On Proofs of Plaintext Knowledge for the Joye-Libert Encryption Scheme

Abstract. We provide a simpler and more efficient way to prove knowledge of the plaintext in (a variant of) the Joye-Libert additively homomorphic cryptosystem. We also extend it into a proof that two ciphertexts encrypt the same plaintext under different public keys. This provides a more efficient instantiation of the Naor-Yung paradigm for IND-CCA2 security than in earlier proofs of plaintext equalities for JL ciphertexts. As a result of independent interest, we show that applying the Naor-Yung technique to similar-looking schemes is non-trivial when the soundness of the NIZK proof of plaintext equality relies on an assumption that directly relates to the infeasibility of recovering the secret key from the public key in the underlying IND-CPA scheme.

Keywords. Linearly homomorphic encryption, proofs of plaintext knowledge, proofs of plaintext equality, quadratic residuosity.

1 Introduction

The Joye-Libert (JL) cryptosystem [17,5] is an additively homomorphic encryption scheme over the plaintext space $\mathbb{Z}_{2^k}$, which can be seen as a multi-bit generalization of Goldwasser-Micali [15] and relies on the Quadratic Residuosity assumption for RSA moduli $N = pq$ with a special shape. Like Paillier's cryptosystem [20], the JL scheme supports messages of polynomial length k while offering the benefit of shorter ciphertexts if $k < \frac{1}{4} \log N - \lambda$, where $N = pq$ is the underlying RSA modulus and λ is the security parameter.

In contrast with Paillier (which is additively homomorphic modulo a composite $N = pq$), the scheme is well-suited to applications [7,6] requiring a homomorphic operations modulo a power of 2. In addition, it has the useful feature that multiple public keys can share the same message space $\mathbb{Z}_{2^k}$. Working over a common message space can make it easier to prove plaintext equalities in ciphertexts encrypted under different public keys, as required by the Naor-Yung CCA2-security paradigm [18].[1]

Another useful feature of JL is that valid ciphertexts are dense in their ambient space, of which any element is an encryption of some plaintext. While valid

[1] As observed in [9], proving plaintext equalities between Paillier's ciphertexts encrypted using distinct moduli N_0 and N_1 is harder than it seems. A malicious encryptor can encrypt plaintexts m such that $N_0 < m < N_1$. Addressing this issue requires additional tools such as range proofs or special plaintext encodings [10].

ciphertexts are trivially recognizable, the ring structure of the message space $\mathbb{Z}_{2^k}$ raises non-trivial technical difficulties for an encryptor wishing to demonstrate knowledge of the encrypted plaintext. As mentioned in [7,24], this hinders applications of the scheme in multi-party protocols.

The reason why proving plaintext knowledge is non-trivial is the following. The scheme uses an RSA modulus $N = pq$ such that $p = 1 \bmod 2^k$ and $q = 3 \bmod 4$. A message $m \in \mathbb{Z}_{2^k}$ is encrypted by computing $C = g^m \cdot s^{2^k} \bmod N$ for a random $s \xleftarrow{R} \mathbb{Z}_N^*$, where $g \in \mathbb{Z}_N^*$ is a quadratic non-residue (of Jacobi symbol 1). A tempting approach of proving plaintext knowledge is to use a Σ-protocol inspired by [19,14], where a transcript $(A, e, (y, z))$ (made of a first prover message $A \in \mathbb{Z}_N^*$, a challenge $e \in [0, 2^\lambda - 1]$, and a prover response $(y, z) \in \mathbb{Z}_{2^k} \times \mathbb{Z}_N^*$) satisfies a verification equation of the form $A \cdot C^e = g^z \cdot y^{2^k} \bmod N$. Unfortunately, as mentioned in [7,24], this approach does not guarantee special-soundness in the usual sense. The reason is that two accepting transcripts $(A, e, (y, z))$, $(A, e', (y', z'))$ for the same $A \in \mathbb{Z}_N^*$ lead to an equality $C^{e-e'} = g^{z-z'} \cdot (y/y')^{2^k} \bmod N$, which only guarantees that the plaintext m satisfies $(e - e') \cdot m = z - z' \bmod 2^k$. Since $e - e'$ is very likely to be non-invertible in $\mathbb{Z}_{2^k}$, there is no obvious way to recover the witness $(m, s) \in \mathbb{Z}_{2^k} \times \mathbb{Z}_N^*$ such that $C = g^m \cdot s^{2^k} \bmod N$. Of course, a Σ-protocol with binary challenges avoids this issue (since $e - e'$ is always 1) but it requires $O(\lambda)$ parallel repetitions to make sure that a cheating prover succeeds with negligible probability.

To address this problem, Catalano *et al.* [7] considered a relaxed proof of knowledge that only guarantees knowledge of the $k - \lambda$ least significant bits of the plaintext if the challenge space has size 2^λ. More precisely, their knowledge extractor [7, Appendix C] can extract the $k-d$ least significant bits of $m \in \mathbb{Z}_{2^k}$ if 2^d is the largest power of 2 that divides $e-e'$. Unfortunately, this relaxed notion of extractability does not guarantee anything about the d most significant bits. If we extend it to prove plaintext equalities for ciphertexts encrypted under distinct moduli N_0 and N_1, it only ensures that two plaintexts agree in their $k - \lambda$ least significant bits (or that the prover knows the factorization of N_0 or N_1) if the challenge space has size 2^λ. If we want to apply the Naor-Yung transform [18] in order to build an IND-CCA2-secure encryption scheme containing embedded additively homomorphic ciphertexts, it implies that the decryption algorithm of the resulting CCA2 scheme has to discard the λ most significant bits after a JL decryption. As we will see in Section 5.2, it requires a longer modulus for a fixed number of useful plaintext bits.

As a building block for threshold ECDSA signatures, Xue *et al.* [24] suggested a variant of JL that makes it easier to argue knowledge of all plaintext bits or plaintext equalities. Their scheme is obtained by encrypting $m \in \mathbb{Z}_{2^k}$ as $C = g^m \cdot h^r$, for a random $r \xleftarrow{R} \mathbb{Z}_N$, where h is a generator of the subgroup of 2^k-th residue. Their companion Σ-protocol is actually an *argument* of knowledge as its soundness relies on the hardness of the Strong RSA problem in the subgroup of 2^k-th residues in $\mathbb{Z}_N^*$.

Our Contribution. We describe a very simple variant of the scheme which is more efficient than [24] and still allows the encryptor to prove knowledge of the

entire plaintext and/or plaintext equalities without introducing any additional assumption than the one implied by the IND-CPA security of the scheme itself. In addition, this can be done with a smaller communication complexity (by 48 bytes for one proof at the 128-bit security level) and a faster prover than with the protocol of Xue *et al.* [24]. As we show in Appendix C, this efficiency gain is amplified when we apply Fischlin's transform [13] in order to obtain straight-line extractable proofs of plaintext knowledge in the random oracle model.

When applying the Naor-Yung transform to build an IND-CCA2 cryptosystem (in the random oracle model via the Fiat-Shamir heuristic [12]), our scheme allows reducing the ciphertext size enabled by [24] by ≈ 640 bits without relying on any other assumption than the Quadratic Residuositiy assumption. In addition, it remains possible to extract additively homomorphic JL ciphertexts from the global Naor-Yung ciphertext and perform homomorphic additions on them.

Technical Overview. Our modification of the JL encryption scheme proceeds by introducing $\tau = O(\log |\mathcal{C}|)$ "dummy" modular squarings (where $\mathcal{C}$ is the challenge space of the Σ-protocol) in the encryption scheme of [5]. Namely, it encrypts $m \in \mathbb{Z}_{2^k}$ as $C = g^m \cdot s^{2^{k+\tau}} \bmod N$, where $s \xleftarrow{R} \mathbb{Z}_N^*$. The decryption algorithm and the public key remain unchanged.

Our proof of plaintext knowledge proceeds by re-purposing a Σ-protocol of Fischlin and Fischlin [14], which was designed to serve as a factoring-based identification protocol. From two accepting transcripts satisfying the verification equations, we can obtain $\bar{e} \in [0, 2^\lambda - 1]$, $\bar{z} \in \mathbb{Z}$ and $\bar{y} \in \mathbb{Z}_N^*$ such that $C^{\bar{e}} = g^{\bar{z}} \cdot \bar{y}^{2^{k+\tau}} \bmod N$. If d is the largest integer such that $2^d | \bar{e}$ and $\tau \geq d$, we can then show by adapting the analysis of [7, Appendix C] that the knowledge extractor obtains either the entire plaintext or the factorization of N (which reveals the plaintext). The sole purpose of the τ additional squarings in the encryption algorithm is to deal with the largest power of two that divides the difference between two challenges e and e' in the proof of special-soundness. This proof also exploits the fact that, in the Σ-protocol, the prover's response z contains τ more bits than in the protocol of [7, Appendix C].

When the Σ-protocol is extended to prove plaintext equalities between ciphertexts $C_0 \in \mathbb{Z}_{N_0}^*$ and $C_1 \in \mathbb{Z}_{N_1}^*$ encrypted under distinct moduli N_0 and N_1, we introduce additional checks to have quasi-unique responses [13]. This means that a prover should be unable to find two valid transcripts that only differ in the final message of the prover. As shown by Faust *et al.* [11], quasi-uniqueness is a necessary property to obtain simulation-sound NIZK proofs [23] when a Σ-protocol is compiled with the Fiat-Shamir heuristic. In order to obtain quasi-uniqueness, our verifier checks that the $\mathbb{Z}_{N_0}^*$ and $\mathbb{Z}_{N_1}^*$ components of the prover's response have Jacobi symbol 1. Without this check, an adversary would be able to tamper with honestly generated NIZK proofs (and break their simulation-soundness) by multiplying their components with -1. To make sure that prover responses have Jacobi symbol 1, we also modify the underlying IND-CPA cryptosystem and choose its encryption randomness s in the subgroup of elements with Jacobi symbol 1 (which does not affect the distribution of ciphertexts). Still, we can only argue quasi-uniqueness under the assumption that factoring

N_0 and N_1 is hard, which creates complications when we apply Naor-Yung.

As it turns out, applying the Naor-Yung transform is non-trivial in our setting since the soundness (and the simulation-soundness) of the Σ-protocol relies on the assumption that it is infeasible to recover the secret key that underlies one of the two public keys $N_0 = p_0 q_0$ and $N_1 = p_1 q_1$. One issue is that, in the security proof of Naor-Yung-based cryptosystems, the reduction always needs one of the two secret keys to simulate the decryption oracle. So, when we want to rely on the hardness of factoring to argue that the adversary cannot break the soundness of the proof of plaintext equality, we need to make sure that (with some probability) a soundness adversary provides the reduction with the factorization that it does not already know. To do this, we randomize the choice of which secret key among $\mathsf{sk}_0 = (p_0, q_0)$ and $\mathsf{sk}_1 = (p_1, q_1)$ is used to decrypt and keep this choice independent of the adversary's view until the moment where it creates a fake proof of plaintext equality.

Related Work. Bartoli and Cascudo [2] recently described a method of proving plaintext knowledge in the original scheme [5] using Σ-protocols based on secret sharing. Using black-box secret sharing, they give a technique allowing to extract all plaintext bits in their knowledge extractor. However, the communication complexity of a single proof is the same as in the approach of repeating in parallel Σ-protocols with binary challenges (which requires λ parallel repetitions in order to achieve a knowledge error $2^{-\lambda}$). They also suggest an improved amortized technique allowing to prove plaintext knowledge in multiple ciphertexts at once. Still, the communication cost of an individual proof remains similar to that of repeating in parallel an atomic protocol with binary challenges. In contrast, we obtain a knowledge error $2^{-\lambda}$ after a single protocol execution.

2 Background and Definitions

Notations and definitions. For a prime p and an element $a \in \mathbb{Z}_p^*$, the Legendre symbol of a is defined as $(a \mid p) \triangleq a^{(p-1)/2} \bmod p$ and satisfies $(a \mid p) = 1$ if a is a square in $\mathbb{Z}_p^*$ and $(a \mid q) = -1$ otherwise.

When $N = pq$ is an RSA modulus, the Jacobi symbol of $a \in \mathbb{Z}_N^*$ is defined as $J_N(a) = (a \mid p) \cdot (a \mid q)$ and is efficiently computable from a and N. We denote by $\mathbb{J}_N \triangleq \{a \in \mathbb{Z}_N^* \mid J_N(a) = 1\}$ the multiplicative subgroup of elements with Jacobi symbol 1 in $\mathbb{Z}_N^*$. We denote by $\mathbb{QR}_N \triangleq \{x^2 \bmod N \mid x \in \mathbb{Z}_N^*\}$ the subgroup of squares. The order of $\mathbb{J}_N$ is $|\mathbb{J}_N| = (p-1)(q-1)/2 = 2 \cdot |\mathbb{QR}_N|$. The maximal order of an element modulo N is given by $\lambda(N) = \mathrm{lcm}(p-1, q-1)$. $\mathbb{J}_N$ is cyclic if and only if $\gcd((p-1)/2, (q-1)/2) = 1$.

2.1 Hardness Assumptions and Useful Lemmas

Definition 2.1. *Let an RSA modulus $N = pq$ where p, q are primes of at least $l(\lambda)$ bits for some polynomial $l : \mathbb{N} \to \mathbb{N}$. The* **Quadratic Residuosity** *(QR)*

assumption states that, for any PPT distinguisher $\mathcal{A}$, we have

$$\mathbf{Adv}^{\mathsf{QR}}_{\mathcal{A}}(\lambda) = \big|\Pr[1 \leftarrow \mathcal{A}(N,y) \mid y = x^2 \bmod N,\ x \xleftarrow{R} \mathbb{Z}^*_N] - \Pr[1 \leftarrow \mathcal{A}(N,y) \mid\ y \xleftarrow{R} \mathbb{J}_N \setminus \mathbb{QR}_N]\big| = \mathsf{negl}(\lambda)$$

In the following, we will rely on the k-QR assumption (k-QR) defined in [5], which is the QR assumption for an RSA modulus $N = pq$ with prime factors of the form $p = 2^k p' + 1$, $q = 2q' + 1$ for odd p', q'.

In [5, Theorem 5], it was shown that the k-QR assumption is implied by the following assumption when $p = 1 \bmod 2^k$ and $q = 3 \mod 4$.

Definition 2.2. *Let an RSA modulus $N = pq$ such that $p = 2^k p' + 1$ and $q = 2q' + 1$ for odd p', q', where p, q are primes of at least $l(\lambda)$ bits for some polynomial $l : \mathbb{N} \to \mathbb{N}$. The* **Gap 2^k-Residuosity** *(*$\mathsf{Gap}\ 2^k$-res*) assumption states that, for any PPT distinguisher $\mathcal{A}$, we have*

$$\big|\Pr[1 \leftarrow \mathcal{A}(N,y) \mid y = x^{2^k} \bmod N,\ x \xleftarrow{R} \mathbb{Z}^*_N] - \Pr[1 \leftarrow \mathcal{A}(N,y) \mid\ y \xleftarrow{R} \mathbb{J}_N \setminus \mathbb{QR}_N]\big| = \mathsf{negl}(\lambda)$$

In Section 5, we will rely on the following standard fact.

Lemma 2.3. *For a prime $p > 3$, the Legendre symbol $(3 \mid p) \triangleq 3^{(p-1)/2} \bmod p$ satisfies the property*

$$(3 \mid p) = 1 \quad \Leftrightarrow \quad p = \pm 1 \bmod 12.$$

Consequently, we have $(3 \mid p) = -1$ in the following cases:

- *If $p = 5 \bmod 12$ (or, equivalently, $p = 1 \bmod 4$ and $p = 2 \bmod 3$).*
- *If $p = 7 \bmod 12$ (or, equivalently, $p = 3 \bmod 4$ and $p = 1 \bmod 3$).*

Lemma 2.3 will help us create pseudo-squares (i.e., non-squares $a \in \mathbb{Z}^*_N$ such that $J_N(a) = 1$) when $N = pq$ has prime factors of the form $p = 2^k p' + 1$, $q = 2q' + 1$ for odd p', q' such that $\gcd(p', q') = 1$. If we choose odd p' and q' such that 3 does not divide p' but $3|q'$, then we have $p = 2 \bmod 3$ and $q = 1 \bmod 3$, which implies $(3|p) = (3|q) = -1$.

A simple way to sample a uniformly random $y \in \mathbb{J}_N \setminus \mathbb{QR}_N$ without knowing the factorization of N is then to set $y = 3 \cdot a^2 \bmod N$ for a random $a \xleftarrow{R} \mathbb{Z}^*_N$. The distribution of $\{y = 3 \cdot a^2 \bmod N \mid a \xleftarrow{R} \mathbb{Z}^*_N\}$ is uniform over $\mathbb{J}_N \setminus \mathbb{QR}_N$ because $3 \cdot a^2 \bmod p$ (resp. $3 \cdot a^2 \bmod q$) is uniform in $\mathbb{Z}^*_p \setminus \mathbb{QR}_p$ (resp. $\mathbb{Z}^*_q \setminus \mathbb{QR}_q$). We also note that 2^k divides the order of all pseudo-squares in $\mathbb{Z}^\star_N$. Indeed, since $\gcd(p', q') = 1$, let $\bar{g}$ be a generator of $\mathbb{J}_N$. That is, $\bar{g}$ has maximal order $2^k p' q'$ and then any pseudo-square can be written as $\bar{g}^x$ for some odd x.

In the forthcoming sections, we rely on the following standard fact about non-trivial roots of unity in $\mathbb{Z}^*_N$.

Lemma 2.4. *Let a modulus $N = pq$ for primes p, q of the form $p = 2^k p + 1$, $q = 2q' + 1$ with odd p', q'. For any $d \in [1, k]$, finding $\omega \notin \{-1, 1\}$ such that $\omega^{2^d} = 1 \bmod N$ is as hard as factoring N.*

Proof. We are interested in the roots ω of the polynomial $X^{2^d}-1$ in $\mathbb{Z}_N^*$, which must satisfy $\omega^{2^d}-1=0 \bmod p$ and $\omega^{2^d}-1=0 \bmod q$. We thus consider the roots in $\mathbb{Z}_p$ and $\mathbb{Z}_q$ of $X^{2^d}-1=(X-1)\cdot\prod_{i=1}^{d}(X^{2^{d-i}}+1)$. If $\omega^{2^d}=1 \bmod p$, we have either $\omega \in \{-1,1\}$ or there exists $i \in [1,d-1]$ such that $\omega^{2^{d-i}}=-1 \bmod p$. However, we cannot have $\omega^{2^{d-i}}=-1 \bmod q$ for any $i \in [1,d-1]$ since -1 is a non-square in $\mathbb{Z}_q^*$. Moreover, if $\omega=1 \bmod p$ (resp. $\omega=-1 \bmod p$), we must have $\omega=-1 \bmod q$ (resp. $\omega=1 \bmod q$) if $\omega \neq \pm 1 \bmod N$. From a non-trivial 2^d-th root of unity ω in $\mathbb{Z}_N^*$, we can then obtain $\gcd(\omega^{2^{d-i}}+1, N) \in \{p,q\}$ by trying all indexes $i \in [1,d]$, which can be done in polynomial time since $d \in \mathsf{poly}(\lambda)$. □

2.2 Σ-Protocols

Definition 2.5. *A Σ-protocol $\Sigma=(\mathcal{P},\mathcal{V})$ for an NP language $\mathcal{L} \subseteq \mathcal{X}$ associated with a relation $R=\{(x,w) \in \mathcal{X}\times\mathcal{W} \mid x \in \mathcal{L}\}$ is an interactive proof systems between a prover $\mathcal{P}=(\mathcal{P}_0,\mathcal{P}_1)$ and a verifier $\mathcal{V}=(\mathcal{V}_0,\mathcal{V}_1)$ that are PPT algorithms with the following properties:*

- **3-Move Form:** *For any statement $x \in \mathcal{L}$, $\mathcal{P}$ and $\mathcal{V}$ proceed as follows: (i) $\mathcal{P}_0$ inputs a witness $w \in R(x) \triangleq \{w \in \mathcal{W} : (x,w) \in R\}$, computes $(a,st) \leftarrow \mathcal{P}_0(x,w)$ and sends a to $\mathcal{V}_0$; (ii) $\mathcal{V}_0$ sends back a random challenge $c \in \mathcal{C}$, where $\mathcal{C}$ is the challenge space; (iii) $\mathcal{P}_1$ finally sends a response $z=\mathcal{P}_1(x,w,a,c,st)$ to $\mathcal{V}_1$; (iv) On input of (a,c,z), $\mathcal{V}_1$ outputs 1 or 0.*
- **Completeness:** *If $(x,w) \in R$ and $\mathcal{P}$ honestly computes (a,z) for a challenge c, then $\mathcal{V}_1(x,(a,c,z))$ outputs 1 with probability $1-\mathsf{negl}(\lambda)$.*
- **Special soundness:** *There exists a PPT knowledge extractor $\mathcal{E}$ that, for any public input x, on input of two accepting transcripts (a,c,z) and (a,c',z') with $c \neq c'$, outputs a witness w' such that $(x,w') \in R$.*
- **Honest-verifier zero-knowledge:** *There is a PPT simulator $\mathcal{S}$ such that, for any PPT distinguisher $(\mathcal{D}_0,\mathcal{D}_1)$ and any $(x,w) \in R$, we have*

$$\mathbf{Adv}^{\mathrm{HVZK}}_{\mathcal{S},\mathcal{D}}(\lambda) \triangleq \left|\Pr[\mathsf{Exp}^{\mathrm{REAL}}_{\Sigma,\mathcal{D}}(1^\lambda)=1]-\Pr[\mathsf{Exp}^{\mathrm{SIM}}_{\Sigma,\mathcal{D}}(\mathcal{S},1^\lambda)=1]\right| \leq \mathsf{negl}(\lambda),$$

where the real and ideal experiments are defined as

$\mathsf{Exp}^{\mathrm{REAL}}_{\Sigma,\mathcal{D}}(1^\lambda)$:	$\mathsf{Exp}^{\mathrm{SIM}}_{\Sigma,\mathcal{D}}(\mathcal{S},1^\lambda)$:
1. $(x,w,st) \leftarrow \mathcal{D}_0(1^\lambda)$	1. $(x,w,st) \leftarrow \mathcal{D}_0(1^\lambda)$
2. $\pi \leftarrow \langle \mathcal{P}(x,w;1^\lambda), \mathcal{V}(x;1^\lambda)\rangle$	2. $\pi \leftarrow \mathcal{S}(x,1^\lambda)$
3. *Output* $\mathcal{D}_1(\pi,st)$.	3. *Output* $\mathcal{D}_1(\pi,st)$.

where $\langle \mathcal{P}(x,w;1^\lambda), \mathcal{V}(x;1^\lambda)\rangle$ denotes the proof produced by the interaction between $\mathcal{P}$ and $\mathcal{V}$ on a common input x and private witness w.

Fischlin [13] introduced a property, called *quasi-uniqueness of responses*. Informally, it means that a PPT prover should not be able to come up with two valid transcripts that only differ in their response z for the same statement.

Definition 2.6 ([13]). *A Σ-protocol has quasi-unique responses if, for any PPT algorithm $\mathcal{A}$ and any security parameter λ, we have*

$$\Pr[\ (x,a,c,z,z') \leftarrow \mathcal{A}(1^\lambda) : \mathcal{V}_1(x,a,c,z) = \mathcal{V}_1(x,a,c,z') = 1 \wedge z \neq z'\ ] \leq \mathsf{negl}(\lambda).$$

Faust *et al.* [11] showed that Σ-protocol need to have quasi-unique responses in order to provide simulation-sound [23] NIZK proofs by applying Fiat-Shamir.

2.3 Simulation-Extractable NIZK in the Random Oracle Model

We now recall the notions of simulation-soundness and simulation-extractability for NIZK proofs in the random oracle model, as they were defined in [11].

Definition 2.7 (Simulation-Soundness). *Let $\mathcal{L}$ an NP language and a proof system $(\mathcal{P}^H, \mathcal{V}^H)$ for $\mathcal{L}$ with a zero-knowledge simulation $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$, where $\mathcal{S}_1(q_i)$ simulates the random oracle H and returns the first output of $(h_i = H(q_i), st) \leftarrow \mathcal{S}(1, st, q_i)$ while $\mathcal{S}_2(x)$ returns the first output of $(\pi, st) \leftarrow \mathcal{S}(2, st, x)$. We say that $(\mathcal{P}^H, \mathcal{V}^H)$ is simulation-sound with respect to $\mathcal{S}$ in the ROM if, for any PPT adversary $\mathcal{A}$, we have*

$$\mathbf{Adv}^{\text{sim-sound}}_{\mathcal{A}}(\lambda) \triangleq \Pr[\ (x^\star,\pi^\star) \leftarrow \mathcal{A}^{\mathcal{S}_1(\cdot),\mathcal{S}_2(\cdot)} : (x^\star,\pi^\star) \notin \mathcal{T} \wedge \quad x^\star \notin \mathcal{L} \quad \wedge \quad \mathcal{V}^{\mathcal{S}_1}(x^\star,\pi^\star) = 1\ ] \leq \mathsf{negl}(\lambda)$$

where $\mathcal{T}$ is the list of pairs (x_i, π) created by $\mathcal{S}$ (when $\mathcal{S}_2$ returns a simulated proof π_i on input of a queried statement x_i).

Definition 2.8 (Simulation-Extractability). *Let $\mathcal{L}$ an NP language and a proof system $(\mathcal{P}^H, \mathcal{V}^H)$ for $\mathcal{L}$ with a zero-knowledge simulation $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$, where $\mathcal{S}_1(q_i)$ simulates the random oracle H and returns the first output of $(h_i = H(q_i), st) \leftarrow \mathcal{S}(1, st, q_i)$ while $\mathcal{S}_2(x)$ returns the first output of $(\pi, st) \leftarrow \mathcal{S}(2, st, x)$. In the ROM, we say that $(\mathcal{P}^H, \mathcal{V}^H)$ is simulation-extractable with extraction error ν with respect to $\mathcal{S}$ if, for any PPT adversary $\mathcal{A}$, there exists a PPT extractor $\mathcal{E}_{\mathcal{A}}$ with access to the answers $\mathcal{T}_H$, $\mathcal{T}$ of $(\mathcal{S}_1, \mathcal{S}_2)$ such that the following holds: Define*

$$\varepsilon \triangleq \Pr[\ (x^\star,\pi^\star) \leftarrow \mathcal{A}^{\mathcal{S}_1(\cdot),\mathcal{S}_2(\cdot)}(1^\lambda;\rho) : (x^\star,\pi^\star) \notin \mathcal{T};\ \mathcal{V}^{\mathcal{S}_1}(x^\star,\pi^\star) = 1\ ]$$

$$ext \triangleq \Pr[\ (x^\star,\pi^\star) \leftarrow \mathcal{A}^{\mathcal{S}_1(\cdot),\mathcal{S}_2(\cdot)}(1^\lambda;\rho), \quad w^\star \leftarrow \mathcal{E}_{\mathcal{A}}(x^\star,\pi^\star;\rho,\mathcal{T}_H,\mathcal{T}) : (x^\star,\pi^\star) \notin \mathcal{T};\ (x^\star,w^\star) \in R_{\mathcal{L}}\ ]$$

where the probability space is over the coin tosses of $\mathcal{S}$ and the adversary's random tape ρ. Then, there exists a constant $d > 0$ and a polynomial $P(\lambda)$ such that, whenever $\varepsilon \geq \nu$, we have $ext \geq (\varepsilon - \nu)^d / P(\lambda)$.

2.4 Forking Lemma

In this section, we recall the Forking Lemma [22]. In the following, we will use the version that was proven by Bellare and Neven [4].

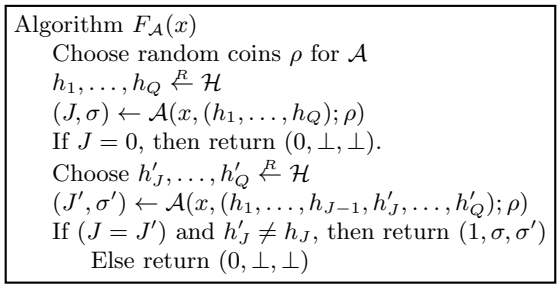

Fig. 1. Forking algorithm $F_{\mathcal{A}}(x)$.

Lemma 2.9 (General Forking Lemma [4, Lemma 1]). *Let an integer $Q \geq 1$ and let a set $\mathcal{H}$ of size $h \geq 2$. Let $\mathcal{A}$ a randomized algorithm that, on input $(x, h_1, \ldots, h_Q)$, outputs a pair comprised of an integer $J \in [0, Q]$ and a side output σ. Let IG a randomized input generator. The accepting probability acc of $\mathcal{A}$ is the probability that $J \geq 1$ in the experiment*

$$x \leftarrow \mathsf{IG};\ (h_1, \ldots, h_Q) \xleftarrow{R} \mathcal{H};\ (J, \sigma) \leftarrow \mathcal{A}(x, (h_1, \ldots, h_Q)).$$

The forking algorithm $F_{\mathcal{A}}(x)$ associated with $\mathcal{A}$ is described in Figure 1. If we define the probability $\mathrm{frk} := \Pr[b = 1 \ : \ x \leftarrow \mathsf{IG};\ (b, \sigma, \sigma') \leftarrow F_{\mathcal{A}}(x)]$, *then we have* $\mathrm{frk} \geq acc \cdot \left(\frac{acc}{Q} - \frac{1}{h} \right)$.

2.5 The (Modified) JL Cryptosystem

We first describe a simple modification of the Joye-Libert cryptosystem [17,5] that will make it easier to prove knowledge of the plaintext. The original scheme uses an RSA modulus $N = pq$ such that $p = 2^k p' + 1$ and $q = 2q' + 1$, for odd integers p', q', where k is the plaintext length. An encryption of $m \in \mathbb{Z}_{2^k}$ is computed as $C = g^m \cdot s^{2^k} \bmod N$ for a random $s \xleftarrow{R} \mathbb{Z}_N^*$. Such a ciphertext can be decrypted by first computing $C_0 = C^{p'} \bmod p = g^{m \cdot p'} \bmod p$ before solving a discrete logarithm in a subgroup of smooth order (which is a power of 2) using the Pohlig-Hellman algorithm [21].

Our modification consists in encrypting m as $C = g^m \cdot s^{2^{k+\tau}} \bmod N$, where g is a random pseudo-square (not necessarily of maximal order), and $\tau \geq \lambda - 1$ if $\lambda \in \mathbb{N}$ is the security parameter (looking ahead, it will be the soundness parameter in the proof of plaintext knowledge). We note that this change does not modify the distribution of C since $s^{2^{k+\tau}} = (s^{2^k})^{2^\tau}$ and squaring is a permutation in the subgroup of 2^k-th residue. Therefore the security proof of [5] immediately carries over to the modified scheme. The security proof first modifies the public key and resorts to the $\mathsf{Gap}\ 2^k\text{-}\mathsf{res}$ assumption in order to replace $g \in \mathbb{J}_N \setminus \mathbb{QR}_N$ by a 2^k-th residue. After this change, the encryption algorithm becomes lossy [25,3,16] and ciphertexts reveal no information about the plaintext.

Keygen$(1^\lambda, 1^k)$**:** Given a security parameter λ and a message length $k \in \mathsf{poly}(\lambda)$,

1. Choose an RSA modulus $N = pq$ for large primes $p = 2^k p' + 1$ and $q = 2q' + 1$ for odd (not necessarily prime) integers p', q' such that $p, q > 2^{l(\lambda)}$, for some polynomial $l : \mathbb{N} \to \mathbb{N}$, and $\gcd(p', q') = 1$.
2. Choose a random element $g \in \mathbb{J}_N \setminus \mathbb{QR}_N$. Choose a parameter τ such that $k \geq \tau \geq \lambda - 1$.

Return the key pair $(\mathsf{pk}, \mathsf{sk})$ where $\mathsf{sk} := (p, q, p', q')$ and $\mathsf{pk} := (N,\ k,\ \tau,\ g)$.

Encrypt(pk, m)**:** Given a public key pk and a message $m \in \mathbb{Z}_{2^k}$, choose $s \xleftarrow{R} \mathbb{Z}_N^*$ and compute and output the ciphertext $C = g^m \cdot s^{2^{k+\tau}} \bmod N$.

Decrypt(sk, C)**:** Given $\mathsf{sk} = (p, q, p', q')$ and $C \in \mathbb{J}_N$, do the following:

1. Compute $C_0 = C^{p'} \bmod p$.
2. Use the Pohlig-Hellman algorithm (see [5, Section 3.2] for details) to compute $m \in \mathbb{Z}_{2^k}$ such that $C_0 = (g^{p'})^m \bmod p$.

A useful property of the scheme is that the ciphertext space is dense in its ambient space $\mathbb{J}_N$: all elements of $\mathbb{J}_N$ are encryptions of some plaintext $m \in \mathbb{Z}_{2^k}$. Indeed, since $\gcd(p', q') = 1$, $\langle \bar{g} \rangle = \mathbb{J}_N$, for some $\bar{g}$. Therefore, $g = \bar{g}^x$ for an odd $x \in \mathbb{Z}$. If $C \in \mathbb{J}_N$, then $C = \bar{g}^\gamma$, for some γ. For (x', ℓ) such that $xx' + \ell 2^k = 1$, we can write $C = \bar{g}^{\gamma x x' + \gamma \ell 2^k} = g^{\bar{m}} (\bar{g}^{x \lfloor \gamma x' / 2^k \rfloor + \gamma \ell})^{2^k}$, where $\bar{m} = \gamma x' \bmod 2^k$.

Since computing a Jacobi symbol does not require to know the factorization of N, a verifier can efficiently check that a given $C \in \mathbb{Z}_N^*$ is in an encryption of some plaintext. On the other hand, proving knowledge of the plaintext is non-trivial since the plaintext space is a ring with many zero divisors.

Another useful property of the scheme is that the distribution of the ciphertext does not change if we sample the randomness s uniformly in the subgroup $\mathbb{J}_N$ (instead of $\mathbb{Z}_N^*$). Indeed, since $J_N(-1) = -1$, we have $s^{2^{k+\tau}} = (-s)^{2^{k+\tau}} \bmod N$ and exactly one of s and $-s$ is in $\mathbb{J}_N$. The encryptor can thus choose the one in $\mathbb{J}_N$ without changing the distribution of C. This property will be useful in order to obtain quasi-unique responses in our Σ-protocol proving plaintext equalities.

3 Shorter Proofs of Plaintext Knowledge

In this section, we give a modified Σ-protocol that allows extracting *all* plaintext bits if the ciphertext is computed as in Section 2.5. The Σ-protocol is almost identical to the identification scheme of Fischlin and Fischlin [14]. There are actually two differences with [14]. The first one is that the Fischlin-Fischlin identification protocol involves a prover knowing a representation $(m, s) \in \mathbb{Z}_{2^k} \times \mathbb{Z}_N^*$ of $C = g^m \cdot s^{2^{k+\tau}} \bmod N$ where g is also a 2^k-th residue (instead of being a non-square in our setting). The second difference is that our verifier explicitly checks that C has a positive Jacobi symbol (i.e., that $J_N(C) = 1$). Besides these differences, the Σ-protocol is identical to the one of [14, Section 3.1].

3.1 The Σ-Protocol

Inputs: The prover P has $(m, s) \in \mathbb{Z}_{2^k} \times \mathbb{Z}_N^*$ such that $C = g^m \cdot s^{2^{k+\tau}} \bmod N$. The verifier V only knows C and wants to be convinced that P knows the plaintext underlying C.

1. P chooses $u \xleftarrow{R} \mathbb{Z}_{2^{k+\tau}}$, $v \xleftarrow{R} \mathbb{Z}_N^*$ and computes
$$A = g^u \cdot v^{2^{k+\tau}} \bmod N$$
which is sent to V
2. V sends a random challenge $e \xleftarrow{R} [0, 2^\lambda - 1]$ to P
3. P computes the response
$$z = u + e \cdot m \bmod 2^{k+\tau}, \qquad y = v \cdot s^e \cdot g^{\lfloor (u+em)/2^{k+\tau} \rfloor} \bmod N$$
4. V accepts if $J_N(C) = 1$ and
$$A \cdot C^e = g^z \cdot y^{2^{k+\tau}} \bmod N \tag{1}$$

The protocol is complete since

$$\begin{aligned} g^z \cdot y^{2^{k+\tau}} &\equiv g^{u+e\cdot m \bmod 2^{k+\tau}} \cdot v^{2^{k+\tau}} s^{2^{k+\tau}\cdot e} \cdot g^{2^{k+\tau}\cdot\lfloor (u+em)/2^{k+\tau}\rfloor} \\ &\equiv g^{u+e\cdot m} \cdot \left(v \cdot s^e\right)^{2^{k+\tau}} \\ &\equiv g^u \cdot v^{2^{k+\tau}} \cdot \left(g^m \cdot s^{2^{k+\tau}}\right)^e \equiv A \cdot C^e \pmod N \end{aligned}$$

The protocol is honest-verifier zero-knowledge for the same reason as in [14], via the canonical HVZK simulator that chooses $e \xleftarrow{R} [0, 2^\lambda - 1]$, $z \xleftarrow{R} \mathbb{Z}_{2^{k+\tau}}$, $y \xleftarrow{R} \mathbb{Z}_N^*$ and computes $A = C^{-e} \cdot g^z \cdot y^{2^{k+\tau}} \bmod N$.

We now turn to proving its special soundness. We note that, just like the original protocol of Fischlin and Fischlin [14], it does not prove knowledge of a representation of C as $C = g^m \cdot s^{2^{k+\tau}} \bmod N$. Instead, it proves knowledge (either directly or via the factorization of N) of a representation $(m, w) \in \mathbb{Z}_{2^k} \times \mathbb{Z}_N^*$ of C as $C = g^m \cdot w^{2^k} \bmod N$, which is sufficient to guarantee that C decrypts to m. In the proof of Lemma 3.1, the extractor exploits the fact that z lives in $\mathbb{Z}_{2^{k+\tau}}$ (instead of $\mathbb{Z}_{2^k}$ in [7]) to completely recover m.

Lemma 3.1. *The above Σ-protocol provides special soundness as a proof of knowledge of a pair $(m, w) \in \mathbb{Z}_{2^k} \times \mathbb{Z}_N^*$ such that $C = g^m \cdot w^{2^k} \bmod N$.*

Proof. Suppose that we have two accepting transcripts $(A, e, (y, z))$, $(A, e', (y', z'))$ satisfying (18). We assume w.l.o.g. that $e > e'$ so that $\bar{e} \triangleq e - e' \in [0, 2^\lambda - 1]$. If we define $\bar{z} = z - \bar{z}$ (over $\mathbb{Z}$) and $\bar{y} = y/y' \bmod N$, we have

$$C^{\bar{e}} = g^{\bar{z}} \cdot \bar{y}^{2^{k+\tau}} \bmod N \tag{2}$$

Let d the largest integer such that $2^d | \bar{e}$ and write $\bar{e} = 2^d \cdot \bar{e}_o$ for an odd $\bar{e}_o$ (by construction, we have $d \leq \lambda - 1 \leq \tau$). Let also the largest t_z such that $\bar{z} = 2^{t_z} \cdot \bar{z}_o$ for an odd $\bar{z}_o \in \mathbb{Z}$. We now consider two cases.

• $t_z < d$. In this case, (2) becomes

$$C^{2^d \cdot \bar{e}_o} = g^{2^{t_z} \cdot \bar{z}_o} \cdot \bar{y}^{2^{k+\tau}} \bmod N \tag{3}$$

which implies

$$C^{2^{d-t_z} \cdot \bar{e}_o} = \mu_0 \cdot g^{\bar{z}_o} \cdot \bar{y}^{2^{k+\tau-t_z}} \bmod N \tag{4}$$

for some μ_0 such that $\mu_0^{2^{t_z}} = 1 \bmod N$. As pointed out in [7, Appendix C], we cannot have $\mu_0 = -1$ (since $J_N(-1) = -1$ and $J_N(g) = 1$ so that the two members of (4) would have distinct Jacobi symbols) nor $\mu_0 = 1$ (since g is not a square). At this point, if $t_z < d$ and (3) holds, we know that

$$\mu_0 = C^{2^{d-t_z} \cdot \bar{e}_o} \cdot g^{-\bar{z}_o} \cdot \bar{y}^{-2^{k+\tau-t_z}} \bmod N \tag{5}$$

is a non-trivial root of unity. This would reveal the factorization of N and the plaintext. However, we claim that μ_0 cannot be non-trivial either. The reason is that (5) implies $J_N(\mu_0) = 1$ (i.e., $(\mu_0 \mid p) = (\mu_0 \mid q)$), meaning that

$$\mu_0^{2^{k-1}p'} \bmod p = \left(\frac{\mu_0}{p}\right) = \left(\frac{\mu_0}{q}\right) = \mu_0^{q'} \bmod q \tag{6}$$

Moreover, (5) also implies $\mu_0^{q'} \bmod q = -1$ because $t_z < d$, $\bar{z}_o$ is odd, and $g \in \mathbb{Z}_N^*$ was chosen in such a way that $g^{q'} \bmod q = \left(\frac{g}{q}\right) = \left(\frac{g}{p}\right) = -1$. However, since $t_z < d \leq \lambda - 1 \leq k$, $\mu_0^{2^{t_z}} = 1 \bmod N$ implies $\mu_0^{2^{k-1}p'} = 1 \bmod p$, which contradicts the equality $\mu_0^{q'} = -1 \bmod q$ if (6) holds. Since μ_0 is neither in $\{-1, 1\}$ nor among the non-trivial 2^{t_z}-th roots of 1, the above means that we cannot have $t_z < d$ at all.

• $t_z \geq d$. In this case, we depart from the knowledge extractor of [7]. Since

$$C^{2^d \cdot \bar{e}_o} = g^{2^{t_z} \cdot \bar{z}_o} \cdot \bar{y}^{2^{k+\tau}} \bmod N, \tag{7}$$

we have

$$C^{\bar{e}_o} = \mu_1 \cdot g^{2^{t_z-d} \cdot \bar{z}_o} \cdot \bar{y}^{2^{k+\tau-d}} \bmod N, \tag{8}$$

for some μ_1 such that $\mu_1^{2^d} = 1 \bmod N$. As in the first case, we cannot have $\mu_1 = -1 \bmod N$ since $J_N(-1) = -1$ and we have $J_N(C) = J_N(g) = 1$.[2] If

$$\mu_1 = C^{\bar{e}_o} \cdot g^{-2^{t_z-d} \cdot \bar{z}_o} \cdot \bar{y}^{-2^{k+\tau-d}} \bmod N \tag{9}$$

is a 2^d-th root of 1 such that $\mu_1 \notin \{-1, 1\}$, it reveals the factorization of N (by Lemma 2.4) and then the plaintext $\bar{m}$. From $\bar{m} \in \mathbb{Z}_{2^k}$ and the factorization, the knowledge extractor can then also compute a 2^k-th root w of $C \cdot g^{-\bar{m}}$.

We are left with the case $\mu_1 = 1$, meaning that

$$C^{\bar{e}_o} = g^{2^{t_z-d} \cdot \bar{z}_o} \cdot \bar{y}^{2^{k+\tau-d}} \bmod N \tag{10}$$

[2] This is where we use the explicit check that $J_N(C) = 1$.

Since $\gcd(\bar{e}_o, 2^{k+\tau-d}) = 1$, we can find $\alpha, \beta \in \mathbb{Z}$ such that $\alpha \cdot \bar{e}_o + \beta \cdot 2^{k+\tau-d} = 1$, which yield

$$\begin{aligned} C &= g^{2^{t_z-d} \cdot \bar{z}_o \cdot \alpha} \cdot \left(\bar{y}^{\alpha} \cdot C^{\beta}\right)^{2^{k+\tau-d}} \bmod N \\ &= g^{2^{t_z-d} \cdot \bar{z}_o \cdot \alpha} \cdot \left((\bar{y}^{\alpha} \cdot C^{\beta})^{2^{\tau-d}}\right)^{2^k} \bmod N \\ &= g^{2^{t_z-d} \cdot \bar{z}_o \cdot \alpha \bmod 2^k} \cdot \left((\bar{y}^{\alpha} \cdot C^{\beta})^{2^{\tau-d}} \cdot g^{\lfloor 2^{t_z-d} \cdot \bar{z}_o \cdot \alpha / 2^k \rfloor}\right)^{2^k} \bmod N \end{aligned} \tag{11}$$

Since $d \leq \lambda - 1 \leq \tau$, this implies that C decrypts to

$$\bar{m} \triangleq 2^{t_z-d} \cdot \bar{z}_o \cdot \alpha \bmod 2^k = (\bar{z}/2^d) \cdot \bar{e}_o^{-1} \bmod 2^k$$

and the witness components $(\bar{m}, w \triangleq (\bar{y}^{\alpha} \cdot C^{\beta})^{2^{\tau-d}} \cdot g^{\lfloor 2^{t_z-d} \cdot \bar{z}_o \cdot \alpha / 2^k \rfloor} \bmod N)$ are both computable by the knowledge extractor. □

3.2 Comparisons

In Appendix A, we recall the scheme and the proof of plaintext knowledge of Xue *et al.* [24] and show that the above construction provides shorter poofs (by 384 bits for $\lambda = 128$).

Another advantage of our protocol over their proof of plaintext knowledge is that, in the latter, the knowledge extractor obtains either the plaintext or the solution of a Strong RSA [1] instance.[3] As such, it is "only" an argument (instead of a proof) of knowledge of the plaintext since extracting the solution of a Strong RSA instance does not seem to enable plaintext reconstruction. In contrast, the Σ-protocol of Section 3.1 does not rely on any hardness assumption to make sure that the knowledge extractor can always obtain the plaintext.

Yet another advantage of our construction over [24] is that the knowledge extractor does not require $p' = (p-1)/2^k$ and $q' = (q-1)/2$ to not have any small prime divisor. We only need p' and q' to be odd and co-prime. In order to apply the Naor-Yung paradigm in Section 5, we actually need q' to be divisible by 3 (in order for the security reduction to publicly generate pseudo-squares) but the proof of plaintext knowledge itself does not rely on this condition.

4 Proof of Plaintext Equality

We now adapt the Σ-protocol of Section 3.1 so as to prove that two ciphertexts encrypt the same plaintext under two distinct public keys corresponding to the same message space $\mathbb{Z}_{2^k}$. We note that this is not immediately possible [9] in Paillier's cryptosystem [20], where the message space depends on the public key.

This Σ-protocol will be useful to apply the Naor-Yung paradigm [18] in order to construct an IND-CCA2 encryption scheme where ciphertexts can be

[3] More precisely, they rely on the "Strong JL" assumption, which is essentially the Strong RSA assumption in the subgroup of 2^k-th residues.

"downgraded" into ciphertexts of an additively homomorphic scheme. To do this, we need the underlying NIZK proof of plaintext equalities to be simulation-sound [23]. In order to ensure the simulation-soundness of Fiat-Shamir-based proofs, we need to apply a result of Faust *et al.* [11] which requires the underlying Σ-protocol to have quasi-unique responses. We thus have to make sure that a malicious prover is not be able to come up with two valid Σ-protocol transcripts that only differ in their last message. This is the reason why we require the response components $y_0 \in \mathbb{Z}^*_{N_0}$, $y_1 \in \mathbb{Z}^*_{N_1}$ to have Jacobi symbol 1. Without this restriction, the adversary could break the quasi-unique response property by, e.g., changing y_0 into $-y_0 \in \mathbb{Z}^*_{N_0}$ in a valid transcript $((A_0, A_1), e, (z, y_0, y_1))$.

4.1 The Σ-Protocol

Inputs: Let two distinct public keys $\mathsf{pk}_0 := (N_0, k, \tau, g_0)$, $\mathsf{pk}_1 := (N_1, k, \tau, g_1)$ sharing the same parameters k, τ. Let a statement comprised of pk_0, pk_1 and two ciphertexts $C_0 \in \mathbb{J}_{N_0}$, $C_1 \in \mathbb{J}_{N_1}$ encrypting the same $m \in \mathbb{Z}_{2^k}$. The prover P has a witness $(m, s_0, s_1) \in \mathbb{Z}_{2^k} \times \mathbb{J}_{N_0} \times \mathbb{J}_{N_1}$ such that

$$C_0 = g_0^m \cdot s_0^{2^{k+\tau}} \bmod N_0 \qquad C_1 = g_1^m \cdot s_1^{2^{k+\tau}} \bmod N_1.$$

The verifier V only knows $(\mathsf{pk}_0, \mathsf{pk}_1, C_0, C_1)$ and wants to get convinced that C_0 and C_1 decrypt to the same plaintext, which is known to P.

1. P chooses $u \xleftarrow{R} \mathbb{Z}_{2^{k+\tau}}$, $v_0 \xleftarrow{R} \mathbb{J}_{N_0}$, $v_1 \xleftarrow{R} \mathbb{J}_{N_1}$ and computes

$$A_0 = g_0^u \cdot v_0^{2^{k+\tau}} \bmod N_0, \qquad A_1 = g_1^u \cdot v_1^{2^{k+\tau}} \bmod N_1$$

   which is sent to V
2. V sends a random challenge $e \xleftarrow{R} [0, 2^\lambda - 1]$ to P
3. P computes the response (z, y_0, y_1) as $z = u + e \cdot m \bmod 2^{k+\tau}$ and

$$y_0 = v_0 \cdot s_0^e \cdot g_0^{\lfloor (u+em)/2^{k+\tau} \rfloor} \bmod N_0$$
$$y_1 = v_1 \cdot s_1^e \cdot g_1^{\lfloor (u+em)/2^{k+\tau} \rfloor} \bmod N_1$$

4. V accepts if $J_{N_0}(C_0) = J_{N_1}(C_1) = 1$, $J_{N_0}(y_0) = J_{N_1}(y_1) = 1$ and

$$A_0 \cdot C_0^e = g_0^z \cdot y_0^{2^{k+\tau}} \bmod N_0, \qquad A_1 \cdot C_1^e = g_1^z \cdot y_1^{2^{k+\tau}} \bmod N_1 \tag{12}$$

We note that there is no need to explicitly check that $J_{N_0}(A_0) = J_{N_1}(A_1) = 1$ at step 1 since it is implied by (12) and the conditions $J_{N_0}(C_0) = J_{N_1}(C_1) = 1$.

4.2 Security

Lemma 4.1. *The above protocol provides special soundness as a Σ-protocol proving of knowledge of either: (i) A witness $(m, w_0, w_1) \in \mathbb{Z}_{2^k} \times \mathbb{Z}^*_{N_0} \times \mathbb{Z}^*_{N_1}$ such that $C_0 = g_0^m \cdot w_0^{2^k} \bmod N_0$ and $C_1 = g_1^m \cdot w_1^{2^k} \bmod N_1$; or (ii) The factorization of at least one of the moduli N_0 and N_1.*

Proof. Suppose that we have two accepting transcripts $((A_0, A_1), e, (y_0, y_1, z))$, $((A_0, A_1), e', (y_0', y_1', z'))$ satisfying (12). We give an efficient extractor that computes either a witness (m, w_1, w_2) satisfying condition (i) in the statement of the lemma or a prime factor of N_0 or N_1.

We assume w.l.o.g. that $e > e'$ and define $\bar{e} \triangleq e - e' \in [0, 2^\lambda - 1]$, $\bar{z} = z - \bar{z}$ (over $\mathbb{Z}$) and $\bar{y}_0 = y_0/y_0' \bmod N_0$, $\bar{y}_1 = y_1/y_1' \bmod N_1$. Then, we have

$$C_0^{\bar{e}} = g_0^{\bar{z}} \cdot \bar{y}_0^{2^{k+\tau}} \bmod N_0, \qquad C_1^{\bar{e}} = g_1^{\bar{z}} \cdot \bar{y}_1^{2^{k+\tau}} \bmod N_1 \tag{13}$$

Let the largest $d \in \mathbb{Z}$ such that $2^d | \bar{e}$ and let $\bar{e} = 2^d \cdot \bar{e}_o$ for an odd $\bar{e}_o$ (note that $d \leq \lambda - 1 \leq \tau$ by construction). By the same argument as in the proof of Lemma 3.1, we must have $2^d | \bar{z}$: namely, if we define the largest t_z such that $\bar{z} = 2^{t_z} \cdot \bar{z}_o$ for an odd $\bar{z}_o \in \mathbb{Z}$, we have $t_z \geq d$. Then, (13) becomes

$$C_0^{2^d \cdot \bar{e}_o} = g_0^{2^{t_z} \cdot \bar{z}_o} \cdot \bar{y}_0^{2^{k+\tau}} \bmod N_0, \qquad C_1^{2^d \cdot \bar{e}_o} = g_1^{2^{t_z} \cdot \bar{z}_o} \cdot \bar{y}_1^{2^{k+\tau}} \bmod N_1,$$

which implies

$$C_0^{\bar{e}_o} = \mu_0 \cdot g_0^{2^{t_z - d} \cdot \bar{z}_o} \cdot \bar{y}_0^{2^{k+\tau-d}} \bmod N_0, \tag{14}$$

$$C_1^{\bar{e}_o} = \mu_1 \cdot g_1^{2^{t_z - d} \cdot \bar{z}_o} \cdot \bar{y}_1^{2^{k+\tau-d}} \bmod N_1 \tag{15}$$

for some $\mu_0 \in \mathbb{Z}_{N_0}^*$ and $\mu_1 \in \mathbb{Z}_{N_1}^*$ such that $\mu_0^{2^d} = 1 \bmod N_0$ and $\mu_1^{2^d} = 1 \bmod N_1$. As in the proof of Lemma 3.1, we cannot have $\mu_0 = -1 \bmod N_0$ nor $\mu_1 = -1 \bmod N_1$ because pk_0 and pk_1 were chosen to have $J_{N_0}(-1) = J_{N_1}(-1) = -1$, $J_{N_0}(g_0) = J_{N_1}(g_1) = 1$ and the verifier checks that $J_{N_0}(C_0) = J_{N_1}(C_1) = 1$. If

$$\mu_0 = C_0^{\bar{e}_o} \cdot g_0^{-2^{t_z - d} \cdot \bar{z}_o} \cdot \bar{y}_0^{-2^{k+\tau-d}} \bmod N_0$$

is a non-trivial 2^d-th root of 1 in $\mathbb{Z}_{N_0}^*$, it reveals the factorization of N_0 by Lemma 2.4. Similarly, the factorization of $N_1 = p_1 q_1$ is revealed if the product $\mu_1 = C_1^{\bar{e}_o} \cdot g_1^{-2^{t_z - d} \cdot \bar{z}_o} \cdot \bar{y}_1^{-2^{k+\tau-d}} \bmod N_1$ is a non-trivial root of 1 in $\mathbb{Z}_{N_1}^*$.

We now assume that $\mu_0 = 1$ and $\mu_1 = 1$ in (14). Since $\gcd(\bar{e}_o, 2^{k+\tau-d}) = 1$, we can find $\alpha, \beta \in \mathbb{Z}$ such that $\alpha \cdot \bar{e}_o + \beta \cdot 2^{k+\tau-d} = 1$, which turns (14) into

$$\begin{aligned}
C_0 &= g_0^{2^{t_z-d} \cdot \bar{z}_o \cdot \alpha} \cdot \left(\bar{y}_0^\alpha \cdot C_0^\beta\right)^{2^{k+\tau-d}} \bmod N_0 \\
&= g_0^{2^{t_z-d} \cdot \bar{z}_o \cdot \alpha} \cdot \left((\bar{y}_0^\alpha \cdot C_0^\beta)^{2^{\tau-d}}\right)^{2^k} \bmod N_0 \\
&= g_0^{2^{t_z-d} \cdot \bar{z}_o \cdot \alpha \bmod 2^k} \cdot \left((\bar{y}_0^\alpha \cdot C_0^\beta)^{2^{\tau-d}} \cdot g_0^{\lfloor 2^{t_z-d} \cdot \bar{z}_o \cdot \alpha / 2^k \rfloor}\right)^{2^k} \bmod N_0 \\
C_1 &= g_1^{2^{t_z-d} \cdot \bar{z}_o \cdot \alpha} \cdot \left(\bar{y}_1^\alpha \cdot C_1^\beta\right)^{2^{k+\tau-d}} \bmod N_1 \\
&= g_1^{2^{t_z-d} \cdot \bar{z}_o \cdot \alpha} \cdot \left((\bar{y}_1^\alpha \cdot C_1^\beta)^{2^{\tau-d}}\right)^{2^k} \bmod N_1 \\
&= g_1^{2^{t_z-d} \cdot \bar{z}_o \cdot \alpha \bmod 2^k} \cdot \left((\bar{y}_1^\alpha \cdot C_1^\beta)^{2^{\tau-d}} \cdot g_1^{\lfloor 2^{t_z-d} \cdot \bar{z}_o \cdot \alpha / 2^k \rfloor}\right)^{2^k} \bmod N_1
\end{aligned} \tag{16}$$

Since $d \leq \lambda - 1 \leq \tau$, this implies that C_0 and C_1 both decrypt to

$$\bar{m} \triangleq 2^{t_z - d} \cdot \bar{z}_o \cdot \alpha \bmod 2^k = (\bar{z}/2^d) \cdot \bar{e}_o^{-1} \bmod 2^k$$

and the following elements are also computable by the knowledge extractor:

$$(w_0, w_1) \triangleq \Big((\bar{y}_0^{\alpha} \cdot C_0^{\beta})^{2^{\tau-d}} \cdot g_0^{\lfloor 2^{t_z-d} \cdot \bar{z}_o \cdot \alpha / 2^k \rfloor} \bmod N_0, \\ (\bar{y}_1^{\alpha} \cdot C_1^{\beta})^{2^{\tau-d}} \cdot g_1^{\lfloor 2^{t_z-d} \cdot \bar{z}_o \cdot \alpha / 2^k \rfloor} \bmod N_1 \Big).$$

□

We now prove that the protocol has quasi-unique responses assuming the hardness of factoring. The proof exploits the fact that the prover's responses y_0 and y_1 are restricted to have Jacobi symbol 1.

Lemma 4.2. *Under the assumption that factoring N_0 and N_1 is hard, the above Σ-protocol has quasi-unique responses.*

Proof. Let us assume that a prover is able to compute two accepting transcripts $((A_0, A_1), e, (y_0, y_1, z))$, $((A_0, A_1), e, (y_0', y_1', z'))$ s.t. $(y_0', y_1', z') \neq (y_0, y_1, z)$ and

$$A_0 \cdot C_0^e = g_0^z \cdot y_0^{2^{k+\tau}} \bmod N_0 = g_0^{z'} \cdot (y_0')^{2^{k+\tau}} \bmod N_0 \qquad (17)$$
$$A_1 \cdot C_1^e = g_1^z \cdot y_1^{2^{k+\tau}} \bmod N_1 = g_1^{z'} \cdot (y_1')^{2^{k+\tau}} \bmod N_1$$

We note that (17) implies $z = z' \bmod 2^k$ since $A_0 \cdot C_0^e \bmod N_0$ (resp. $A_1 \cdot C_1^e \bmod N_1$) lives in the ciphertext space of pk_0 (resp. pk_1) by construction. Since pk_0 and pk_1 are injective public keys because $\gcd(p_i', q_i') = 1$, each ciphertext has a unique underlying plaintext in $\mathbb{Z}_{2^k}$.

Therefore, we must have $z' = z + \delta \cdot 2^k$ for some $\delta \in \mathbb{Z}_{2^\tau}$, and (17) implies

$$y_0^{2^{k+\tau}} = g_0^{\delta \cdot 2^k} \cdot (y_0')^{2^{k+\tau}} \bmod N_0, \qquad y_1^{2^{k+\tau}} = g_1^{\delta \cdot 2^k} \cdot (y_1')^{2^{k+\tau}} \bmod N_1$$

so that $\left((y_0'/y_0)^{2^\tau} \cdot g_0^{\delta}\right)^{2^k} = 1 \bmod N_0$ and $\left((y_1'/y_1)^{2^\tau} \cdot g_1^{\delta}\right)^{2^k} = 1 \bmod N_1$, meaning that $w_0 \triangleq (y_0'/y_0)^{2^\tau} \cdot g_0^{\delta} \bmod N_0$ and $w_1 \triangleq (y_1'/y_1)^{2^\tau} \cdot g_1^{\delta} \bmod N_1$ are 2^k-th roots of 1 in $\mathbb{Z}_{N_0}^*$ and $\mathbb{Z}_{N_1}^*$, respectively. We claim that at least one of them is non-trivial or reveals a non-trivial 2^τ-th root.

If $\mathcal{A}$ breaks quasi-uniqueness, we have at least one of the inequalities $\delta \neq 0$ or $y_0' \neq y_0 \bmod N_0$ or $y_1' \neq y_1 \bmod N_1$. Let us assume that $(\delta, y_0) \neq (0, y_0')$ (the case $(\delta, y_1) \neq (0, y_1')$ is handled similarly). We first note that $w_0 \neq -1 \bmod N_0$ since $J_{N_0}(-1) = -1$ and $J_{N_0}(g_0) = 1$. Moreover, we cannot have $w_0 = 1 \bmod N_0$ either unless the factorization of N_0 is exposed. This is because:

- If $\delta \neq 0$, we cannot have $g_0^{\delta} = 1 \bmod N_0$ because 2^k divides the order of the pseudo-square g_0 and we chose $k \geq \tau$ while we would have $0 < \delta < 2^\tau$;
- If $\delta = 0$, we have $y_0' \neq y_0 \bmod N_0$, so that $y_0'/y_0 \bmod N_0$ is a non-trivial 2^τ-th root of 1 in $\mathbb{Z}_{N_0}^*$ (which reveals the factorization of N_0 by Lemma 2.4) since the condition $J_{N_0}(y_0) = J_{N_0}(y_0') = 1$ prevents $y_0' = -y_0 \bmod N_0$;
- We cannot have $g_0^{\delta} = (y_0/y_0')^{2^\tau} \neq 1 \bmod N_0$ since $g_0 \notin \mathbb{QR}_{N_0}$ and $\delta \leq 2^\tau - 1$: the largest power of 2 that divides the order of $(y_0/y_0')^{2^\tau}$ is $2^{k-\tau}$ while $2^{k-\tau+1}$ divides the order of g^{δ}.

Hence, unless $y_0/y_0' \bmod N$ is a non-trivial 2^τ-th root of 1, w_0 can only be a non-trivial 2^k-th root of 1 in $\mathbb{Z}_{N_0}^*$, which leads to the factorization of N_0.

We conclude that, if a prover breaks quasi-uniqueness, there is a polynomial-time reduction that computes the factorization of either N_0 or N_1. □

By adapting a result of [11], we can use Lemma 4.1 and Lemma 4.2 to show that the Fiat-Shamir transform provides a simulation-extractable proof of plaintext equalities if factoring N_0 and N_1 is infeasible.

Lemma 4.3 (Adapted from [11, Theorem 3]). *The NIZK argument obtained from the above Σ-protocol by applying the Fiat-Shamir transform (and computing the challenge as $e = H(\mathsf{pk}_0, \mathsf{pk}_1, C_0, C_1, A_0, A_1)$, where $H : \{0,1\}^* \to [0, 2^\lambda - 1]$ is a random oracle) provides simulation-extractability with extraction error $(Q_H+1)/2^{\lambda-1}$ as a proof of knowledge of either: (i) A witness $(m, w_0, w_1) \in \mathbb{Z}_{2^k} \times \mathbb{Z}_{N_0}^* \times \mathbb{Z}_{N_1}^*$ such that $C_0 = g_0^m \cdot w_0^{2^k} \bmod N_0$ and $C_1 = g_1^m \cdot w_1^{2^k} \bmod N_1$; or (ii) The factorization of at least one of the moduli N_0 and N_1. Concretely, if the prover outputs a valid non-trivial proof (i.e., such that $(x^\star, \pi^\star) \notin \mathcal{T}$ in Definition 2.8) with probability ε, the extractor $\mathcal{E}$ succeeds with probability at least*

$$\frac{1}{Q_H} \cdot \left(\varepsilon - \frac{Q_H+1}{2^{\lambda-1}}\right)^2,$$

if Q_H denotes the number of random oracle queries made by $\mathcal{A}$. (The proof is given in Appendix B.1.)

5 Applying the Naor-Yung Paradigm

In this section, we apply the Naor-Yung transformation to build an IND-CCA2 version of the scheme where ciphertexts contain built-in ciphertexts of an IND-CPA system that supports homomorphic additions.

In the proof of CCA2-security, we need to be able to sample uniformly random pseudo-squares in $\mathbb{J}_N \setminus \mathbb{QR}_N$ without knowing the factorization of N. For this reason, we additionally impose the constraints $p = 5 \bmod 12$ and $q = 7 \bmod 12$ on the modulus $N = pq$.

In order for the security proof to work out, we need to slightly depart from the Naor-Yung paradigm in that the secret key sk_b used by the decryption algorithm is determined by a random bit $b \in \{0,1\}$ chosen at key generation time (instead of being arbitrarily fixed to sk_0 in the usual application of Naor-Yung).

Keygen$(1^\lambda, 1^k)$**:** Given a security parameter λ and a message length $k \in \mathsf{poly}(\lambda)$,

1. Choose RSA moduli $N_0 = p_0 q_0$ and $N_1 = p_1 q_1$ for primes of the form
$$p_0 = 2^k p_0' + 1, \quad q_0 = 2q_0' + 1, \quad p_1 = 2^k p_1' + 1, \quad q_1 = 2q_1' + 1,$$
where p_0', q_0', p_1', q_1' are odd integers such that $p_0', q_0', p_1', q_1' > 2^{l(\lambda)}$, for some polynomial $l : \mathbb{N} \to \mathbb{N}$, and $\gcd(p_0', q_0') = \gcd(p_1', q_1') = 1$. In addition, we require $p_0, p_1 = 5 \bmod 12$ and $q_0, q_1 = 7 \bmod 12$ so as to have
$$(3 \mid p_0) = (3 \mid q_0) = (3 \mid p_1) = (3 \mid q_1) = -1.$$

2. Choose random elements $g_0 \in \mathbb{J}_{N_0}$ and $g_1 \in \mathbb{J}_{N_1}$ that have maximal orders $\lambda(N_0) = 2^k p'_0 q'_0$ and $\lambda(N_1) = 2^k p'_1 q'_1$ in $\mathbb{Z}^*_{N_0}$ and $\mathbb{Z}^*_{N_1}$, respectively. Choose a parameter τ such that $k \geq \tau \geq \lambda - 1$.
3. Choose a hash function $H : \{0,1\}^* \to [0, 2^\lambda - 1]$ modeled as a random oracle, which will be used to apply the Fiat-Shamir heuristic.
4. Choose a random $b \xleftarrow{R} \{0,1\}$.

Return the key pair $(PK, SK) = \big((\mathsf{pk}_0, \mathsf{pk}_1, H), (b, \mathsf{sk}_b)\big)$, where

$$\mathsf{pk}_0 := (N_0,\ k,\ \tau,\ g_0)\,, \qquad\qquad \mathsf{pk}_1 := (N_1,\ k,\ \tau,\ g_1)$$

and $\mathsf{sk}_b := (p_b, q_b, p'_b, q'_b)$.

Encrypt(PK, m)**:** Given $PK = (\mathsf{pk}_0, \mathsf{pk}_1, H)$ and a message $m \in \mathbb{Z}_{2^k}$,

1. Choose $s_0 \xleftarrow{R} \mathbb{J}_{N_0}$, $s_1 \xleftarrow{R} \mathbb{J}_{N_1}$ and compute

$$C_0 = g_0^m \cdot s_0^{2^{k+\tau}} \bmod N_0, \qquad\qquad C_1 = g_1^m \cdot s_1^{2^{k+\tau}} \bmod N_1$$

2. Generate a NIZK proof of plaintext equality using the Σ-protocol of Section 4.1. This proof is a tuple $\boldsymbol{\pi}_{\mathsf{NY}} = (e, z, y_0, y_1) \in [0, 2^\lambda - 1] \times \mathbb{Z}_{2^{k+\tau}} \times \mathbb{J}_{N_0} \times \mathbb{J}_{N_1}$, where

$$e = H\big(\mathsf{pk}_0, \mathsf{pk}_1, C_0, C_1, A_0, A_1\big)$$

with $A_0 = C_0^{-e} \cdot g_0^z \cdot y_0^{2^{k+\tau}} \bmod N_0$ and $A_1 = C_1^{-e} \cdot g_1^z \cdot y_1^{2^{k+\tau}} \bmod N_1$.

Output the final ciphertext $C = (C_0, C_1, \boldsymbol{\pi}_{\mathsf{NY}})$.

Decrypt(SK, C)**:** Given a ciphertext $C = (C_0, C_1, \boldsymbol{\pi}_{\mathsf{NY}})$ and the secret key $SK = (b, \mathsf{sk}_b = (p_b, q_b, p'_b, q'_b))$, do the following:

1. Return $\perp$ if $\boldsymbol{\pi}_{\mathsf{NY}}$ does not properly verify.
2. Compute $C_{0,b} = C_b^{p'_b} \bmod p_b$.
3. Compute $m \in \mathbb{Z}_{2^k}$ such that $C_{0,b} = \big(g_b^{p'_b}\big)^m \bmod p_b$ using the Pohlig-Hellman algorithm.

5.1 Security

The proof of Theorem 5.1 requires the bit b to be independent of the adversary's view. To see why, consider a variant where b is fixed to 0 and imagine an adversary that can factor in polynomial time. The adversary is then able to create a ciphertext with $C_0 = g_0^m \cdot s_0^{2^{k+\tau}} \bmod N_0$ and $C_1 = \mu_1 \cdot g_1^m \cdot s_1^{2^{k+\tau}} \bmod N_1$ for some non-trivial square root of unity $\mu_1 \in \mathbb{Z}^*_{N_1}$ such that $\mu_1 = -1 \bmod p_1$ and $\mu_1 = 1 \bmod q_1$. Then, if $e = H\big(\mathsf{pk}_0, \mathsf{pk}_1, C_0, C_1, A_0, A_1\big)$ is even (the adversary can try different pairs (A_0, A_1) until this happens), it can generate a valid fake proof of plaintext equality $\boldsymbol{\pi}_{\mathsf{NY}}$ although C_0 and C_1 do not decrypt to the same plaintext. This allows the adversary to distinguish Game_1 (which decrypts using sk_1 instead of sk_0) from Game_0 without implying a reduction from the factoring assumption. The reason is that, from the fake proof $\boldsymbol{\pi}_{\mathsf{NY}}$, the knowledge extractor can only extract the factorization of N_1. Unfortunately, the reduction needs

sk_1 to answer decryption queries, which prevents it from relying on the hardness of factoring N_1. Similarly, in Game_5, we would not be able to bound $\Pr[F_5]$ by relying on the hardness of factoring if b was fixed to 0. Indeed, an adversary that would be able to factor N_1 could tamper with the NIZK proof of plaintext equality in the challenge ciphertext and replace $y_1 \in \mathbb{Z}^*_{N_1}$ by $y'_1 = \mu_1 \cdot y_1 \bmod N_1$, where $\mu_1 = -1 \bmod p_1$ and $\mu_1 = 1 \bmod q_1$. While this would break the quasi-uniqueness of the Σ-protocol, it would only provide the reduction of Lemma 5.3 with the factorization of N_1, which it already knows.

To address this issue, we randomize the choice of which secret key among sk_0 and sk_1 is used by the real decryption algorithm. By doing so, we can argue that, with probability 1/2, a fake proof of plaintext equality $\boldsymbol{\pi}_{\mathsf{NY}}$ provides the reduction with the factorization that it does not already know.

Theorem 5.1. *The scheme provides IND-CCA2 security in the random oracle model under the k-QR assumption. For any PPT adversary $\mathcal{A}$, we can construct either a PPT factoring algorithm $\mathcal{B}_0$ or a PPT k-QR distinghuisher $\mathcal{B}_1$ such that*

$$\mathbf{Adv}^{\mathrm{cca2}}_{\mathcal{A}}(\lambda) \leq 2 \cdot \sqrt{2Q \cdot Q_H \cdot \mathbf{Adv}^{\mathrm{fact}}_{\mathcal{B}_0}(\lambda)} + (k+1) \cdot \mathbf{Adv}^{k\text{-}\mathsf{QR}}_{\mathcal{B}_1}(\lambda) + \frac{Q_H + 1}{2^{\lambda-2}}$$

Proof. We proceed with a sequence of games. For each index i, we denote by W_i the event that the adversary wins in Game_i.

Game_0: This is the real IND-CCA2 experiment where the challenger encrypts the message m_β. Namely, the challenger runs the adversary $\mathcal{A}$ on input of a public key containing $(\mathsf{pk}_0, \mathsf{pk}_1)$ and answers all decryption queries using sk_b, where $b \in \{0,1\}$ is part of the secret key SK. In the challenge phase, $\mathcal{A}$ chooses messages $(m_0, m_1) \in \mathbb{Z}_{2^k} \times \mathbb{Z}_{2^k}$ and obtains an encryption of m_β, where $\beta \xleftarrow{R} \{0,1\}$ is a random bit chosen by the challenger. When $\mathcal{A}$ halts, it outputs a bit $\beta' \in \{0,1\}$ and we denote by W_0 the event that $\beta' = \beta$.

Game_1: In this game, we modify the decryption oracle and answer all decryption queries using $\mathsf{sk}_{1-b} = (p_{1-b}, q_{1-b}, p'_{1-b}, q'_{1-b})$ instead of sk_b. From $\mathcal{A}$'s view, Game_1 is identical to Game_0 until the event F_1 that $\mathcal{A}$ makes a decryption query $C = (C_0, C_1, \boldsymbol{\pi}_{\mathsf{NY}})$ such that C_0 and C_1 decrypt to distinct messages. However, event F_1 would break the soundness of the proof of plaintext equalities and the assumption that factoring is hard. Lemma 5.2 shows that we can build a reduction $\mathcal{B}_0$ such that

$$|\Pr[W_1] - \Pr[W_0]| \leq \Pr[F_1] \leq \left(\sqrt{2Q \cdot Q_H \cdot \mathbf{Adv}^{\mathrm{fact}}_{\mathcal{B}_0}(\lambda)} + \frac{1}{2^{\lambda-1}}\right),$$

Game_2: This game is like Game_1 except that, in the challenge phase, the challenger computes $\boldsymbol{\pi}_{\mathsf{NY}}$ as a simulated NIZK proof, which entails running the HVZK simulator of the Σ-protocol and programming H. The perfect HVZK property ensures that the simulated $\boldsymbol{\pi}^\star_{\mathsf{NY}}$ is distributed as a real proof. Hence, Game_2 can only be distinguished from Game_1 in the unlikely event F_2 that the simulation fails because it has to program H on an input where it was previously defined. In the ROM, this occurs with probability $\Pr[F_2] \leq Q_H/2^\lambda$. We then have $|\Pr[W_2] - \Pr[W_1]| \leq \Pr[F_2] \leq Q_H/2^\lambda$.

Game_3: In this game, we change the distribution of the public key and replace $g_b \in \mathbb{Z}^*_{N_b}$ (which lives in $\mathbb{J}_{N_b} \setminus \mathbb{QR}_{N_b}$ in Game_2) by a random 2^k-th residue in pk_b. A straightforward reduction shows that Game_3 is computationally indistinguishable from Game_2 under the Gap 2^k-res assumption. The reduction of [5, Theorem 5] implies that this would contradict the k-QR assumption. We then have $|\Pr[W_3] - \Pr[W_2]| \leq \frac{(k+1)}{2} \cdot \mathbf{Adv}^{k\text{-}\mathsf{QR}}_{\mathcal{B}_1}(\lambda)$.

Game_4: We modify the challenge ciphertext and replace $C^\star_b$ by an encryption of $m_{1-\beta}$. Namely, the challenger chooses $s_0 \xleftarrow{R} \mathbb{J}_{N_0}$, $s_1 \xleftarrow{R} \mathbb{J}_{N_1}$ and computes

$$C^\star_b = g_b^{m_{1-\beta}} \cdot s_b^{2^{k+\tau}} \bmod N_b, \qquad C^\star_{1-b} = g_{1-b}^{m_\beta} \cdot s_{1-b}^{2^{k+\tau}} \bmod N_{1-b}$$

while the proof $\boldsymbol{\pi}^\star_{\mathsf{NY}}$ is simulated as in Game_2. Since g_b is a 2^k-th residue in $\mathbb{Z}^*_{N_b}$, this change does not affect the distribution of the challenge ciphertext whatsoever. Indeed, regardless of the value of the bit $\beta \in \{0,1\}$, the distribution $\{C^\star_b = g_b^{m_\beta} \cdot s_b^{2^{k+\tau}} \bmod N_b \mid s_b \xleftarrow{R} \mathbb{J}_{N_b}\}$ is exactly identical to the distribution $\{C^\star_b = s_b^{2^{k+\tau}} \bmod N_b \mid s_b \xleftarrow{R} \mathbb{J}_{N_b}\}$, so that $\Pr[W_4] = \Pr[W_5]$.

Game_5: We change again the distribution of PK and restore $g_b \in \mathbb{Z}^*_{N_b}$ back to its initial distribution, by choosing uniformly in $\mathbb{J}_{N_b} \setminus \mathbb{QR}_{N_b}$. By the same argument as in the transition from Game_2 to Game_3, we know that Game_5 is indistinguishable from Game_4 under the k-QR assumption. We have

$$|\Pr[W_5] - \Pr[W_4]| \leq \tfrac{k+1}{2} \cdot \mathbf{Adv}^{k\text{-}\mathsf{QR}}_{\mathcal{B}_1}(\lambda).$$

In order to bound $\Pr[W_5]$, we define F_5 to be the event that $\mathcal{A}$ makes a decryption query $(C_0, C_1, \boldsymbol{\pi}_{\mathsf{NY}})$ such that $\boldsymbol{\pi}_{\mathsf{NY}}$ verifies but C_0 and C_1 decrypt to distinct plaintexts in Game_5. We then have $\Pr[W_5] \leq \Pr[W_5 \mid \neg F_5] + \Pr[F_5]$.

We first claim that $\Pr[W_5 \mid \neg F_5] = 1/2$, so that the advantage of $\mathcal{A}$ in Game_5 is at most $|\Pr[W_5] - 1/2| \leq \Pr[F_5]$. Indeed, the challenge is of the form

$$C^\star_0 = g_0^{m_1} \cdot s_0^{2^{k+\tau}} \bmod N_0, \qquad C^\star_1 = g_1^{m_0} \cdot s_1^{2^{k+\tau}} \bmod N_1 \qquad \text{if} \qquad b = \beta$$

$$C^\star_0 = g_0^{m_0} \cdot s_0^{2^{k+\tau}} \bmod N_0, \qquad C^\star_1 = g_1^{m_1} \cdot s_1^{2^{k+\tau}} \bmod N_1 \qquad \text{if} \qquad b \neq \beta$$

Therefore, the only information revealed by the challenge ciphertext $(C^\star_0, C^\star_1, \boldsymbol{\pi}^\star_{\mathsf{NY}})$ is whether the equality $b = \beta$ holds. However, conditionally on $\neg F_5$, the secret key bit $b \in \{0,1\}$ remains independent of $\mathcal{A}$'s view throughout the entire game. Consequently, the same holds for $\beta \in \{0,1\}$ and we have $\Pr[\beta' = \beta \mid \neg F_5] = 1/2$.

As for $\Pr[F_5]$, Lemma 5.3 gives a reduction $\mathcal{B}_2$ such that

$$\Pr[F_5] \leq \sqrt{2Q \cdot Q_H \cdot \mathbf{Adv}^{\mathrm{fact}}_{\mathcal{B}_2}(\lambda)} + \frac{Q_H + 1}{2^{\lambda-1}}$$

since event F_5 would contradict the assumption that factoring is hard by the simulation-extractabilty of $\boldsymbol{\pi}_{\mathsf{NY}}$. □

Lemma 5.2. *In* Game_1, F_1 *occurs with negligible probability if factoring is hard. There is a PPT algorithm* $\mathcal{B}_0$ *such that* $\Pr[F_1] \leq \sqrt{2Q \cdot Q_H \cdot \mathbf{Adv}^{\mathrm{fact}}_{\mathcal{B}_0}(\lambda)} + \frac{1}{2^{\lambda-1}}$.

Proof. The proof is identical to the proof of Lemma 5.3 with the difference that the adversary never gets to see a simulated NIZK proof. □

Lemma 5.3. *In* Game_5, F_5 *occurs with negligible probability if factoring is hard. There is a PPT algorithm* $\mathcal{B}_2$ *such that* $\Pr[F_5] \leq \sqrt{2Q \cdot Q_H \cdot \mathbf{Adv}^{\mathrm{fact}}_{\mathcal{B}_2}(\lambda)} + \frac{Q_H+1}{2^{\lambda-1}}$. (The proof is given in Appendix B.2.)

5.2 Comparisons

Comparison with [24]. In Appendix A.2, we provide a detailed comparison between the above scheme and the one obtained by applying Naor-Yung to the modified JL scheme of Xue *et al.* [24]. We show that the above construction provides shorter ciphertexts (by about 640 bits for $\lambda = 128$) and a significantly faster encryption algorithm.

As in our setting, applying Naor-Yung to [24] would also require to randomize the choice of the secret key used by the decryption algorithm among sk_0 and sk_1. The reason is that their proof of plaintext equality provides soundness under the assumption that the prover is not able to break the Strong RSA assumption [1] (in the subgroup of 2^k-th residues) for one of the two moduli. However, the reduction always needs the factorization of one of these two moduli in order to simulate the decryption oracle and it can only rely on the Strong RSA assumption for the modulus that it does not know the factorization of.

As a side note, applying Naor-Yung to [24] would require the ability to publicly generate pseudo-squares (as the reduction does in the proof of our Lemma 5.3 by exploiting the fact that $3|q'$) with overwhelming probability. Since their modulus N should be chosen in such a way that the order of the subgroup of 2^k-th residues has no small prime factor, it is not clear how to do this.[4]

Comparison with [7]. We can compare our application of Naor-Yung with the one that can be obtained by applying Catalano *et al.*'s proof of plaintext equalities [7] to a variant of the original JL scheme where the decryption algorithm only outputs the least $k - \lambda$ bits of the decrypted plaintexts (because applying Naor-Yung to their Σ-protocol would only guarantee plaintext equalities for the $k-\lambda$ LSBs). We note that, due to the limitation of their Σ-protocols, they need a longer modulus for the same number $k' = k-\lambda$ of actual plaintext bits (we have $k' = k$ in our setting): for security reasons, they need $k = k' + \lambda < \frac{1}{4} \cdot \log N - \lambda$ (equivalently, $\log N > 4(k' + 2\lambda)$) while we only need $\log N > 4(k' + \lambda)$. Their Naor-Yung ciphertext would consist of two JL ciphertexts of $\log N$ bits and a proof of plaintext equality $(e, (z, y_0, y_1)) \in [0, 2^{\lambda-1}] \times \mathbb{Z}_{2^{k'+\lambda}} \times \mathbb{Z}^*_{N_0} \times \mathbb{Z}^*_{N_1}$, which would cost $2(\log N_0 + \log N_1) + k' + 2\lambda > 17(k' + 2\lambda)$ bits. Our ciphertext only takes $2(\log N_0 + \log N_1) + k' + 2\lambda > 17(k' + \lambda) + \lambda$ bits when we set $\tau \geq \lambda$ since we can afford smaller moduli N_0 and N_1 for the same k'.

[4] One option is to rely on the assumption that factoring remains hard when a pseudo-square is explicitly given.

References

1. N. Baric and B. Pfitzmann. Collision-Free Accumulators and Fail-Stop Signature Schemes Without Trees. In *Eurocrypt 1997*, 1997.
2. C. Bartoli and I. Cascudo. On Sigma-Protocols and (packed) Black-Box Secret Sharing Schemes. In *PKC*, 2024.
3. M. Bellare, D. Hofheinz, and S. Yilek. Possibility and impossibility results for encryption and commitment secure under selective opening. In *Eurocrypt*, 2009.
4. M. Bellare and G. Neven. Multi-signatures in the plain public key model and a general forking lemma. In *ACM-CCS*, 2006.
5. F. Benhamouda, J. Herranz, M. Joye, and B. Libert. Efficient cryptosystems from 2^k-th power residue symbols. *Journal of Cryptology*, 30(2), 2017.
6. G. Castagnos, F. Laguillaumie, and I. Tucker. Threshold linearly homomorphic encryption on $\mathbb{Z}/(2^k\mathbb{Z})$. In *Asiacrypt*, 2022.
7. D. Catalano, M. Di Raimondo, D. Fiore, and I. Giacomelli. MonZa: Fast Maliciously Secure Two Party Computation on $\mathbb{Z}_{2^k}$. In *PKC*, 2020.
8. Y.-H. Chen and Y. Lindell. Optimizing and Implementing Fischlin's Transform for UC-Secure Zero-Knowledge. *IACR Commun. Cryptol.*, 1(2), 2024.
9. J. Devevey, B. Libert, K. Nguyen, T. Peters, and M. Yung. Non-interactive CCA2-secure threshold cryptosystems: Achieving adaptive security in the standard model without pairings. In *PKC*, 2021.
10. J. Devevey, B. Libert, and T. Peters. Rational Modular Encoding in the DCR Setting: Non-interactive Range Proofs and Paillier-Based Naor-Yung in the Standard Model. In *PKC*, 2022.
11. S. Faust, M. Kohlweiss, G. Marson, and D. Venturi. On the non-malleability of the Fiat-Shamir transform. In *Indocrypt*, 2012.
12. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Crypto*, 1986.
13. M. Fischlin. Communication-efficient non-interactive proofs of knowledge with online extractors. In *Crypto*, 2005.
14. M. Fischlin and R. Fischlin. The representation problem based on factoring. In *CT-RSA*, 2002.
15. S. Goldwasser and S. Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In *STOC*, 1982.
16. B. Hemenway, B. Libert, R. Ostrovsky, and D. Vergnaud. Lossy encryption: Constructions from general assumptions and efficient selective opening chosen ciphertext security. In *Asiacrypt*, 2011.
17. M. Joye and B. Libert. Efficient cryptosystems from 2^k-th power residue symbols. In *Eurocrypt*, 2013.
18. M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *STOC*, 1990.
19. T. Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In *Crypto*, 1992.
20. P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Eurocrypt*, 1999.
21. S. Pohlig and M. Hellman. An improved algorithm for computing logarithms over GF(p) and its cryptographic significance. *IEEE Trans. Inf. Theory*, 24(1), 1978.
22. D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *J. of Cryptology*, 2000.

23. A. Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *FOCS*, 1999.
24. H. Xue, M.-H. Au, M. Liu, K.-Y. Chan, H. Cui, X. Xie, T.-H. Yuen, and C. Zhang. Efficient Multiplicative-to-Additive Function from Joye-Libert Cryptosystem and Its Application to Threshold ECDSA. In *ACM-CCS*, 2023.
25. A. Young and M. Yung. Questionable encryption and its applications. In *Mycrypt*, 2005.

A The Proof of Plaintext Knowledge of Xue *et al.*

Xue *et al.* [24] consider a variant of the JL scheme where messages $m \in \mathbb{Z}_{2^k}$ are encrypted as $C = g^m \cdot h^r \bmod N$, where $h \in \mathbb{Z}_{N^*}$ is a public generator[5] of the subgroup of 2^k-th residues and r is chosen uniformly $\mathbb{Z}_N$. Decryption proceeds as in the original scheme, by exploiting the fact that g has maximal order in $\mathbb{Z}_N^*$.

A.1 Description

As described in [24, Section 4], their proof of plaintext knowledge assumes that the plaintext is contained in some interval $[0, B]$. In the verification algorithm, the condition $z \in [0, 2^{2\lambda}B]$ is used to demonstrate that the plaintext belongs to a specific range (i.e., the Σ-protocol actually provides a range proof with a slack), which we do not consider in this paper.

Inputs: The prover P has a witness $(m, r) \in [0, B] \times \mathbb{Z}_N$ such that

$$C = g^m \cdot h^r \bmod N.$$

The verifier V only knows C. P and V first send C in the subgroup of 2^k-th residues by computing $c = C^{2^k} \bmod N$.

1. P chooses $u \xleftarrow{R} [0, 2^{2\lambda}B]$, $v \xleftarrow{R} [0, 2^{2\lambda}N]$ and computes
$$a = g^{2^k \cdot u} \cdot h^{2^k \cdot v} \bmod N$$
which is sent to V
2. V sends a random challenge $e \xleftarrow{R} [0, 2^\lambda - 1]$ to P
3. P computes the following response components over $\mathbb{Z}$
$$z = u + e \cdot m$$
$$y = v + e \cdot r$$
4. V accepts if $J_N(c) = J_N(a) = 1$,
$$a \cdot c^e = g^{2^k \cdot z} \cdot h^{2^k \cdot y} \bmod N \tag{18}$$
and $z \in [0, 2^{2\lambda}B]$. If any of these conditions does not hold, V rejects.

[5] They assume that $\gcd(p', q') = 1$ to make sure that the subgroup of 2^k-th residues is cyclic. Moreover, the computational soundness of their proof system requires that p' and q' have no prime factor smaller than 2^λ.

A.2 More Detailed Comparisons with [24]

In order to provide a fair comparison with our construction, we assume that the plaintext m is as large as possible and that $B = 2^k - 1$. When the protocol is made non-interactive using the Fiat-Shamir heuristic, the prover only needs to send (e, z, y), which takes $5\lambda + k + \log N$ bits (namely, λ bits for e, $2\lambda + k$ bits for z and $2\lambda + \log N$ bits for y).

In the Σ-protocol of Section 3.1, the prover only needs to send $2\lambda + k + \log N$ bits as the prover's response $z \in \mathbb{Z}_{2^{k+\tau}}$, $y \in \mathbb{Z}_N^*$ fits in $k + \lambda + \log N$ bits when $\tau \geq \lambda$. For $\lambda = 128$, we thus save 48 bytes for a proof of plaintext knowledge.

When it comes to applying the Naor-Yung transform, adapting the construction of Xue *et al.* so as to prove plaintext equalities requires a prover sending (e, z, y_1, y_2), where e costs λ bits; z takes $2\lambda + k$ bits while $y_1 \in [0, 2^{2\lambda} N_1]$ and $y_2 \in [0, 2^{2\lambda} N_2]$ require at least $2\lambda + \min_{b \in \{0,1\}}(\log N_b)$ bits each. This leads to a proof of plaintext equality $\boldsymbol{\pi}_{\mathsf{NY}}$ comprised of $7\lambda + k + \log N_0 + \log N_1$ bits.

Our Σ-protocol of Section 4.1 only requires the prover to send $2\lambda + k + \log N_0 + \log N_1$ bits, which saves $5\lambda \approx 640$ bits in terms of communication at the 128-bit security level. Moreover, it does not introduce any other intractability assumption than the one that underlies the IND-CPA security of the scheme. We finally remark that the CCA2-secure encryption scheme described in Section 5 has a faster encryption algorithm than the one obtained by applying the Naor-Yung transform to the scheme of [24] because the underlying IND-CPA system is faster. Indeed, [24] requires full-fledged modular exponentiations $h^r \bmod N$ with 3072-bit exponents $r \in \mathbb{Z}_N$. The scheme of Section 2.5 only requires $k + \tau \approx k + \lambda < \frac{1}{4} \cdot \log N$ modular squarings, where the last inequality holds because k must be smaller than $\frac{1}{4} \cdot \log N - \lambda$ for security reasons. If an exponentiation $h^r \bmod N$ is computed via a standard square-and-multiply algorithm, it is roughly 6 times (on average assuming that r has Hamming weight $\approx \log N/2$) as expensive as computing $s^{2^{k+\tau}} \bmod N$ for relevant parameters in our setting.

B Deferred Proofs

B.1 Proof of Lemma 4.3

Proof. If a prover $\mathcal{A}$ has non-negligible probability ε of coming up with a valid non-trivial proof (i.e., that differs from the pair $(x, \boldsymbol{\pi})$ obtained from the simulator $\mathcal{S}_2$), we can build an extractor $\mathcal{E}$ that computes a witness satisfying either condition (i) or condition (ii) in the statement of the lemma. The extractor proceeds by rewinding $\mathcal{A}$ at most once and applying Lemma 2.9.

Algorithm $\mathcal{E}$ runs algorithm $F_{\mathcal{A}'}(1^\lambda)$ of Figure 1 using random coins ρ. When executing $\mathcal{A}$, $F_{\mathcal{A}'}$ internally runs a wrapper $\mathcal{A}'$ for $\mathcal{A}$ which takes care of simulating $\mathcal{S}_1$ and $\mathcal{S}_2$ using the randomness ρ supplied by $F_{\mathcal{A}'}$ together with random hash values $h_1, \ldots, h_{Q_H} \xleftarrow{R} [0, 2^\lambda - 1]$. During the simulation, $\mathcal{A}'$ uses ρ and $h_1, \ldots, h_{Q_H}$ to simulate random oracle queries via $\mathcal{S}_1$. Simulating $\mathcal{S}_2$ consists in providing $\mathcal{A}$ with a simulated proof $\boldsymbol{\pi} = (e, z, y_0, y_1)$ for a statement consisting

of public keys $\mathsf{pk}_0 = (N_0, k, \tau, g_0)$ and $\mathsf{pk}_1 = (N_1, k, \tau, g_1)$ and a pair of ciphertexts $(C_0, C_1) \in \mathbb{J}_{N_0} \times \mathbb{J}_{N_1}$ of $\mathcal{A}$'s choice. To simulate this proof, $\mathcal{A}'$ chooses $e \xleftarrow{R} [0, 2^\lambda - 1]$, $z \xleftarrow{R} \mathbb{Z}_{2^{k+\tau}}$, $y_0 \xleftarrow{R} \mathbb{J}_{N_0}$, $y_1 \xleftarrow{R} \mathbb{J}_{N_1}$, computes

$$A_0 = C_0^{-e} \cdot g_0^z \cdot y_0^{2^{k+\tau}} \bmod N_0, \qquad A_1 = C_1^{-e} \cdot g_1^z \cdot y_1^{2^{k+\tau}} \bmod N_1$$

and programs $e = H\big(\mathsf{pk}_0, \mathsf{pk}_1, C_0, C_1, A_0, A_1\big)$. If H was already defined for this input, the wrapper $\mathcal{A}'$ aborts the execution of $\mathcal{A}$ and uses $(0, \perp)$ as a replacement for $\mathcal{A}$'s output. This occurs with probability $Q_H/2^\lambda$ since the simulated pair (A_0, A_1) has min-entropy at least $\sum_{i\in\{0,1\}} \log p'_i q'_i > \lambda$ due to the randomness of $y_0^{2^{k+\tau}} \bmod N_0$ and $y_1^{2^{k+\tau}} \bmod N_1$.

A successful adversary $\mathcal{A}$ is then expected to come up with a statement $x^\star = (\mathsf{pk}_0, \mathsf{pk}_1, C_0^\star, C_1^\star)$, which comprised of public keys $(\mathsf{pk}_0, \mathsf{pk}_1)$ and ciphertexts $(C_0^\star, C_1^\star)$ of $\mathcal{A}$'s choice, together with a proof of plaintext equality $\boldsymbol{\pi}^\star = (e^\star, z^\star, y_0^\star, y_1^\star) \in [0, 2^\lambda - 1] \times \mathbb{Z}_{2^{k+\tau}} \times \mathbb{J}_{N_0} \times \mathbb{J}_{N_1}$. At this point, $\mathcal{A}'$ aborts and replaces $\mathcal{A}$'s output by $(0, \perp)$ if $\boldsymbol{\pi}^\star$ is not a valid proof or if $(x^\star, \boldsymbol{\pi}^\star) \in \mathcal{T}$. Otherwise, it computes

$$A_0^\star = C_0^{\star -e^\star} \cdot g_0^{z^\star} \cdot {y_0^\star}^{2^{k+\tau}} \bmod N_0, \qquad A_1^\star = C_1^{\star -e^\star} \cdot g_1^{z^\star} \cdot {y_1^\star}^{2^{k+\tau}} \bmod N_1$$

which satisfy $e^\star = H\big(\mathsf{pk}_0, \mathsf{pk}_1, C_0^\star, C_1^\star, A_0^\star, A_1^\star\big)$.

If $\big(\mathsf{pk}_0, \mathsf{pk}_1, A_0^\star, A_1^\star\big) = \big(\mathsf{pk}_0, \mathsf{pk}_1, A_0, A_1\big)$, then $\mathcal{E}$ can stop the execution of $F_{\mathcal{A}'}$ after line 3 since it has broken the quasi-uniqueness of the Σ-protocol. Indeed, since

$$e = H\big(\mathsf{pk}_0, \mathsf{pk}_1, C_0, C_1, A_0, A_1\big) = H\big(\mathsf{pk}_0, \mathsf{pk}_1, C_0^\star, C_1^\star, A_0^\star, A_1^\star\big) = e^\star,$$

$F_{\mathcal{A}'}$ has obtained two transcripts $(A_0^\star, A_1^\star, e^\star, z^\star, y_0^\star, y_1^\star)$, $(A_0^\star, A_1^\star, e^\star, z, y_0, y_1)$ that differ in $(z, y_0, y_1) \neq (z, y_0^\star, y_1^\star)$ without even rewinding $\mathcal{A}$. By Lemma 4.2, this directly reveals a prime factor of either N_0 or N_1 to $\mathcal{E}$.

If $\big(\mathsf{pk}_0, \mathsf{pk}_1, A_0^\star, A_1^\star\big) \neq \big(\mathsf{pk}_0, \mathsf{pk}_1, A_0, A_1\big)$, we know that the hash value $e^\star = H\big(\mathsf{pk}_0, \mathsf{pk}_1, C_0^\star, C_1^\star, A_0^\star, A_1^\star\big)$ was not programmed by the NIZK simulator. Then, $\mathcal{A}'$ aborts and replaces $\mathcal{A}$'s output by $(0, \perp)$ if $\mathcal{A}$ did not query H on the relevant input $(\mathsf{pk}_0, \mathsf{pk}_1, C_0^\star, C_1^\star, A_0^\star, A_1^\star)$. Since H is a random oracle, this is only possible with probability $1/2^\lambda$. Otherwise, by replaying $\mathcal{A}'$ using the forking lemma and exploiting Lemma 4.1, $F_{\mathcal{A}'}$ will be able to compute a witness.

When applying the forking lemma, we define $J \in [0, Q_H]$ to be the index of the random oracle query $H(\mathsf{pk}_0, \mathsf{pk}_1, C_0^\star, C_1^\star, A_0^\star, A_1^\star)$ among H-queries and we set $J = 0$ if $\mathcal{A}'$ aborted. If $J \geq 1$ (we call *acc* the probability that this happens), $F_{\mathcal{A}'}$ runs $\mathcal{A}'$ a second time using the same random tape as in its first execution. In the second execution, $\mathcal{A}'$ forks on H at the J-th query $H(\mathsf{pk}_0, \mathsf{pk}_1, C_0^\star, C_1^\star, A_0^\star, A_1^\star)$ and, from this point forward, uses fresh random oracle values $h'_J, \ldots, h'_{Q_H} \xleftarrow{R} [0, 2^\lambda - 1]$. The first $J - 1$ random oracle queries obtain the same responses $h_1, \ldots, h_{J-1}$ as in the first execution. We denote the two related executions by

$\big(J, (x^\star, (A_0^\star, A_1^\star), e^\star, (z^\star, y_0^\star, y_1^\star))\big) \leftarrow \mathcal{A}'(1^\lambda, (h_1, \ldots, h_{Q_H}); \rho_{\mathcal{A}})$, and

$$\big(J', (x^{\star\star}, (A_0^{\star\star}, A_1^{\star\star}), e^{\star\star}, (z^{\star\star}, y_0^{\star\star}, y_1^{\star\star}))\big) \leftarrow \mathcal{A}'(1^\lambda, (h_1, \ldots, h_{J-1}, h'_J, \ldots, h'_{Q_H}); \rho_{\mathcal{A}}).$$

Lemma 2.9 tells us that, with probability

$$\mathrm{frk} \geq acc \cdot \left(\frac{acc}{Q_H} - \frac{1}{2^\lambda}\right),$$

the two executions are such that $J = J' \geq 1$ and $e^\star = h_J \neq h_{J'} = e^{\star\star}$. In this case, we must have $x^\star = x^{\star\star}$ (so that the statement is $(\mathsf{pk}_0, \mathsf{pk}_1, C_0^\star, C_1^\star)$ in both executions) and $(A_0^\star, A_1^\star) = (A_0^{\star\star}, A_1^{\star\star})$. This provides $F_{\mathcal{A}'}$ with two Σ-protocol transcripts for the same $(\mathsf{pk}_0, \mathsf{pk}_1, C_0^\star, C_1^\star, A_0^\star, A_1^\star)$ and distinct challenges $e^\star \neq e^{\star\star}$. Using these outputs of $F_{\mathcal{A}'}$, algorithm $\mathcal{E}$ can either extract a witness showing plaintext equalities or a non-trivial factor of N_0 or N_1, by the same arguments as in the proof of Lemma 4.1.

We now assess $\mathcal{E}$'s success probability and assume that $acc \geq (Q_H+1)/2^\lambda \triangleq \nu$ so as to have $\nu^2 - acc \cdot \nu \leq 0$. This implies

$$\begin{aligned}\mathrm{frk} \geq \left(\frac{acc^2}{Q_H} - \frac{acc}{2^\lambda}\right) &= \frac{1}{Q_H} \cdot \left(acc^2 - acc \cdot \nu\right) \\ &\geq \frac{1}{Q_H} \cdot \left(acc^2 - 2acc \cdot \nu + \nu^2\right) = \frac{1}{Q_H} \cdot (acc - \nu)^2 .\end{aligned}$$

Now, acc is exactly the probability ε decreased by the probability that $\mathcal{A}'$ has to abort in the first execution of $\mathcal{A}$ (because of a collision on H when simulating $\mathcal{S}_2$) or because $\mathcal{A}$ managed to forge a proof without asking the relevant hash query $H(\mathsf{pk}_0, \mathsf{pk}_1, C_0^\star, C_1^\star, A_0^\star, A_1^\star)$. Since H is a random oracle, we then have $acc = \varepsilon - (Q_H + 1)/2^\lambda$, which satisfies the condition $acc \geq (Q_H + 1)/2^\lambda$ if ε is non-negligible. □

B.2 Proof of Lemma 5.3

Proof. In the random oracle model, assuming that F_5 occurs with non-negligible probability, we build a PPT algorithm $\mathcal{B}_2$ that has non-negligible advantage in factoring an RSA modulus $N = pq$, where p and q are of the form $p = 2^k p' + 1$ and $q = 2q' + 1$ for odd integers p', q' such that $p = 5 \bmod 12$, $q = 7 \bmod 12$ and $\gcd(p', q') = 1$. We construct $\mathcal{B}_2$ using the knowledge extractor of the simulation-extractable NIZK proof of plaintext equality.

Algorithm $\mathcal{B}_2$ generates a public key PK by first choosing a random $b \xleftarrow{R} \{0, 1\}$ and setting $N_b = N$. Then, it generates $N_{1-b} = p_{1-b}q_{1-b}$ by choosing p_{1-b} and q_{1-b} as in the real key generation algorithm. It generates $g_b \in \mathbb{J}_{N_b}$ by choosing $a \xleftarrow{R} \mathbb{Z}_N^*$ and setting $g_b = 3a^2 \bmod N$. Then, it sample a random generator g_{1-b} of $\mathbb{J}_{N_{1-b}}$ and feeds $\mathcal{A}$ with a public key PK containing $\mathsf{pk}_0 := (N_0, k, \tau, g_0)$ and $\mathsf{pk}_1 := (N_1, k, \tau, g_1)$. At the outset of the experiment, $\mathcal{B}_2$ chooses a random

$i^\diamond \xleftarrow{R} [1, Q]$ as a guess for the first occurrence of event F_5 in a decryption query. Then, $\mathcal{B}_2$ interacts with $\mathcal{A}$ in the same way as the challenger of Game_5 does.

During its interaction with $\mathcal{A}$, $\mathcal{B}_0$ simulates the random oracle in the standard way using lazy sampling. It answers all decryption queries using $\mathsf{sk}_{1-b} = (p_{1-b}, q_{1-b}, p'_{1-b}, q'_{1-b})$. If the challenge phase occurs before the $i^\diamond$-th decryption query, $\mathcal{B}_2$ creates a challenge ciphertext by first choosing $s_0 \xleftarrow{R} \mathbb{J}_{N_0}$, $s_1 \xleftarrow{R} \mathbb{J}_{N_1}$ and computing

$$C_b^\star = g_b^{m_{1-\beta}} \cdot s_b^{2^{k+\tau}} \bmod N_b, \qquad C_{1-b}^\star = g_{1-b}^{m_\beta} \cdot s_{1-b}^{2^{k+\tau}} \bmod N_{1-b}$$

Then, $\mathcal{B}_2$ computes $\boldsymbol{\pi}_{\mathsf{NY}}^\star$ as a simulated proof for the statement $(\mathsf{pk}_0, \mathsf{pk}_1, C_0^\star, C_1^\star)$, which is used to form the challenge ciphertext $C^\star = (C_0^\star, C_1^\star, \boldsymbol{\pi}_{\mathsf{NY}}^\star)$ that is given to $\mathcal{A}$. We note that $C^\star$ is distributed as in Game_5.

At the $i^\diamond$-th decryption query $(C_0^\diamond, C_1^\diamond, \boldsymbol{\pi}_{\mathsf{NY}}^\diamond)$, $\mathcal{B}_2$ halts and interprets $x^\diamond = (\mathsf{pk}_0, \mathsf{pk}_1, C_0^\diamond, C_1^\diamond)$ and $\boldsymbol{\pi}_{\mathsf{NY}}^\diamond$ as the output of a malicious prover in the simulation-extractability experiment. By Lemma 4.3, there exists a PPT extractor $\mathcal{E}$ that outputs a witness with probability $\frac{1}{Q_H} \cdot \left(\varepsilon - (Q_H+1)/2^{\lambda-1}\right)^2$ if ε denotes $\mathcal{A}$'s probability to send a valid decryption query $(C_0^\diamond, C_1^\diamond, \boldsymbol{\pi}_{\mathsf{NY}}^\diamond)$. The extracted witness is either a tuple $(m^\diamond, w_0^\diamond, w_1^\diamond) \in \mathbb{Z}_{2^k} \times \mathbb{Z}_{N_0}^* \times \mathbb{Z}_{N_1}^*$ such that $C_0^\diamond = g_0^{m^\diamond} \cdot {w_0^\diamond}^{2^k} \bmod N_0$ and $C_1^\diamond = g_1^{m^\diamond} \cdot {w_1^\diamond}^{2^k} \bmod N_1$ or a prime factor of either N_0 or N_1.

If the $i^\diamond$-th query $(C_0^\diamond, C_1^\diamond, \boldsymbol{\pi}_{\mathsf{NY}}^\diamond)$ involves ciphertexts $C_0^\diamond$ and $C_1^\diamond$ that encrypt distinct messages, the witness obtained by the extractor $\mathcal{E}$ can only be a non-trivial factor of N_0 or N_1. Since the uniform choice of $i^\diamond \in_R [Q]$ is independent of $\mathcal{A}$' view, the probability that $i^\diamond$ is indeed the index of the first occurrence of F_5 is $1/Q$. In this case, since all earlier queries were made on ciphertexts $(C_0, C_1, \boldsymbol{\pi}_{\mathsf{NY}})$ such that C_0 and C_1 decrypt to the same plaintext, the secret key bit $b \in \{0,1\}$ has remained independent of $\mathcal{A}$'s view until the $i^\diamond$-th decryption query (and thus during the entire interaction of $\mathcal{B}_2$ with $\mathcal{A}$). Hence, with probability $1/2$, the extracted factor is a factor of $N_b = N$. The overall probability that this happens is then $\frac{1}{2Q \cdot Q_H} \cdot \left(\varepsilon - (Q_H+1)/2^{\lambda-1}\right)^2$ if $\varepsilon = \Pr[F_5]$.

By running the extractor of Lemma 4.3, $\mathcal{B}_2$ thus obtains a factor of N with probability $\frac{1}{2Q \cdot Q_H} \cdot \left(\Pr[F_5] - (Q_H+1)/2^{\lambda-1}\right)^2$, which yields the stated upper bound for $\Pr[F_5]$. □

C Straight-line Extractability via Fischlin's Transform

In some applications, it may be useful to have NIZK proofs where the knowledge extractor does not have to rewind. Fischlin's transform [13] provides such a straight-line knowledge extractor at the expense of somewhat longer proofs.

The transform uses a random oracle $H : \{0,1\}^* \to \{0,1\}^\lambda$ and proceeds with $\rho = O(\lambda / \log \lambda)$ parallel repetitions of a Σ-protocol with a polynomial-size challenge space $\mathcal{C}$. In short, the prover computes the first Σ-protocol messages $(A_1, \ldots, A_\rho)$ of each parallel repetition and obtains a common hash value $h = H(x, (A_1, \ldots, A_\rho))$, where x is the statement (comprised of the ciphertext

and the public key here). For each repetition $i \in [\rho]$, the prover then computes responses (z_{ij}, y_{ij}) for multiple challenge candidates $j \in \mathcal{C}$ until it finds a Σ-protocol transcript $(A_i, j, (z_{ij}, y_{ij}))$ for which the first b bits of $H(h, i, j, (z_{ij}, y_{ij}))$ are zeroes, for some parameter $b = O(\log \lambda)$. The NIZK proof then consists of a tuple $\boldsymbol{\pi} = \big((A_i)_{i\in[\rho]}, (j_i)_{i\in[\rho]}, (z_{ij_i}, y_{ij_i})_{i\in[\rho]}\big)$ where, for each index $i \in [\rho]$, j_i denote the first $j_i \in \mathcal{C}$ such that $H(h, i, j_i, (z_{ij_i}, y_{ij_i})) = \mathbf{0}^b \mid h'_i$ for some $h'_i \in \{0,1\}^{\lambda-b}$. The straight-line knowledge extractor proceeds by inspecting the list of random oracle queries and relies on the observation that, with overwhelming probability, there is at least one repetition $i^\star \in [\rho]$ for which the prover has made at least two random oracle queries $H(h, i^\star, j_{i^\star}, (z_{i^\star j_{i^\star}}, y_{i^\star j_{i^\star}}))$, $H(h, i^\star, j'_{i^\star}, (z_{i^\star j'_{i^\star}}, y_{i^\star j'_{i^\star}}))$ for distinct $j_{i^\star}, j'_{i^\star} \in \mathcal{C}$ before obtaining an output that begins with the all-zeroes string $\mathbf{0}^b$.

In order to apply this transform and prove plaintext knowledge while enabling straight-line extraction, we can use a smaller parameter τ in the encryption scheme. Indeed, the proof of Lemma 3.1 does not require τ to be larger than the security parameter but only requires $\tau \geq \log |\mathcal{C}| - 1$ to make sure that $\tau - d \geq 0$ in (11). The analysis of Chen and Lindell [8, Section 4] shows that $\tau = b + 6$ always suffices in practice. Hence, $\tau = 12$ is enough when $b \leq 6$.

Each parallel repetition requires the prover to send $2\log N + 2\tau + k$ bits if we use our Σ-protocol of Section 3.1. For the same challenge space $\mathcal{C}$ of size $\approx 2^\tau$, the Σ-protocol of Xue *et al.* [24] (which is recalled in Appendix A.1) requires $2\log N + 3\tau + 2\lambda + k$ bits of communication at each repetition. We thus save $2\lambda + \tau$ bits (which is at least 260 for $\lambda = 128$ since $\tau = 4$ never suffices as shown in [8, Section 4]) per repetition. Our version of the JL scheme and the protocol of Section 3.1 thus offer significant savings as the number of parallel repetitions should be $\rho \geq 26$ if $b = 5$ due to the constraint $\rho \cdot b \geq \lambda$.

Moreover, as explained in Appendix A.2, we obtain a faster prover. Indeed, at each iteration, the first Σ-protocol message is of the form $A = g^u \cdot v^{2^{k+\tau}} \bmod N$, where $k + \tau < \log(N)/4$, while [24] computes $A = g^{2^k \cdot u} \cdot h^{2^k \cdot v} \bmod N$ for an exponent v chosen uniformly in $[0, 2^{\lambda+\tau}N]$. For each iteration, this amounts to replacing a modular exponentiation $(h^{2^k})^v \bmod N$ with $(\lambda + \tau + \log N)$-bit exponents v by τ modular squarings allowing to compute $(v^{2^k})^{2^\tau} \bmod N$.