-Verifier Zero-Knowledge:** there exists an efficient simulator which, on input (x, e) for a word x in the language and a challenge e, outputs a conversation (a, e, z) with distribution statistically indistinguishable from an honest interaction between the prover and the verifier.
- **Linear Answer:** the last flow z of the protocol is a sequence of the form $(u_i + e \cdot v_i)_i$, where the u_i, v_i are integers that can be computed efficiently from x, the random coins of the prover, and its witness.

The DFN compilation technique uses an additively homomorphic encryption scheme S, where the plaintext space is of the form $\mathbb{Z}_k$, for some k which is efficiently computable from the encryption key ek. For simplicity, we use the same notations as [DFN06], rather than the notations used throughout our paper. In particular, we use multiplicative notation for the group and the homomorphic operations on ciphertexts (*i.e.*, if c denotes an encryption of x, c^y denotes an encryption of xy). To avoid confusions, we still use λ for the security parameter, instead of k in [DFN06]. Consider a Σ-protocol satisfying the requirements defined above. The DFN compilation proceeds as follows:

- **Key Setup:** set $(\mathsf{ek}, \mathsf{dk}) \xleftarrow{\$} S.\mathsf{KeyGen}(1^{\lambda'})$, where λ' is an appropriate security parameter computed from λ. Pick a challenge e as in the Σ-protocol, and set c_e to be a random encryption of e with S. Set $\mathsf{pk} \leftarrow (\mathsf{ek}, c_e)$ and $\mathsf{vk} \leftarrow (\mathsf{dk}, e)$.

– **Proof Computation:** given a word x, compute the first flow a as in the Σ-protocol. Let $(u_i, v_i)_i$ be such that the last message z of the Σ-protocol is of the form $(u_i + e \cdot v_i)_i$; compute $\boldsymbol{c_z} \leftarrow (c_i c^{v_i})_i$, where each c_i is a random encryption with S of u_i. Send $x, (a, \boldsymbol{c_z})$.
– **Proof Verification:** decrypt $\boldsymbol{c_z}$ to some $z = (u_i + e \cdot v_i)_i$ using the decryption key dk, and execute the verification procedure of the Σ-protocol using e.

A.2 Proving Equality of Discrete Logarithms

The main example considered in [DFN06] is a Σ-protocol for proving equality of two discrete logarithm. We recall the Σ-protocol, taken from Sections 2.2 of [DFN06], and its compiled version.

Protocol $\mathcal{P}_{\mathsf{eqdlog}}$. Consider a prover P and a verifier V getting a common input (p, g_1, g_2, h_1, h_2) where p is a strong $(\lambda + 1)$-bit prime (hence $p' = (p-1)/2$ is a λ-bit prime), g_1 generates the subgroup of squares of $\mathbb{Z}_p^*$ (which is of order p'), and (g_2, h_1, h_2) are in the subgroup generated by g_1. The witness of P is an exponent $w \in \mathbb{Z}_{p'}$ such that $(h_1, h_2) = (g_1^w, g_2^w)$.

1. P picks $r \xleftarrow{\$} \{0,1\}^{3\lambda}$ and sends $(a_1, a_2) \leftarrow (g_1^r, g_2^r)$ to V.
2. V sends $e \xleftarrow{\$} \mathbb{Z}_{p'}$ to P.
3. P sends $z \leftarrow e \cdot w + r$ to V, who checks that $(g_1^z, g_2^z) = (a_1 h_1^e, a_2 h_2^e)$.

As observed in [DFN06], this protocol satisfies all required properties for the DFN compilation technique to work (relaxed special soundness, special honest-verifier zero-knowledge, and linear answer).

Compiled Protocol. We now describe the DVNIZK obtained by compiling $\mathcal{P}_{\mathsf{eqdlog}}$ with the DFN transform. Let $\lambda' \leftarrow 3\lambda$.[7] Let S be an additively homomorphic encryption scheme over a plaintext space $\mathbb{Z}_k$, where k is a λ'-bit number thant can be computed from the public key generated by $S.\mathsf{KeyGen}(1^{\lambda'})$.

– **Key Setup:** set $(\mathsf{ek}, \mathsf{dk}) \xleftarrow{\$} S.\mathsf{KeyGen}(1^{\lambda'})$. Pick $e \xleftarrow{\$} \mathbb{Z}_{p'}$, and set c_e to be a random encryption of e with S. Set $\mathsf{pk} \leftarrow (\mathsf{ek}, c_e)$ and $\mathsf{vk} \leftarrow (\mathsf{dk}, e)$.
– **Prover:** on common input (p, g_1, g_2, h_1, h_2) and witness w such that $(h_1, h_2) = (g_1^w, g_2^w)$, pick $r \xleftarrow{\$} \{0,1\}^{3\lambda}$, compute $(a_1, a_2) \leftarrow (g_1^r, g_2^r)$, a random encryption c_r of r, and set $c_z \leftarrow c_e^w c_r$. Send (a_1, a_2, c_z) to V.
– **Verifier:** upon receiving (a_1, a_2, c_z) from P, decrypt c_z to some value z using dk, and apply the verification algorithm of $\mathcal{P}_{\mathsf{eqdlog}}$ on (a_1, a_2, z) using challenge e.

By Theorem 1 and Theorem 2 of [DFN06, Section 3], the above protocol is complete, statistically zero-knowledge, and sound against provers generating up to $O(\log \lambda)$ proofs under a complexity-leveraging assumption. The authors report at the end of Section 3 that they believe the compiled protocol to be also secure against provers generating an unbounded number of proofs, and Theorem 5 from Appendix A claims that this belief holds (still under the same complexity-leveraging assumption) in an idealized model where the prover is restricted to use S in a black-box way. In the next section, we challenge

[7] The DFN transform allows for more general choices of security parameter, and larger security parameters lead to a weaker complexity-leveraging assumption. We focus on the simplest case, but our results apply to any generalization as well.

this claim by exhibiting an attack on the above compiled protocol. Our attack extends to any protocol compiled with the DFN transform, showing that, contrary to the beliefs expressed in [DFN06], protocols compiled with the DFN transform do not enjoy unbounded soundness.

A.3 An Attack Against the Unbounded Soundness of the Compiled Protocol

Our attack proceeds as follows: we construct a malicious prover P^* that adaptively sends λ proofs to the verifier V, learning from V for each proof whether it was accepted or not. The idea of the attack is very simple: P^* will compute almost-honest proofs on true statements. The only way P^* deviates from the behavior of an honest prover is when picking r: instead of picking it at random from $\{0,1\}^{3\lambda}$, P^* will set it so that the value $z = e \cdot w + r$ will be larger than k if e exceeds some threshold, and smaller otherwise. Indeed, observe that V does not recover z over the integers: by decrypting c_z, V only recovers $z \bmod k$. This value is equal to z over the integers if and only if $z < k$. When this holds, the verification will succeed, as the statement is true and the proof was honestly constructed from w and r. However, when $z \geq k$ (a modulo reduction occurs when decrypting c_z), the verification will always fail unless k is a multiple of p'. Note that k has no reason to be a multiple of p' in practice; furthermore, when instantiating S with the Paillier encryption scheme (as advocated in [DFN06]), using a modulus which is a multiple of p' would completely break the security of the scheme (as it would leak the factorization of the modulus). Therefore, by learning whether the proof was accepted or rejected, P^* learns whether e is larger than the chosen threshold. After receiving λ such feedbacks, P^* can entirely learn e bit by bit, which allows him to forge proofs on arbitrary statements with probability 1 afterward. We formally describe the attack below.

Formal Description of the Attack. Let $b_0 \leftarrow 1$ and (p, g_1, g_2) be a common input defined as in $\mathcal{P}_{\mathsf{eqdlog}}$, and let $\mathsf{pk} = (\mathsf{ek}, c_e)$ be the public key of the compiled protocol. For $i = 1$ to λ, given $(b_0, \ldots, b_{i-1})$, P^* performs the following steps:

– P^* picks $w_i \overset{\$}{\leftarrow} \mathbb{Z}_{p'}$ and computes $(h_{1,i}, h_{2,i}) \leftarrow (g_1^{w_i}, g_2^{w_i})$. P^* sets $(p, g_1, g_2, h_{1,i}, h_{2,i})$ to be the common input for the ith proof. Note that this corresponds to a true statement, with witness w_i.
– P^* sets $r_i \leftarrow k - w_i \cdot \left\lceil (\sum_{j=0}^{i-1} b_j / 2^{j+1}) \cdot p' \right\rceil$, and computes $(a_{1,i}, a_{2,i}) \leftarrow (g_1^{r_i}, g_2^{r_i})$, a random encryption $c_{r,i}$ of r_i, and $c_{z,i} \leftarrow c_e^{w_i} c_{r,i}$.
– P^* sends the proof $(a_{1,i}, a_{2,i}, c_{z,i})$ to V. If V accepts the proof, P^* sets $b_i \leftarrow 0$. Otherwise, P^* sets $b_i \leftarrow 1$.

Note that when the plaintext z_i decrypted from $c_{z,i}$ is indeed equal to $e \cdot w_i + r_i$ over the integers (hence, no modulo reduction occured), the proof is always accepted by V, as in this case it holds that $(g_1^{z_i}, g_2^{z_i}) = (a_{1,i} h_{1,i}^e, a_{2,i} h_{2,i}^e)$. Otherwise, assuming that p' does not divide k (*i.e.*, $[e \cdot w_i + r_i \bmod k] \neq e \cdot w_i + r_i \bmod p'$), the proof is rejected. By construction, the condition $e \cdot w_i + r_i < k$ reduces to $e < \left\lceil (\sum_{j=0}^{i-1} b_j / 2^{j+1}) \cdot p' \right\rceil$; therefore, by checking whether the proof was rejected or accepted, P^* learns whether this condition holds. One easily observes that, by induction, b_i is exactly the ith bit of e; given the $i-1$ first bits of e, P^* therefore learns the next bit. Hence, after λ such interactions with V, P^* learns all of e. Given e, P^* can forge any further proof of his choice, even on false statements, by proceeding exactly as the simulator for the zero-knowledge property of the compiled protocol.

A.4 The Flaw in the Proof

For the sake of completeness, we attempt to isolate the flaw in the security proof. The proof of Theorem 5 is given in an idealized model, where the adversary is restricted to compute the ciphertext c_z as $S.\mathsf{Enc}(u) \cdot c^v$; *i.e.*, the adversary only uses the homomorphic properties of the scheme in a black-box way (note that this is indeed the case in our attack). This is formalized by saying that all ciphertexts are computed by sending such a pair (u, v) to an oracle, which returns the corresponding $S.\mathsf{Enc}(u) \cdot c^v$. The proof proceeds by letting the simulator, which is given the pair (u, v) corresponding to the ciphertext c_z (as he simulates the oracle), check the following equations:

$$h_1 = g_1^v, \quad h_2 = g_2^v, \quad a_1 = g_1^u, \quad a_2 = g_2^u.$$

The proof goes on by distinguishing two cases: when the statement is false, yet the proof passes the verifier test (but not the simulator test), and when the statement is true, the proof passes the verifier test, but was not created according to the protocol (and is thus rejected by the simulator).

When distinguishing these two cases, the authors implicitly assumed that the following third case never happens: the statement is true and the proof passes the check of the simulator, yet it is refused by the verifier. Intuitively, if c_z was guaranteed to be decrypted over the integers (i.e. no modulo reduction occurs), this would indeed never happen: $(h_i = g_i^v) \wedge (a_i = g_i^u)$ obviously implies that $h_i^e a_i = g_i^{e \cdot v + u}$, for $i = 1, 2$. However, the value z decrypted by the verifier is not $e \cdot v + u$, but $e \cdot v + u \bmod k$, which is different if $e \cdot v + u > k$. When this is the case, it can be that $(h_i = g_i^v) \wedge (a_i = g_i^u)$, yet $h_i^e a_i \neq g_i^z = g_i^{[e \cdot v + u \bmod k]}$; this distinction is missing in the proof; this is exactly the case exploited by our attacker P^* in the previous section.