Updatable Public Key Encryption from DCR: Efficient Constructions With Stronger Security

Calvin Abou Haidar
ENS de Lyon, Laboratoire LIP
(U. Lyon, CNRS, ENSL, Inria, UCBL)
INRIA
France

Benoît Libert
ENS de Lyon, Laboratoire LIP
(U. Lyon, CNRS, ENSL, Inria, UCBL)
CNRS, Laboratoire LIP
France

Alain Passélègue
ENS de Lyon, Laboratoire LIP
(U. Lyon, CNRS, ENSL, Inria, UCBL)
INRIA
France

ABSTRACT
Forward-secure encryption (FS-PKE) is a key-evolving public-key paradigm that preserves the confidentiality of past encryptions in case of key exposure. Updatable public-key encryption (UPKE) is a natural relaxation of FS-PKE, introduced by Jost et al. (Eurocrypt’19), which is motivated by applications to secure messaging. In UPKE, key updates can be triggered by any sender – via special update ciphertexts – willing to enforce the forward secrecy of its encrypted messages. So far, the only truly efficient UPKE candidates (which rely on the random oracle idealization) only provide rather weak security guarantees against passive adversaries as they are malleable. Also, they offer no protection against malicious senders willing to hinder the decryption capability of honest users. A recent work of Dodis et al. (TCC’21) described UPKE systems in the standard model that also hedge against maliciously generated update messages in the chosen-ciphertext setting (where adversaries are equipped with a decryption oracle). While important feasibility results, their constructions lag behind random-oracle candidates in terms of efficiency. In this paper, we first provide a drastically more efficient UPKE realization in the standard model using Paillier’s Composite Residuosity (DCR) assumption. In the random oracle model, we then extend our initial scheme so as to achieve chosen-ciphertext security, even in a model that accounts for maliciously generated update ciphertexts. Under the DCR and Strong RSA assumptions, we thus obtain the first practical UPKE systems that satisfy the strongest security notions put forth by Dodis et al.

CCS CONCEPTS
• Security and privacy → Public key encryption.

KEYWORDS
DCR, Paillier, Updatable Public Key Encryption, Forward Security

1 INTRODUCTION
Secure communication aims to guarantee the confidentiality of communication over an insecure channel. In some cases involving long-lasting communication between multiple parties, e.g. in the context of secure group messaging applications, a desirable feature is to guarantee the confidentiality of past communications after a user gets compromised. This can be achieved by having users refresh the communication keys regularly: if an attacker compromises a user, messages sent before the last refresh remain confidential. Rather than resampling fresh keys, one would like an efficient procedure to update user keys while preserving functionality and still guaranteeing confidentiality of pre-update messages.

By “efficient”, we mean that the complexity in all metrics (in particular, key sizes) is at most poly-logarithmic in the number of time periods.
To circumvent the aforementioned barriers while still guaranteeing forward-security, an intermediate notion, termed updatable PKE (UPKE), was recently proposed by Jost et al. [34] and Alwen et al. [1]. This notion was notably motivated by secure group messaging applications. The main difference between UPKE and FS-PKE is that, in the former, new updated keys can be derived at any point by any sender. Specifically, any sender can update the public/secret keys \((pk, sk)\) of a target recipient by computing a public update consisting of a pair \((pk', ct')\), where \(pk'\) is the updated public key of the target recipient and \(ct'\) encrypts under \(pk\) the update to perform on \(sk\) in order to obtain \(sk'\).

Typically, the secret key lives in an additive group \((G, +)\) and the corresponding public key is derived from \(sk\) by a public (one-way) group homomorphism \(f: (G, +) \rightarrow (\mathbb{Z}/\ell\mathbb{Z}, \ast)\) as \(pk = f(sk)\). Updating the key can then be performed by sampling a random element \(r\) in \(G\) and revealing \(pk' = pk \ast f(r)\) and an encryption \(Enc(pk, r)\). Forward security is then guaranteed by the fact that \(f\) is hard to invert: given a compromized (fresely updated) secret key \(sk + r\) and \(f(r)\), it is hard to recover \(sk\) (as long as \(r\) was sampled from a distribution on which \(f\) is one-way). Notably, any honest user can produce a safe update that will guarantee the confidentiality of all past messages. For instance, it is possible to update the key of a recipient immediately after having sent a message to it, so that the message remains confidential in case the recipient’s secret key gets compromised in the future. In this paper, we provide new practical realizations of UPKE schemes that rely on standard assumptions and satisfy the strongest security notions in the literature.

Prior Works. A UPKE scheme is a standard PKE scheme (KGen, Enc, Dec) enhanced with two additional algorithms (UpdatePk, UpdateSk). The UpdatePk algorithm can be run by any sender on the current public key \(pk_{i-1}\) to produce a new public key \(pk_i\) as well as an update ciphertext \(up_i\). This update ciphertext \(up_i\) can then be used by the recipient, together with the current secret key \(sk_{i-1}\), to run the UpdateSk algorithm and obtain the new secret key \(sk_i\) corresponding to \(pk_i\). The main security notion that has been considered so far [1, 21, 34] is indistinguishability under chosen-plaintext attacks (IND-CPA). Security notions slightly differ in prior works and we adopt the more general and recent formalization of Dodis et al. [21] for the notions of indistinguishability under chosen-randomness chosen-plaintext attacks (IND-CR-CPA). The latter guarantees that, even if an attacker has some control on the sequence of updates by choosing (potentially bad) randomness for these updates, a single honest and non-compromised update (that is, an honest update for fresh, high-entropy randomness) suffices to guarantee that exposure of the secret key does not break the IND-CPA security of ciphertexts encrypted before the last non-compromised update.

Three constructions of IND-CR-CPA secure UPKE are known so far. A first one, based on the CDH assumption in the random oracle model, was proposed in [1, 34]. It simply consists of a hashed Elgamal encryption and is extremely efficient, but it crucially relies on the random oracle to circumvent a circular-security issue [6]. In the standard model, we are only aware of two constructions due to Dodis et al. [21]. They achieve IND-CR-CPA security in the standard model under the DDH assumption and under the LWE assumption [43] (with super-polynomial modulus-to-noise rate).

The two constructions follow a similar approach: to achieve IND-CR-CPA secure under the DDH (resp. LWE) assumption, they start by constructing a circular-secure, leakage-resilient PKE scheme obtained by tweaking the BHHO [9] (resp. Dual-Regev [27, 43]) cryptosystem. These PKE schemes use a binary secret key \(s \in \{0, 1\}^\ell\), and can retain IND-CR-CPA security even if the adversary is provided with \(s + r \in \{0, 1, 2\}^\ell\) for a uniformly random \(r \in \{0, 1\}^\ell\), as well as a (bit-by-bit) encryption of \(r\). This property is dubbed CS+LR security [21], where CS (resp. LR) stands for Circular Security (resp. Leakage-Resilience). These schemes further benefit from a homomorphic key structure as defined above: in the BHHO scheme [9], we have \(pk = \prod_{i \in [\ell]} g_i^{s_i}\) for some public group elements \(g_i\) while, in the Dual-Regev scheme [27], \(pk = A \cdot s\) for a public matrix \(A \in \mathbb{Z}_q^{n \times \ell}\).

These two schemes were shown [21] amenable to constructing UPKE schemes, where updating simply consists in sampling a random \(r \in \{0, 1\}^\ell\), encrypting it (bit-by-bit) and using the homomorphic key structure to update \(pk\). IND-CR-CPA security is then achievable from CS+LR security by leveraging this homomorphic key structure. If the adversary does not make a single update, IND-CR-CPA and CS+LR security are actually equivalent: \(r\) plays the role of the non-compromised, honest update, while \(s + r\) is the resulting secret key. Potential prior updates made by the adversary can be dealt with in this reduction using the homomorphic properties of the schemes.

For the time being, IND-CR-CPA security is guaranteed by a simple and efficient construction under CDH in the ROM and two constructions under DDH and LWE in the standard model. However, the standard model candidates are rather inefficient due to the bit-by-bit encryption procedure of update messages. In the LWE case, the scheme of [21] further relies on noise flooding, leading to a super-polynomial modulus-to-noise rate which eventually leads to very large keys/ciphertexts.

IND-CR-CPA security is a fairly weak notion and presents two noticeable weaknesses. The first one is well-known: it lacks security against adversaries equipped with a decryption oracle. To circumvent this issue, Dodis et al. [21] defined IND-CR-CCA security, in which the attacker is allowed to make decryption queries. The second weakness is that nothing prevents a malicious sender from generating updates in which the new public key and the update ciphertext are unrelated. By doing so, a user could maliciously turn a public key into a new public key for which the recipient does no longer have a secret key, thus preventing the targeted user from decrypting until it publicizes a fresh public key. To prevent this, it is desirable to preserve security even if the adversary can submit malicious updates (and not just malicious randomness for the update as in the chosen-randomness setting). This notion was termed indistinguishability under chosen-update, chosen-ciphertext attacks (IND-CU-CCA) in [21]. Generic transformations from IND-CR-CPA to IND-CR-CCA security and from IND-CR-CCA to IND-CU-CCA security have been proposed in

\[2\] In [21], the authors actually reveal a bit-by-bit encryption of \(sk\) rather than \(r\). We use this variant here and in the paper as it makes the argument for proving IND-CR-CPA easier.

\[3\] In [21], since CS+LR security reveals an encryption of \(s\) rather than \(r\), generating the update ciphertext requires to transform the encryption of \(s\) into an encryption of \(r\) by exploiting the additional homomorphic properties of the scheme.
We show that this PKE construction can be turned into an efficient under which is efficient as we work over \( \mathbb{Z} = \mathbb{Z}_{pq} \) (unknown) order. The starting point of our constructions is the Elgamal-Paillier PKE scheme additionally gets from \([\ell]\) as well as an encryption \(\text{Enc}(pk, r)\). To prove CS+LR security, we first use the algebraic structure and knowledge of \(sk + r\) to transform the secret key underlying the distribution given to the adversary from \(sk \rightarrow -r\). This is done by computing a new public key \(g^{-r} = pk \cdot g^{-sk - r}\) and by multiplying the \(c_1\) component of ciphertexts \((c_0, c_1) = (g^t, pk^t \cdot T^m)\) by \((g^{-t})^{-sk - r}\). By doing so, we remove all information about \(sk\) except \(sk + r\), and now use \(-r\) as secret key. Assuming that \(sk\) is (exponentially) larger than \(r\), a flooding argument shows that \(sk + r\) over \(\mathbb{Z}\) does not reveal any information about \(r\). We are left with showing a standard form of circular-security [6], which is doable [35, 37] from the DCR assumption. This yields a DCR-based CS+LR PKE system based on the Elgamal-Paillier PKE, with the difference that \(sk\) is now sampled from \(\mathcal{U}(0, 2^k(N - 1)/4)\) in order to apply the flooding argument. Using the CS+LR to IND-CR-CPA argument described above, we then obtain the first DCR-based IND-CR-CPA PKE scheme in the standard model. Our construction significantly outperforms prior constructions in the standard model (as evidenced by efficiency comparisons in Section 5.1) by encrypting the whole update information in one shot, without relying on bit-by-bit encryption.

From IND-CR-CPA to IND-CR-CCA Security. As a second step, we aim to upgrade our construction so as to achieve IND-CR-CCA security. To this end, we exploit the algebraic structure of ciphertexts that makes it compatible with efficient zero-knowledge proofs (a property which is lacking in the efficient hashed Elgamal constructions \((g^t, M@H(h^{\ell}))\) of [1, 34], where the hash function hinders the use of zero-knowledge proofs). We apply the standard Naor-Yung double-encryption + NIZK-paradigm [39] to our construction and show that it leads to IND-CR-CCA security. In more details, we modify our construction as follows: we add a second random generator \(h_d\) of the subgroup of order \(pq\) in the public parameters. \(Gen\) remains unchanged, but encrypting a message \(m\) to a recipient with public key \(pk\) now leads to a ciphertext \((ct, ct_d, \pi)\), where \(ct\) encrypts \(m\) under \(pk\); \(ct_d\) encrypts \(m\) under \(h_d\); and \(\pi\) is a NIZK proof that plaintexts underlying \(ct, ct_d\) are equal. \(UpdatePk\), \(UpdateSk\) algorithms are unchanged. Note, in particular, that the second key \(h_d\) is part of the public parameters and is never updated by the senders. In the proof, the second public key \(h_d\) can serve as a backdoor to answer decryption queries and simulation-soundness [44] guarantees that the adversary cannot generate malicious decryption queries, even if the challenge ciphertext contains a simulated proof at some point. It is then easy using standard Naor-Yung gymnastics to argue about IND-CR-CCA security, assuming the IND-CR-CPA security of our first construction. To obtain an efficient simulation-sound NIZK proof for plaintext equality, we first build a standard \(\Sigma\)-protocol for plaintext equality and prove it statistically sound and statistically special honest-verifier zero-knowledge. We also show that it satisfies a notion of quasi-unique responses [23] assuming that factoring is hard (which is implied by the DCR assumption). The Fiat-Shamir heuristic [24] can then compile this \(\Sigma\)-protocol into a simulation-sound NIZK in the random oracle model, by applying a result from [23]. Our \(\Sigma\)-protocol is directly inspired by Schnorr-like protocols in groups of unknown order [14, 26]. As a result, we obtain an efficient IND-CR-CCA PKE scheme based

---

\footnote{We could reverse the argument by picking exponentially-large updates \(r\) and relying on the same CS+LR security notion as Dodis et al. [21], but as secret keys are sampled once while updates could be sampled every time a message is sent, we opt for large secret keys and small updates. Taking large updates would also prevent us from using \(\zeta = 1\) as we need the updates to fit into the plaintext space \(N^k\).}
on the DCR assumption in the random oracle model. In terms of efficiency, this IND-CR-CCA construction only incurs a fairly small overhead w.r.t. its IND-CR-CPA counterpart.

From IND-CR-CCA to IND-CU-CCA Security. We finally enhance the latter IND-CR-CCA construction to achieve IND-CU-CCA security. The main change is to add a NIZK argument in the update message to prove its well-formedness: each update comes with an argument showing that the update randomness $r$ being encrypted is the randomness that allows moving from $pk_{i-1}$ to $pk_i = pk_{i-1} \cdot g^r$. In order to prove CU-CCA security, we also need to apply the Naor-Yung paradigm in update messages. Eventually, an update with several crucial observations: (i) The challenger only generates $ct$ with $ct'$ where $(ct, ct', \pi_N, \pi_{WF})$ form a standard Naor-Yung-like double encryption and $\pi_{WF}$ is a proof that $pk_{i-1,1}, pk_i, ct$ are consistent. We instantiate the Naor-Yung part exactly as in $\text{Enc}$, except that we use a second public parameters $h'_d$ as a public key for $ct'_d'$. The reason is that we prove IND-CU-CCA security via a reduction from the IND-CR-CCA security of our second scheme. The security proof follows standard Naor-Yung techniques again, with several crucial observations: (i) The challenger only generates an update at the very end of the security game, and therefore even if the NIZK argument of well-formedness in this update is simulated, the adversary cannot take advantage of this to generate proofs for ill-formed updates. In particular, we only need the NIZK argument of update well-formedness to achieve computational soundness, rather than simulation-soundness. (ii) Using a third public key $h'_d$ for the Naor-Yung encryption of updates prevents the adversary from sending the challenge ciphertext as part of an update query. It also serves as a trapdoor to decrypt the randomness used in updates in order to reduce IND-CU-CCA security to IND-CR-CCA, where an adversary can only produce updates based on (possibly malicious) randomness. The rest of the proof follows standard techniques.

To construct the NIZK argument of update well-formedness, we build a $\Sigma$-protocol and apply the Fiat-Shamir transform to make it non-interactive. The underlying $\Sigma$-protocol bears similarities with Schnorr-like protocols for groups of unknown order [26] based on the Strong RSA assumption. It is a variant of a construction due to Camenisch and Shoup [14] to prove that a Paillier encryption coincides with a discrete logarithm in a prime order group. The main difference is that, in our setting, the encrypted discrete logarithm lives in a cyclic group of order $pq$ (instead of a group of prime order $p < 2^{\ell}pq$ in [14, Section 5.2]). By enlarging the message space $\mathbb{Z}_{pq}$ with $\ell \geq 2$ in the fashion Damgård and Jurik [19], we can adapt the proof of soundness of [14].

Implementation. To complement our work, we provide an implementation of our first two constructions. The key/ciphertext/update size of our schemes makes the computation tractable, especially compared to the DDH-based standard-model construction from [21]. This implementation is heavily optimizable. The running times are a factor away of what one would expect in real world applications, but as UPKEs are meant to achieve forward security in the world of secure messaging, we believed that it was interesting to provide a very simple implementation as a proof of concept.

2 BACKGROUND AND DEFINITIONS

In this preliminary section, we recall formal definitions and security models for UPKE in Section 2.1 and the hardness assumptions on which we rely in Section 2.2. We also remind some useful lemmas in Section 2.3. Additional preliminaries on $\Sigma$-protocols, non-interactive zero-knowledge proofs, and the Fiat-Shamir transform are provided in Appendix A for completeness.

2.1 Syntax and Definitions of Updatable Public Key Encryption

We follow the syntax of [21], which is recalled hereunder.

**Definition 1.** An updatable public-key encryption (UPKE) scheme is a tuple $(GGen, KGen, Enc, Dec, UpdatePk, UpdateSk)$ of efficient algorithm with the following syntax:

- **Public parameters:** On input of a security parameters $\lambda$, $GGen$ generates a set of common public parameters $pp$.
- **Key generation:** $KGen$ takes in public parameters $pp$. It outputs a fresh secret key $sk_0$ and an initial public key $pk_0$.
- **Encryption:** $Enc$ takes in a public key $pk$ and a message $m$. It outputs a ciphertext $ct$.
- **Decryption:** $Dec$ inputs a secret key $sk$ and a ciphertext $ct$ to output a plaintext $m$.
- **Update Public Key:** Given a public key $pk$, $UpdatePk$ produces an update ciphertext $ct$ and a new public key $pk'$.
- **Update Secret Key:** $UpdateSk$ receives an update ciphertext $ct$ and secret key $sk$. It outputs a new secret key $sk'$.

**Correctness:** Let a key pair $(pk_0, sk_0) \leftarrow GGen(\lambda)$ and let $i \in \mathbb{N}$. Define $(pk_i, up_i) \leftarrow UpdatePk(pk_{i-1})$ and $sk_i \leftarrow UpdateSk(sk_{i-1}, up_{i-1})$ for $j \leq \ell$. An UPKE scheme provides correctness if, for any $m$ in the message space $M$ and all $i \leq \ell$: $Pr[Dec(sk_i, Enc(pk_i, m)) = m] = 1$.

We recall the notion of IND-CR-CPA security, as formalized by Dodis et al. [21], which strengthens the original definition of [34].

**Definition 2 ([[21], IND-CR-CPA security].** Let $UPKE = (GGen, KGen, Enc, Dec, UpdatePk, UpdateSk)$ be an UPKE scheme.

We first define the update oracle $O_{upp}(\cdot)$:

$O_{upp}(\cdot)$: Given a randomness $r_i$, the challenger increments the epoch to $i+1$. It then performs the following actions:

$(up_{i+1}, pk_{i+1}) \leftarrow \text{UpdatePk}(pk_i; r_i); sk_{i+1} \leftarrow \text{UpdateSk}(sk_i, up_{i+1})$.

For any PPT adversary $A$, we consider the IND-CR-CPA game:

1. Run $GGen(\lambda)$ to get the public parameters $pp$, then sample $(sk_0, pk_0) \leftarrow KGen(\lambda, pp), b \leftarrow \{0, 1\}$
2. $(m^*_0, m^*_1, state) \leftarrow A^{O_{upp}(\cdot)}(pk_0)$
3. Compute $c^* \leftarrow Enc(pk_f, m^*_b)$ where $\ell$ is the current epoch.
4. $state \leftarrow A^{O_{upp}(\cdot)}(c^*, state)$
5. Choose uniformly random $r^*$ and then compute

$(up^*, pk^*) \leftarrow \text{UpdatePk}(pk_f, r^*)$
$sk^* \leftarrow \text{UpdateSk}(sk_f, up^*)$

where $\ell$ is the current epoch.
6. $b' \leftarrow A(pk^*, sk^*, up^*, state)$.
7. $A$ wins the game if $b = b'$.

The advantage of $A$ in winning the above game is denoted by $\text{Adv}_{UPKE}^{\lambda_{cr-cpa}}(A) = \Pr[b = b'] - \frac{1}{2}$. 

Calvin Abou Haidar, Benoît Libert, and Alain Passelègue
An UPKE scheme is IND-CR-CPA-secure if for all PPT attackers $\mathcal{A}$, its advantage $\text{Adv}_{\text{CR-PKA}}(\mathcal{A})$ is negligible.

Definition 2 only considers passive adversaries. In the chosen-ciphertext scenario, the following definition considers stronger adversaries that are granted access to a decryption oracle.

**Definition 3 ([21], IND-CPA-CCA security).** Let UPKE = (KGen, Enc, Dec, UpdatePk, UpdateSk) be an UPKE scheme.

The update oracle is specified as in Definition 2 while the decryption oracle $O_{\text{dec}}(\cdot)$ is defined as follows:

$O_{\text{dec}}(\cdot) : \text{Given a ciphertext } c, \text{ compute and return } m = \text{Dec}(sk_i, c) \text{ (which may be } \perp \text{ if } c \text{ is an invalid ciphertext)}, \text{ where } sk_i \text{ is the secret key of the current epoch.}$

For any PPT adversary $\mathcal{A}$, the IND-CPA-CCA game goes as follows:

1. Sample $(sk_o, pk_o) \leftarrow \text{KGen}(1^\lambda)$, $b \leftarrow \mathcal{U}([0, 1])$.
2. $(m_o, m_1, \text{state}) \leftarrow \mathcal{A}^{O_{\text{upd}}(\cdot), O_{\text{dec}}(\cdot)}(pk_o)$
3. Compute $c^* \leftarrow \text{Enc}(pk_f, c^*)$ where $\ell$ is the current epoch.
4. state $\leftarrow \mathcal{A}^{O_{\text{upd}}(\cdot), O_{\text{dec}}(\cdot)}(c^*, \text{state})$.
5. No query is allowed to $O_{\text{dec}}(\cdot)$ for the challenge $c^*$ until $\mathcal{A}$ makes at least one query to the update oracle $O_{\text{upd}}(\cdot)$.
6. $c^* \leftarrow \text{UpdatePk}(pk_f, r', \ell)$.
7. $\mathcal{A}$ wins the game if $b = c^*$. The advantage of $\mathcal{A}$ in winning the above game is denoted by $\text{Adv}_{\text{UPKE}}^\text{CR-CCA}(\mathcal{A}) = \left| \Pr [b = b^*] - \frac{1}{2} \right|$.

While Definition 3 allows running the update oracle on input of adversarially-generated randomness, it still assumes that the update algorithm is honestly executed. Following [21], we introduce an even stronger definition where update messages are not assumed to be honestly generated. Instead, we can allow the adversary to come up with update messages of its own (instead of just providing the randomness to be used) that may not be a legitimate output of the UpdatePk algorithm.

In the UPKE syntax, as suggested in [21], we introduce a new algorithm VerifyUpdate$(pk, up, pk')$. This algorithm performs a verification of the update message $(up, pk')$. It returns $1$ if the update $(up, pk')$ is consistent with the public key $pk$ (that is, it corresponds to the output of UpdatePk$(pk, r)$ for some valid random coin $r$) and otherwise.

**Definition 4 ([21], IND-CU-CCA security).** Let UPKE = (KGen, Enc, Dec, UpdatePk, UpdateSk, VerifyUpdate) be an UPKE scheme. We say that it provides IND-CU-CCA security if no PPT adversary has non-negligible advantage in the same game as in the one of Definition 3, except the update oracle is specified in the following way:

$O_{\text{upd}}(\cdot) : \text{Given an update message } (up, pk'), \text{ it runs } \tau \leftarrow \text{VerifyUpdate}(pk_i, up, pk') \text{ (which may be } \perp \text{ if } (up, pk') \text{ if an invalid update, where } pk_i \text{ is the public key of the current epoch. If } \tau \neq \perp, \text{ then it runs } sk' \leftarrow \text{UpdateSk}(sk, up), \text{ else it returns } \perp.$

2.2 Hardness Assumptions

We rely on two number theoretic assumptions that imply the hardness of factoring.

**Definition 5 ([40]).** Let an RSA modulus $N = pq$, for primes $p, q$, and let an integer $\xi \geq 1$. The Decision Composite Residuosity ($\xi$-DCR) assumption holds if, for any PPT adversary $\mathcal{A}$, we have

$\text{Adv}_{\text{DCR}}^\text{\xi,\mathcal{A}}(\lambda) = | \Pr [\mathcal{A} \left( N, r N^\xi \mod N^{\xi+1} \right) = 1 | r \leftarrow \mathcal{U}(\mathbb{Z}_N^*)] - \Pr [\mathcal{A} \left( N, z \right) = 1 | z \leftarrow \mathcal{U}(\mathbb{Z}_N^{\xi+1})] | = \text{negl}(\lambda)$.

Damgård and Jurik [19] showed that all DCR assumptions are equivalent (up to a polynomial loss) for any $\xi \in \text{poly}(\lambda)$.

When $N = pq$ is a product of safe primes $p = 2p + 1$ and $q = 2q + 1$ (i.e., where $p$ and $q$ are also primes), the DCR assumption can equivalently be defined as follows.

**Definition 6 ([31]).** For a safe-prime product $N = pq$, let $T = 1 + N$. The Decision Composite Residuosity (DCR) assumption asserts that, for any PPT adversary $\mathcal{A}$, we have

$\text{Adv}_{\text{DCR}}^\text{\xi,\mathcal{A}}(\lambda) = | \Pr [\mathcal{A} \left( N, g, g^r \mod N^{\xi+1} \right) = 1 | r \leftarrow \mathcal{U}(\mathbb{Z}_N^*)] - \Pr [\mathcal{A} \left( N, g, T \cdot g^r \mod N^{\xi+1} \right) = 1 | r \leftarrow \mathcal{U}(\mathbb{Z}_N^*)] | = \text{negl}(\lambda)$,

where $r \leftarrow \mathcal{U}(\mathbb{Z}_N)$ and $g = \mu N^\xi \mod N^{\xi+1}$, where $\mu \leftarrow \mathcal{U}(\mathbb{Z}_N^*)$.

In our IND-CU-CCA secure UPKE construction, we also rely on the Strong RSA assumption to prove the soundness of a non-interactive argument system in the random oracle model.

**Definition 7 ([3]).** Let a safe-prime product $N = pq$ where $p, q$ are primes of at least $l(\lambda)$ bits for some polynomial $l : \mathbb{N} \rightarrow \mathbb{N}$. The Strong RSA assumption states that, for any PPT algorithm $\mathcal{A}$, we have

$\Pr[y = x^e \mod N \land e > 1 | y \leftarrow \mathcal{U}(\mathbb{Z}_N^*), (x, e) \leftarrow \mathcal{A}(N, y)] = \text{negl}(\lambda)$

Looking ahead, we rely (in the proof of Lemma 6) on the Strong RSA assumption in the subgroup of $2N^{1/2}$-th residues in $\mathbb{Z}_N^{\xi+1}$.

However, an algorithm that computes a non-trivial $e$-th root of $g = \mu N^2 \in \mathbb{Z}_N^{\xi+1}$, for a random $\mu \in \mathbb{Z}_N^*$, can be used to solve the usual Strong RSA problem in the subgroup $\mathbb{QR}_N$ of quadratic residues in $\mathbb{Z}_N^*$ as long as $e$ is restricted to be co-prime to $N$. Given $v = \mu^2 \mod N$, for some $\mu \in \mathbb{Z}_N^*$, one can set $g = v N^e \mod N^{\xi+1}$. From an adversary that outputs $w \in \mathbb{Z}_N^{\xi+1}$, and $e > 1$ such that $\gcd(e, N) = 1$ and $g = w^e \mod N^{\xi+1}$, one can apply Shamir’s trick to compute $\alpha, \beta \in \mathbb{Z}$ such that $\alpha e + \beta N^\xi = 1$, which yields $v = (w^\beta \cdot v^\alpha)^e \mod N^{\xi+1}$. Then, $w' \triangleq w^\beta \cdot v^\alpha \mod N$ satisfies $v = w'^e \mod N$.

2.3 Useful Lemmas

In the next section, we rely on the following lemma, which was proven by Kitagawa et al. [35] (based on earlier ideas from [12, 37]) in the context of key-dependent message security [6].

**Definition 8.** (Interactive vector game from [35].) Let $\xi \geq 1$ be an integer and $k$ be a polynomial of $\lambda$. We define the following IV$_\xi,k$ game between a challenger and an adversary $\mathcal{A}$. 

The challenger chooses a challenge bit \( b \in \mathcal{U}(\{0, 1\}) \) and generates \((N, P, Q, T, g) \leftarrow \text{GGen}(1^\lambda, \zeta)\). The challenger generates \(a_i \leftarrow \mathcal{U}(\{0, (N-1)/4\})\) and computes \(g_i \equiv q^a_i \mod N^\zeta+1\) for every \( i \in [\kappa] \), and sends \( N, g, q, g_1, \ldots, g_\kappa \) to \( \mathcal{A} \).

\( \mathcal{A} \) can adaptively make sample queries \((a_1, \ldots, a_\kappa) \in \mathbb{Z}_N^\kappa\) to the challenger. The challenger generates \( r \leftarrow \mathcal{U}(\{0, (N-1)/4\})\) and computes \( e_i \leftarrow t^b \cdot a_i \cdot q_i^r \mod N^\zeta+1\) for every \( i \in [\kappa] \).

The challenger then returns \((e_1, \ldots, e_\kappa)\) to \( \mathcal{A} \).

\( \mathcal{A} \) outputs \( b' \in \{0, 1\} \) and wins if \( b' = b \).

We remind the following result from [37].

**Lemma 1.** (DCR Interactive Vector Lemma for \( \kappa = 1, 2 \), adapted from [37]). For \( \kappa = 1, 2 \), no polynomial time adversary can have non-negligible advantage in \( IV_{\zeta, \kappa} \) with a polynomial number of queries, under the DCR assumption.

In the next section, we also rely on the following standard smudging lemma.

**Lemma 2 (Smudging Lemma).** Let \( B_1, B_2 \in \mathbb{N} \), \( a \leftarrow \mathcal{U}([-B_1, B_1]) \), \( b \leftarrow \mathcal{U}([-B_2, B_2]) \) with \( B_2/B_1 = \text{negl}(\lambda) \). Then, the distribution of \((a + b, a)\) is statistically indistinguishable from that of \((b, a)\).

**Proof.** Let \( z \in \mathbb{N} \) with \( B_2 - B_1 < z < B_1 - B_2 \) and \( c \leftarrow \mathcal{U}([-B_1, B_1]) \). Then, using the independence of \( a \) and \( b \),

\[
\Pr[a + b = z \land b = y] = \Pr[a + y = z \land b = y] = \Pr[b = y] \cdot \Pr[a = z - y] = \Pr[b = y] \cdot \Pr[a = z] = \Pr[b = y] \cdot \Pr[a = z]
\]

The last equality being valid as \( z - y \in [-B_1, B_1] \) by assumption. Also, \( \Pr[(a + b) > B_1 - B_2] \leq \Pr[|a| > B_1 - 2B_2] = (2B_2 + 1) \cdot (2B_2 + 1) = \text{negl}(\lambda) \) by our assumptions. \( \square \)

### 3 A DCR-BASED IND-CR-CPA-SECURE UPKE

In this section, we construct a UPKE scheme that we prove to be IND-CR-CPA secure assuming the DCR assumption. We follow a similar roadmap as in [21] and start by constructing a circular-secure and leakage-resilient (CS+LR) PKE scheme, for a well-chosen leakage function, under the DCR assumption.

We then extend this construction into a UPKE scheme by appending the additional two key update algorithms, and reduce the security of the resulting scheme to the CS+LR security of the underlying PKE scheme.

#### 3.1 A DCR-Based CR+LR Secure PKE

**Definition 9 (CS+LR security, adapted from [21]).** Let \( \text{GGen}, \text{KGen}, \text{Enc}, \text{Dec} \) be a PKE scheme with message space \( M \) and integer secret keys. For \( B \in \mathbb{Z} \), we say that the scheme is \( B \)-CS+LR secure if, for any PPT adversary \( \mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2) \), we have:

\[
\Pr[pp \leftarrow \text{GGen}(1^\lambda); (pk, sk) \leftarrow \text{KGen}(pp); b \leftarrow \mathcal{U}(\{0, 1\});
\]

\[
r \leftarrow \mathcal{U}([-B, B]); (m_0, m_1, st) \leftarrow \mathcal{A}_1(pp, pk)\ :
\]

\[
\mathcal{A}_2(st, pk, \text{Enc}(pk, m_0), \text{Enc}(pk, m_1)) \leftarrow \mathcal{A}_2(st, pk, \text{Enc}(pk, m_0), \text{Enc}(pk, m_1), sk\cdot\cdot r) = b \leftarrow \mathcal{U}(\{0, 1\}) = \text{negl}(\lambda),
\]

We refer to the above quantity as the advantage of \( \mathcal{A} \) against CS+LR security, denoted by \( \text{Adv}^{\text{CS+LR}}_{\mathcal{A}} \).

**Difference with the Dodis et al. definition.** Our notion of CS+LR security defers slightly from the one defined by Dodis et al. in the sense that it is actually not an extension of circular-security. Here, the adversary is given a randomized leakage of the secret key \( sk + r \) and an encryption of \( r \), whereas the definition of [21] gives an encryption of a function of \( sk \). We could modify our construction to rely on standard CS+LR security, where an encryption of \( sk \) is revealed instead of an encryption of \( sk \). We opt to do the opposite as it allows reducing the size of update messages in our UPKE construction. Indeed, in our security proof we use smudging on the sum \( sk + r \), in which \( sk \) and \( r \) play a symmetric part, allowing us to choose which of the two should be exponentially larger than the other.

We now describe our DCR-based \( B \)-CS+LR scheme.

- \( \text{GGen}(1^\lambda, B) \): Generate two \( \lambda \)-bit safe primes \( P = 2p + 1, Q = 2q + 1 \), where \( p, q \) are also primes, and set \( N = PQ \). Let \( \zeta \geq 1 \), choose \( \mu \leftarrow \mathcal{U}(\mathbb{Z}_N^*) \), and define a generator \( g \) of \( \mathbb{Z}_N^* \) of the subgroup of order \( n = p \cdot q = pq \) by setting \( g = \mu^{N^{-1}} \mod N^{-1} \). Define \( pp = (N, g, \zeta, B) \) for the public parameters.

- \( \text{KGen}(pp) \): Sample a random \( x \leftarrow \mathcal{U}([-2^B - 2^B, 2^B]) \). Set

\[
pk = h = g^x \mod N^\zeta + 1, \quad sk = x.
\]

- \( \text{Enc}(pk, m) \): Pick \( t \leftarrow \mathcal{U}(\mathbb{Z}_{(N^\lambda - 1)/4}) \) and return

\[
ct = \left( g^t \mod N^\zeta + 1, T^m \cdot h^t \mod N^\zeta + 1 \right).
\]

- \( \text{Dec}(sk, ct) \): Parse \( ct \) as \((c_0, c_1)\) and return

\[
m = \text{DLog} \left( c_0 \cdot (c_1 - x^{-1} \mod N^\zeta) \right).
\]

When \( \zeta > 1 \), the decryption algorithm can use the Damgård-Jurik technique [19] to efficiently recover \( m \in \mathbb{Z}_{N^\zeta} \) from \((1 + N)^m \mod N^\zeta + 1 \), even when \( m \) is arbitrarily large.

**Theorem 1.** Under the DCR assumption, the above PKE construction provides \( B \)-CS+LR security for any \( B \in \mathbb{N} \) such that the distributions \( (u \mod pq | u \leftarrow \mathcal{U}([-B, B])) \) and \( \mathcal{U}(\mathbb{Z}_{pq}) \) are statistically close. In particular, it holds for \( B = (N - 1)/4 \).

**Proof.** We prove the result using a sequence of four games.

**Game0:** This is the original CS+LR game obtained by encrypting the message \( m_A \), for some \( b \leftarrow \mathcal{U}(\{0, 1\}) \). The adversary is given \( pk \) and \( sk \) (m0, m1), \( \text{Enc}(pk, m_0), \text{Enc}(pk, m_1) \) and the leakage \( L(x; r) = x + r \in \mathbb{Z} \), where

\[
pk = g^x \mod N^\zeta + 1 \text{ for } x \leftarrow \mathcal{U}([-2^B, 2^B]) \text{ and } r \leftarrow \mathcal{U}([-B, B]).
\]

**Game1:** We change the generation of \( \mathcal{A}_2 \)’s input

\[
\text{Enc}(pk, m_0), \text{Enc}(pk, m_1), x + r.
\]

**Game2:** We first compute

\[
pk', \text{Enc}(pk', m_0), \text{Enc}(pk', r), x + r,
\]

where \( r \leftarrow \mathcal{U}([-B, B]) \) and \( pk' = g^r \mod N^\zeta + 1 \). Then, we compute

\[
pk = pk' \cdot g^{x + r} \mod N^\zeta + 1 \text{ is computed from } \text{Enc}(pk', m_0) = (g^t \cdot T^{m_0} \cdot pk'^t) \text{ as }
\]

\[
\text{Enc}(pk, m_0) = (g^t \cdot T^{m_0} \cdot pk'^t \cdot (g^{x + r})^t).
\]

**Game3:** \( \text{Enc}(pk', r) \) can be computed from \( \text{Enc}(pk', r) \) in a similar way. While the distribution of (1) is statistically close to that of Game1 (in
particular, the distribution of $pk'$ is statistically uniform in the subgroup generated by $g$ thanks to the constraint on $r$, the new distribution does not contain any information about $x$, except in the leakage $x + r \in \mathbb{Z}$, since (1) can be computed from (2).

**Game2:** In this game, we replace the leakage $L(x; r) = x + r$ by a random element $u \leftarrow \mathcal{U}([-2^b B, 2^b B])$ in (2). Since $g^r \mod N^{\xi+1}$, $\text{Enc}(g^{-r}, m_b)$, $\text{Enc}(g^r, r)$ do not depend on $x$ and since

$$\Delta(\mathcal{U}([-2^b B, 2^b B]) \ast \mathcal{U}([-B, B])) \leq B/2^b = \text{negl}(\lambda)$$

by Lemma 2, it follows that Game2 and Game1 are statistically indistinguishable.

**Game3:** In this game, we replace $\text{Enc}(g^{-r}, r) = (g^r, T^r(g^{-r})^t)$ by a pair $(Tg_t, (g^{-r})^t)$ in (2). The DCR assumption guarantees that $g^t \approx_{T} Tg_t$, as $t$ is unknown to the adversary. Hence, assuming DCR, we have

$$(g^t, T^r(g^{-r})^t) \approx_c (Tg_t, (g^{-r})^t),$$

and distributions from Game3 and Game2 are computationally indistinguishable. In Game3, $\mathcal{A}_2$'s input is thus generated from a tuple

$$pk' = (g^{-r}, \text{Enc}(g^{-r}, m_b), (Tg_t, (g^{-r})^t), u),$$

where $r \leftarrow \mathcal{U}([-B, B]), u \leftarrow \mathcal{U}([-2^b B, 2^b B])$.

**Game4:** We finally replace $\text{Enc}(g^{-r}, m_b) = (g^k, \tau^{mk}. (g^{-r})^k)$ and $(Tg_t, (g^{-r})^t)$ from the distribution of (3) by elements $\text{Enc}(g^{-r}, 0) = (g^0, (g^{-r})^0)$ and $(g^t, (g^{-r})^t)$.

Using the Interactive Vector game $IV_{\xi,2}$ for $(g_1, g_2) = (g, (g^{-r}) \mod N^{\xi+1})$ with two queries $(0, m_b)$ and $(1, 0)$ (where the IV challenger replies using randomness $k, t \leftarrow \mathcal{U}(\mathbb{Z}_{N-1})$), it is straightforward to prove that these two distributions are computationally indistinguishable via Lemma 1, assuming that DCR holds. Note that this step again requires the distribution of $g^{r} \mod N^{\xi+1}$ to be statistically close to the distribution $(g^{\alpha} \mod N^{\xi+1}) \alpha \leftarrow \mathcal{U}(\mathbb{Z}_{N-1})$, as guaranteed by the constraint on $B$ in the statement of Theorem 1.

We finally observe that the distribution Game4 is independent of the bit $b \in \{0, 1\}$. Indeed, $g, g^{-r} \mod N^{\xi+1}, (g^k, (g^{-r})^k)$, and $(g^t, (g^{-r})^t)$ do not carry any information about $m_b$. This concludes the proof of Theorem 1.

### 3.2 A DCR-Based IND-CCA-Secure UPKE

To construct our UPKE, we extend the CS+LR PKE scheme described in Section 3.1 with algorithms (UpdatePk, UpdateSk), defined as follows. We set $B = (N-1)/4$ for our UPKE scheme, which satisfies the requirements of Theorem 1. In the description below, we include the current epoch number in each public key as it simplifies our security proofs in the CCA-secure extensions later on.

- **GGen($\ell$):** Generate two $\ell$-bit safe primes $P = 2p + 1, Q = 2q + 1$, where $p, q$ are also primes, and set $N = PQ$. Choose $\mu \leftarrow \mathcal{U}(\mathbb{Z}_N)$, and obtain a generator $g$ of the subgroup of order $n = \varphi(N)/4 = pq$ by setting $g = \mu^2N^\ell \mod N^{\xi+1}$. Define $pp = (N, g)$ as public parameters.

- **KGen($pp$):** Sample a random $x \leftarrow \mathcal{U}([-2^b B, 2^b B])$ and compute $h = g^x \mod N^{\xi+1}$. Let $r = 0$ the current epoch. Set $pk = (r, h), sk = (r, x)$.

- **Enc($pk, m$):** Pick $t \leftarrow \mathcal{U}(\mathbb{Z}_{N-1}/4)$ and return $ct = (g^t \mod N^{\xi+1}, (1 + N)^{h^t} \mod N^{\xi+1})$.

- **Dec($sk, ct$):** Parse $ct$ as $(c_0, c_1)$ and $sk$ as $(r, x)$. Return $m = \text{DLog}(c_1 : c_0^{-x} \mod N^{\xi+1})$.

- **UpdatePk($pk$):** Parse $pk = (r, h)$ and set $r \leftarrow r + 1$. Return $(up, (r, h'))$.

- **UpdateSk($sk$, up):** Given the secret key $sk = (r, x)$, compute $r \leftarrow \text{Dec}(sk, up) \in \mathbb{Z}_{N^r}$. Let $r = \text{min}(r, N^r - r)$ and let $b_r \in \{0, 1\}$ such that $r = (1 - b_r) \cdot r + b_r \cdot (N^r - r) \mod N^r$. Return $sk' = (r + x', 1)$, where $x' = x + (-1)^{b_r} \cdot r \in \mathbb{Z}$.

**Theorem 2.** Under the DCR assumption, the above UPKE construction provides IND-CCA-CPR security.

**Proof.** Let $\mathcal{A}$ be an adversary in the IND-CCA-CPR security game of our UPKE. We build an adversary $\mathcal{B}$ against the CS+LR security of the PKE scheme in Section 3.1 with chosen leakage $L(x; r) = x + r \in \mathbb{Z}$ for $r \leftarrow \mathcal{U}([-B, B])$ and $x \leftarrow \mathcal{U}([-2^b B, 2^b B])$.

- **Our adversary $\mathcal{B}$ receives from its challenger a public key $pk = h = g^x \mod N^{\xi+1}$ associated with a secret key $x$.**

- **$\mathcal{B}$ implicitly sets $x_0 = x$ and sets the epoch $r$ to 0 by forwarding $pk_0 = (0, h)$ to $\mathcal{A}$.**

- **When $\mathcal{A}$ queries oracle $O_{\text{upd}}(\delta)$, $\mathcal{B}$ just bookkeeps $\delta_i$.**

- **At some point, $\mathcal{B}$ receives the challenge messages $m_{\ell_0}^0, m_{\ell_0}^1$ from $\mathcal{A}$. Adversary $\mathcal{B}$ forwards $m_{\ell_0}^0, m_{\ell_0}^1$ to its challenger, using them as challenge messages. In response, $\mathcal{B}$ receives a leakage $z = L(x; r) = x + r \in \mathbb{Z}$, and ciphertexts $c = \text{Enc}(pk, m_{\ell_0}^0)$ and $c' = \text{Enc}(pk, m_{\ell_0}^1)$ in order to build a challenge ciphertext for $\mathcal{A}$. $\mathcal{B}$ needs to compute $c^* = \text{Enc}(pk, m_{\ell_1})$, where $\ell_1$ is the epoch at which $\mathcal{A}$ sent its challenge messages. We have $pk_\ell = (\ell, g^x \cdot z_{\ell_0} \delta, \delta_\ell \mod N^\xi)$. Parsing $c = (c_0, c_1)$ and defining $\delta = \sum_{i=1}^\ell \delta_i$ (computed over $\mathbb{Z}$), $\mathcal{B}$ simply computes**

$$c^* = (c_0, c_0^\delta \cdot c_1 \mod N^{\xi+1})$$

and forwards $c^*$ as challenge ciphertext to $\mathcal{A}$.

- **Then, $\mathcal{A}$ resumes its queries of $O_{\text{upd}}(\cdot)$ and $\mathcal{B}$ keeps registering them until the last query $O_{\text{upd}}(\delta_\ell)$ is made.**

- **$\mathcal{B}$ has to send the final secret key $sk' = (\ell + 1, x')$, which is the result of all update queries made by $\mathcal{A}$ and a last honestly-generated update (of which the randomness is not known to $\mathcal{A}$).** Letting $\delta' = \sum_{i=1}^\ell \delta_i \in \mathbb{Z}$, the reduction $\mathcal{B}$ uses the leakage randomness $r$ as the randomness of the final update, which is hidden from $\mathcal{A}$’s view. Namely, $\mathcal{B}$ sets $x' = x + \delta' = x + r + \delta$ (over $\mathbb{Z}$) and defines the new public for the next epoch to be $pk_\ell = (\ell + 1, g^{x'}) \mod N^{\xi+1})$. To finalize the update message, $\mathcal{B}$ can simply update the public key used
in the encryption of the $c'$ from $g^{\sigma}$ to $g^{\sigma^e}$. This is done using the same technique as for the challenge ciphertext. Namely, given $c' = \text{Enc}(pk, r) = (c'_0, c'_1)$, $B$ simply sets $up^e = (c'_0, c'_1, c'_2 \mod N^e+1)$ and then sends $(pk^e, sk^e, up^e)$ to $A$.

- When $A$ finally halts with some output $b' \in \{0, 1\}$, $B$ outputs the same bit $b'$.

By inspection, it is clear that $B$ perfectly simulates $A$’s challenger in the IND-CR-CPA security game and succeeds whenever $A$ does. Theorem 2 follows. □

4 FROM CR-CPA TO CR-CCA/CU-CCA SECURITY IN THE ROM

We now enhance our UPKE scheme so that it becomes IND-CR-CCA secure. To achieve that we need to build Non Interactive Zero Knowledge (NIZK) arguments that allow to perform the Naor-Yung transformation for our IND-CPA-secure PKE. We start by constructing a $\Sigma$-protocol for proving plaintext equality in Section 4.1, which we transform into a simulation-sound NIZK arguments via the Fiat-Shamir transform and use to build an IND-CR-CCA secure UPKE scheme in Section 4.2.

We then further upgrade the scheme to achieve IND-CU-CCA security in Section 4.4, by adding NIZK arguments that updates are well-formed. The latter NIZK arguments are obtained in a similar way by building a $\Sigma$-protocol in Section 4.3 and applying the Fiat-Shamir transform.

4.1 Proofs of Plaintext Equality

In order to achieve IND-CR-CPA security in the ROM, we apply the Naor-Yung transformation [39] in the random oracle model. Let public parameters $pp = (N, g, \zeta)$, where $N = PQ$ is a safe-prime product and $g \in \mathbb{Z}_{N^e+1}^\times$ generates the subgroup of $2N^5$-th residues in $\mathbb{Z}_{N^e+1}$. Let $h = g^x \mod N^e+1$, $h_d = g^{\xi d} \mod N^e+1$ be two public keys, for some underlying secret keys $x, x_d \in \mathbb{Z}_{2N^5(N-1)/4}$.

We need to prove membership of the language

$$L_{NY} = \{(h, h_d, C_0, C_1, D_0, D_1) \in (\mathbb{Z}_{N^e+1})^6 ~|~ \exists c, t_d \in \mathbb{Z}_{pq}, m \in \mathbb{Z}_{N^e+1} : c_0 = g^{2 \cdot t_c} \mod N^e+1 \land c_1 = (1+N)^{2m} \cdot h^{2 \cdot t_c} \mod N^e+1 \land D_0 = g^{2 \cdot t_d} \mod N^e+1 \land D_1 = (1+N)^{2m} \cdot h_d^{2 \cdot t_d} \mod N^e+1\}.$$

The $\Sigma$-protocol for $L_{NY}$ is standard and goes as follows.

- $P$ picks $t_c, t_d \leftarrow [0, 2^{2 \lambda}, \frac{N-1}{4}, \frac{N-1}{4}]$, $m' \leftarrow \mathbb{Z}_{N^e}$ and sends

  $$C_0' = g^{2 \cdot t_c} \mod N^e+1, \quad C_1' = (1+N)^{2m'} \cdot h^{2 \cdot t_c} \mod N^e+1, \quad D_0' = g^{2 \cdot t_d} \mod N^e+1, \quad D_1' = (1+N)^{2m'} \cdot h_d^{2 \cdot t_d} \mod N^e+1.$$

- $V$ sends a random challenge $c \leftarrow \mathcal{U}(\{0, 2^{2 \lambda} - 1\})$.

- $P$ computes $i_c = t_c' + c \cdot t_c, i_d = t_d' + c \cdot t_d$ (over $\mathbb{Z}$) and $m = m' + c \cdot m \mod N^e$. If $i_c, i_d \in [0, 2^{2 \lambda}, \frac{N-1}{4}]$, it sends $i_c, i_d, m$ to $V$. Otherwise, it aborts.

- $V$ first verifies that $i_c, i_d \in [0, 2^{2 \lambda}, \frac{N-1}{4}]$, then that

  $$c_0' = C_0^{c_0} \cdot g^{2 \cdot i_c} \mod N^e+1, \quad c_1' = C_1^{c_1} \cdot (1+N)^{2m} \cdot h^{2 \cdot i_c} \mod N^e+1,$$

  $$D_0' = D_0^{d_0} \cdot g^{2 \cdot i_d} \mod N^e+1, \quad D_1' = D_1^{d_1} \cdot (1+N)^{2m} \cdot h_d^{2 \cdot i_d} \mod N^e+1.$$

$V$ accepts if all checks succeed, and rejects otherwise.

**Lemma 3.** The above $\Sigma$-protocol provides statistical soundness. (The proof is available in Appendix B.1.)

**Lemma 4.** The $\Sigma$-protocol $\Sigma_{NY}$ is statistically special honest-verifier zero-knowledge. (The proof is given in Appendix B.2.)

The $\Sigma$-protocol is used to prove plaintext equalities in order to apply the Naor-Yung paradigm in the random oracle model. In order to obtain the simulation-soundness property by applying Theorem 6, we need the following lemma.

**Lemma 5.** Under the assumption that factoring $N$ is hard, the above $\Sigma$-protocol has quasi-unique responses. (The proof is given in Appendix B.3.)

4.2 IND-CR-CCA secure UPKE

We now upgrade our IND-CR-CPA scheme in the following way. Since we just aim at IND-CR-CCA (rather than IND-CU-CCA) security in this section, we can set $B = (N-1)/4$ and $\zeta \geq 1$.

- $GGen(1^k, B)$:
  1. Generate two $\lambda$-bit safe primes $P = 2p+1, Q = 2q+1$, where $p, q$ are also primes, and set $N = PQ$. Choose an integer $\zeta \geq 1$.
  2. Choose $\mu, \mu_d \leftarrow \mathcal{U}(\mathbb{Z}_N)$ at random. Then, compute generators of the subgroup of order $n = \phi(N)/4 = pq$ by setting $g = \mu^{2N^5} \mod N^e+1$ and $h_d = \mu_d^{2N^5} \mod N^e+1$.
  3. Choose a hash function $H : \{0, 1\}^* \rightarrow \{0, 1\}^\lambda$ that is modeled as a random oracle in the analysis.

Define $pp = (N, \zeta, g, h, h_d, H)$ as public parameters.

- $KGen(pp)$: Sample a random $x \leftarrow \mathcal{U}([-2^B, 2^B])$. Define the key pair $(sk, pk)$ by setting $sk = (\tau, x)$ and $pk = (\tau, h)$, where $\tau = 0$ is a counter and $h = g^{\sigma} \mod N^e+1$.

- $Enc(pk, m)$: To encrypt $m \in \mathbb{Z}_{N^e}$ under the public key $pk = (\tau, h) \in \mathcal{N} \times \mathcal{N}_{N^e+1}$, conduct the following steps.

  1. Pick $t_c, t_d \leftarrow \mathcal{U}([0, 2^{2 \lambda}])$ and compute

     $$(C_0, C_1) = (g^{t_c} \mod N^e+1, (1+N)^m \cdot h^{t_c} \mod N^e+1) \quad (D_0, D_1) = (g^{t_d} \mod N^e+1, (1+N)^{m} \cdot h_d^{t_d} \mod N^e+1).$$

  2. Using $(t_c, t_d, m)$, generate a NIZK proof $\pi$ that

     $$\pi = (C_0', C_1', D_0', D_1', c, i_c, i_d, m) \in L_{NY}$$

     where $L_{NY}$ is the language defined in Section 4.1. This proof is of the form

     $$\pi = (C_0', C_1', D_0', D_1', c, i_c, i_d, m) \in \{0, 1\}^{\lambda} \times \{0, 2^{2 \lambda} \cdot (N-1)/4\}^2 \times \mathbb{Z}_{N^e},$$

     where $C_0', C_1', D_0', D_1'$ are elements of the ciphertext space $\mathbb{Z}_{N^e+1}$, and $c, i_c, i_d, m$ are elements of the challenge and auxiliary spaces.
with \( c = H(\tau, (h, h_d, C_0, C_1, D_0, D_1), (C_0', C_1', D_0', D_1')) \), and where \((C_0', C_1', D_0', D_1')\) satisfy (4).

Output the ciphertext \( ct = (C_0, C_1, D_0, D_1, \pi) \).

- **Dec(sk, ct):** Parse ct as \((C_0, C_1, D_0, D_1, \pi)\) and sk as \((\tau, x)\) ∈ \( N \times \mathbb{Z} \). Return \( \bot \) if \( \pi \) is not a verifying NIZK argument that \((h, h_d, C_0, C_1, D_0, D_1) \in L_{NY} \). Otherwise, return

\[
m = D\log \left( C_0^2 \cdot C_0^{-2x} \mod N^{\xi+1} \right) \cdot 2^{-1} \mod N^{\xi}.
\]

- **UpdatePk(pk):** To update \( pk = (\tau, h) \in N \times \mathbb{Z} \),
  1. Choose \( \tau' \leftarrow U([-B, B]) \) and compute \( h' = h \cdot g^{\tau'} \mod N^{\xi+1} \).
  2. Choose \( k \leftarrow U(\mathbb{Z}(N-1)/4) \) and compute

\[
up = (U, V) = \left( g^{\tau'} \cdot N^{\xi+1}, (1 + N)^\tau \cdot h^k \mod N^{\xi+1} \right)
\]

and set \( \tau' = \tau + 1 \).

Return \((up = (U, V), pk' = (\tau', h'))\).

- **UpdateSk(sk, up):** Given \( sk = (\tau, x) \in N \times \mathbb{Z} \) and an update message \((up = (U, V), pk' = (\tau', h'))\), return \( \bot \) if \( \tau' \neq \tau + 1 \). Otherwise, conduct the following steps.
  1. Return \( \bot \) if \((U, V) \notin (\mathbb{Z}_N^{\xi+1})^2 \) or \( h \notin \mathbb{Z}_N^{\xi+1} \).
  2. Compute

\[
r = D\log \left( V^2 \cdot U^{-2x} \mod N^{\xi+1} \right) \cdot 2^{-1} \mod N^{\xi}.
\]

  3. Let \( \bar{r} = \min(r, N^{\xi} - r) \) and let \( b_r \in \{0, 1\} \) such that

\[
\bar{r} = (1 - b_r) \cdot r + b_r \cdot (N^{\xi} - r) \mod N^{\xi}.
\]

  Compute \( x' = x + (-1)^{b_r} \cdot r \mod N^{\xi} \).
  4. Return \( sk' = (\tau + 1, x') \in N \times \mathbb{Z} \).

**Theorem 3.** The above construction provides IND-CR-CCA security assuming that:

(i) The DCR assumption holds;

(ii) The NIZK proof for \( L_{NY} \) provides simulation-soundness. (The proof is given in Appendix B.4).

### 4.3 Arguments of Well-formedness for Update Ciphertexts

To achieve IND-CU-CCA security, we rely on proofs that the updates are well-formed. We obtain these proofs by using a classical Schnorr-like proof in hidden-order groups.

Given \( g, h \) of order \( pq \) in \( \mathbb{Z}_N^{\xi+1} \), where \( \xi \geq 2 \). Let \( B = (N - 1)/4 \). We need to prove membership of the language

\[
L_{WFU} = \{ ((h, h') \in (\mathbb{Z}_N^{\xi+1})^2, up = (U, V)) \mid \exists k \in [0, B], r \in [-B, B], U = g^k \mod N^{\xi+1}
\]

\[
\land V = (1 + N)^\tau \cdot h^k \mod N^{\xi+1}
\]

\[
\land (h/h')^2 = g^r \mod N^{\xi+1}\},
\]

which is the language of well-formed updates. In the proof of soundness, however, we can only guarantee membership of

\[
\tilde{L}_{WFU} = \{ ((h, h') \in (\mathbb{Z}_N^{\xi+1})^2, up = (U, V)) \mid \exists k \in [-2^{2\lambda+1}B, 2^{2\lambda+1}B], r \in [-2^{2\lambda+1}B, 2^{2\lambda+1}B] : U^2 = g^{2k} \mod N^{\xi+1} \land (h/h')^2 = g^{2r} \mod N^{\xi+1}
\]

\[
\land V^2 = (1 + N)^{2\tau} \cdot h^{2k} \mod N^{\xi+1} \},
\]

so that we have a soundness slack as the actual witnesses \((k, r)\) live in \([0, B] \times [-B, B] \).

We assume that \( \xi \geq 2 \), so that the interval \([0, 2^{2\lambda}B]\) fits in the message space \( \mathbb{Z}_N^{\xi+1} \) of the encryption scheme. We define the following \( \Sigma \)-protocol \( \Sigma_{WFU} \) for \( L_{WFU} \): 

- \( \mathcal{P} \) picks \( r', k' \leftarrow U([-2^{2\lambda}B, 2^{2\lambda}B]) \), and sends

\[
U' = g^{2k'} \mod N^{\xi+1},
\]

\[
V' = (1 + N)^{2\tau'} \cdot g^{2k'} \mod N^{\xi+1},
\]

\[
H = g^{2r'} \mod N^{\xi+1}
\]

to the verifier.

- \( \mathcal{V} \) sends a random challenge \( c \leftarrow U([0, 2^\lambda - 1]) \).

- \( \mathcal{P} \) computes responses \( k = k' + ck \) and \( \tilde{r} = r' + cr \mod N \). If \( k, \tilde{r} \in [-2^{2\lambda}B, 2^{2\lambda}B] \), it sends \( k, \tilde{r} \) to \( \mathcal{V} \), else it aborts.

- \( \mathcal{V} \) first verifies that \( \tilde{k}, \tilde{r} \in [-2^{2\lambda}B, 2^{2\lambda}B] \), then that

\[
U' = U^{-2c} \cdot g^{2k} \mod N^{\xi+1},
\]

\[
V' = V^{-2c} \cdot (1 + N)^{2\tau} \cdot g^{2k} \mod N^{\xi+1},
\]

\[
H = (h/h')^{-2c} \cdot g^{2r} \mod N^{\xi+1}
\]

(5)

It accepts if all these checks are correct, and rejects otherwise.

The special-soundness property can be proven under the Strong RSA assumption, by adapting a technique from Camenisch and Shoup [14]. One difference is that, here, the encrypted discrete logarithm lives in \([- (N - 1)/4, (N - 1)/4]\) (instead of \( \mathbb{Z}_p \) for a public prime order \( p < 2^\lambda \)). Our use of a larger message space \( \mathbb{Z}_N^{\xi+1} \) with \( \xi \geq 2 \) allows the proof to go through by adapting techniques from [14] and [26].

**Lemma 6.** The above \( \Sigma \)-protocol provides soundness for the language \( L_{WFU} \) under the Strong RSA assumption.

**Proof.** Let us assume that a prover can come up with accepting transcripts \((U', V', H), c_1, (k_1, \tilde{r}_1), (U', V', H), c_2, (k_2, \bar{r}_2)\). We show that, unless the Strong RSA assumption is false, it must be that \((h, h'), (U, V) \in L_{WFU} \).

We define \( \Delta c = c_1 - c_2 \), \( \Delta k = \tilde{k}_1 - k_2 \), \( \Delta r = \tilde{r}_1 - \bar{r}_2 \) with \( \Delta k \in [-2^{2\lambda+1}B, 2^{2\lambda+1}B] \). From the verification equations (5), we have

\[
U^{2\Delta c} = g^{2\Delta k} \mod N^{\xi+1},
\]

\[
V^{2\Delta c} = (1 + N)^{2\Delta r} \cdot h^{2\Delta k} \mod N^{\xi+1},
\]

\[
(h/h')^{2\Delta c} = g^{2\Delta r} \mod N^{\xi+1}.
\]

Using the same arguments as [14, Theorem 4], we first show that, unless the same prover can compute a non-trivial root of \( g \in \mathbb{Z}_N^{*\xi+1\Delta c} \),
which in turn yields

\[ g^2 = \left( g^\alpha \cdot (h/h')^\beta \right)^{2\lambda \Delta /d} \bmod N^{\xi+1}, \]

which in turn yields

\[ g^2 = \left( g^\alpha \cdot (h/h')^\beta \right)^{2\lambda \Delta /d} \bmod N^{\xi+1}, \tag{7} \]

for some \( \psi \in Z_{N^{\xi+1}}^\ast \) such that \( \psi^d = 1 \bmod N^{\xi+1} \), meaning that \( \text{ord}(\psi)|d \). Since \( \psi \in Z_{N^{\xi+1}}^\ast \), we have \( \text{ord}(\psi)|2N^{\xi} \). Then, since we also have \( d|\Delta \) and \( \gcd(\Delta, N^{\xi}pq) = 1 \), this implies that \( \text{ord}(\psi) = 2 \).

Since we cannot have \( \psi = -1 \) (as \(-1\) is not a square in \( Z_{N^{\xi+1}}^\ast \),
when \( P = Q = 3 \bmod 4 \), we have either \( \psi = 1 \), or a non-trivial factor of \( N = PQ \) is revealed by computing \( \gcd(\psi + 1, N) \) (because \( (\psi+1)(\psi-1) = 0 \bmod N^{\xi+1} \)). Unless the factoring assumption (and thus the Strong RSA assumption) is broken, we thus have

\[ g^2 = \left( g^\alpha \cdot (h/h')^\beta \right)^{2\lambda \Delta /d} \bmod N^{\xi+1}. \]

Then, we distinguish three cases. If \( \Delta \Delta /d \) is even, we have

\[ g = \left( g^\alpha \cdot (h/h')^\beta \right)^{\Delta \Delta /d} \bmod N^{\xi+1}, \tag{8} \]

since squaring is a permutation in the subgroup of squares when \( P = Q = 3 \bmod 4 \), and \( p \mid g^\alpha \cdot (h/h')^\beta \bmod N^{\xi+1} \) is then a non-trivial root of \( g \). If \( \Delta \Delta /d \) is odd and

\[ g \neq \pm \left( g^\alpha \cdot (h/h')^\beta \right)^{\Delta \Delta /d} \bmod N^{\xi+1}, \]

we obtain a non-trivial factor of \( N = PQ \). Finally, if \( \Delta \Delta /d \) is odd and

\[ g = \left( g^\alpha \cdot (h/h')^\beta \right)^{\Delta \Delta /d} \bmod N^{\xi+1}, \tag{9} \]

then \( g = \left( \pm g^\alpha \cdot (h/h')^\beta \right)^{\Delta \Delta /d} \bmod N^{\xi+1} \) and we also have a root \( \mu \mid g^\alpha \cdot (h/h')^\beta \bmod N^{\xi+1} \) of \( g \).

We can show in the same way that the Strong RSA assumption is broken if \( \Delta \) does not divide \( \Delta \).

We now assume that \( \Delta \Delta /|\Delta | = \Delta \Delta /\Delta \Delta \Delta \Delta \). Since \( \Delta \Delta \Delta \Delta < \min(p, q) \), we necessarily have \( \gcd(\Delta \Delta \Delta \Delta, pq) = 1 \) and \( \gcd(\Delta \Delta \Delta \Delta, N) = 1 \). Then, if we define \( \hat{\Delta} = \Delta^{-1} \bmod N \), the above arguments imply

\[ U^2 = g^{2\hat{\Delta}k} \bmod N^{\Delta+1} \equiv g^{2\hat{\Delta}k/(\Delta \Delta /\Delta \Delta \Delta \Delta)} \bmod N^{\xi+1}, \]

\[ V^2 = \left( 1 + N \right)2\hat{\Delta}e \cdot h^{2\hat{\Delta}ke} \bmod N^{\xi+1} \]

\[ = \left( 1 + N \right)2\hat{\Delta}e \cdot h^{2\hat{\Delta}ke} \bmod N^{\xi+1}, \]

\[ (h/h')^2 = g^{2\hat{\Delta}e} \bmod N^{\xi+1} \]

\[ \equiv g^{2\hat{\Delta}e} \bmod N^{\xi+1}, \]

which means that \( ((h, h'), (U, V)) \in \mathcal{L}_{WFU} \) since we have \( \Delta \Delta /\Delta \Delta \Delta \Delta \in [-2^{\Delta \Delta \Delta \Delta}, 2^{\Delta \Delta \Delta \Delta}] \).

**Lemma 7.** The \( \Sigma \)-protocol \( \Sigma_{WFU} \) satisfies the special honest verifier zero-knowledge property for the language \( \mathcal{L}_{WFU} \).

Proof. In the special honest verifier setting, the simulator is given the challenge \( e \) and has to simulate an accepting transcript with the proper distribution for \( h^e, (U, V) \in \mathcal{L}_{WFU} \). We can uniformly sample \( k, r \leftarrow \mathcal{U}([-2^{\Delta \Delta \Delta \Delta}, 2^{\Delta \Delta \Delta \Delta}]) \) and then set

\[ U' = U^{2e} \cdot g^{2k} \bmod N^{\xi+1}, \]

\[ V' = V^{2e} \cdot (1 + N)^{2r} \cdot h^{2k} \bmod N^{\xi+1}, \]

\[ H = (h/h')^{2e} \cdot g^{2r} \bmod N^{\xi+1}. \]

We then return \( (U', V', H, c, k, r) \).

This is a valid transcript as it satisfies (5). Moreover, for any true statement \( (h', (U, V)) \in \mathcal{L}_{WFU} \), we know that \( \exists k \in [0, \beta] \) such that \( U^2 = g^{2k} \bmod N^{\xi+1}, V^2 = (1 + N)^{2r} \cdot h^{2k} \bmod N^{\xi+1}, (h/h')^2 = g^{2r} \bmod N^{\xi+1} \). Thus we have:

\[ U' = g^{2k - 2e} \bmod N^{\xi+1}, \]

\[ V' = (1 + N)^{2r} \cdot h^{2k - 2e} \bmod N^{\xi+1}, \]

\[ H = g^{2r - 2e} \bmod N^{\xi+1}. \]

Using Lemma 2, we know that the distributions of \( \tilde{k}, \tilde{r} \) in a real interaction are statistically uniform in \( [-2^{\Delta \Delta \Delta \Delta}, 2^{\Delta \Delta \Delta \Delta}] \), as \( \text{gcd}(\lambda, k) < 2\lambda \cdot B \). We can then use a somewhat larger modulus as we need \( \xi \geq 2 \). In order to ensure the soundness of our \( \Sigma \)-protocol in Section 4.3.

- **GGen(1, k):**
  1. Choose \( \lambda \)-bit safe primes \( P = 2p + 1, Q = 2q + 1 \) and set \( N = PQ \). Choose an integer \( \xi \geq 2 \).
  2. Choose \( \mu, \mu_d, \mu_d' \leftarrow \mathcal{U}(Z_{N^2}) \). Then, compute subgroups of the order \( n = \phi(N)/4 = pq \) by setting \( g = \mu^{2N^2} \bmod N^{\xi+1}, h_d = \mu_d^{2N^2} \bmod N^{\xi+1}, \) and \( h_d' = \mu_d'^{2N^2} \bmod N^{\xi+1} \).
  3. Choose hash functions \( H, H' : \{0, 1\}^* \rightarrow \{0, 1\}^\lambda \) that will be modeled as random oracles in the analysis.

Define \( pp = (N, \xi, g, h_d, h_d', H, H') \) as public parameters.

- **KGen(pp):** Sample a random \( x \sim \mathcal{U}(-2^{\Delta}, 2^\Delta) \). Define the key pair \( (sk, pk) \) by setting \( sk = (t, x) \) and \( pk = (t, h) \) where \( t = 0 \) and \( h = g^t \bmod N^{\xi+1} \).

- **Enc(pk, m):** To encrypt \( m \in Z_{N^\xi} \) under \( pk = (t, h) \in \mathbb{N} \times Z_{N^\xi}^\ast \), conduct the following steps.

  1. Pick \( t, d \leftarrow \mathcal{U}(Z_{N^\xi}/4) \) and compute
     \[
     \begin{align*}
     (C_0, C_1) &= \left( g^{t} \bmod N^{\xi+1}, (1 + N)^{m} \cdot h^{t} \bmod N^{\xi+1} \right) \\
     (D_0, D_1) &= \left( g^{d} \bmod N^{\xi+1}, (1 + N)^{m} \cdot h^{d} \bmod N^{\xi+1} \right).
     \end{align*}
     \]

Calvin Abou Haidar, Benoît Libert, and Alain Passelègue
Efficient UPKE from DCR with Strong Security

<table>
<thead>
<tr>
<th>Secret key</th>
<th>Public Key</th>
<th>Ciphertext</th>
<th>Update</th>
<th>Security</th>
<th>ROM</th>
<th>Assumption</th>
</tr>
</thead>
<tbody>
<tr>
<td>256b</td>
<td>512b</td>
<td>512b</td>
<td>512b</td>
<td>419Mb</td>
<td>Yes</td>
<td>DDH</td>
</tr>
<tr>
<td>1.3kb</td>
<td>328kb</td>
<td>328kb</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Comparison of key/ciphertext/update sizes for existing UPKE schemes with 128-bit strength security

2. Using witnesses \((t_c, t_d, m)\), generate a NIZK proof \(\pi\) that 
\((h, h_d, C_0, C_1, D_0, D_1) \in \mathcal{L}_{NY}\), where \(\mathcal{L}_{NY}\) is the language defined in Section 4.1. This proof \(\pi\) is of the form
\[
(c, t_c, t_d, m) \in \{0, 1\}^4 \times \mathbb{Z}_N^{2^1},
\]
with \(c = H(r, (h, h_d, C_0, C_1, D_0, D_1), (C'_0, C'_1, D'_0, D'_1))\), and where \((C'_0, C'_1, D'_0, D'_1)\) satisfy (4).

Output the ciphertext 
\(ct \leftarrow (C_0, C_1, D_0, D_1, \pi)\).

- \(\text{Dec}(sk)\): Parse ct as \((C_0, C_1, D_0, D_1, \pi)\) and sk as \((r, x)\) \(\in \mathbb{N} \times \mathbb{Z}\). Return \(\perp\) if \(\pi\) is not a verifying NIZK argument that 
\((h, h_d, C_0, C_1, D_0, D_1) \in \mathcal{L}_{NY}\). Otherwise, return
\[
m = D \log \left( c_1^2 c_0^{2x} \mod N^{\xi+1} \right) \cdot 2^{-1} \mod N^{\xi}.
\]

- UpdateSk(pk): To update \(pk \leftarrow (r, h) \in \mathbb{Z}_N^{2^1+1}\).

1. Pick \(r \leftarrow \mathcal{U}((-B, \ldots, B))\), set \(h' = h \cdot g^r \mod N^{\xi+1}\).
2. Compute \((U_0, V_0, U_1, V_1, \pi)\) as
\[
(U_0, V_0) = \left(g^{t_c} \mod N^{\xi+1}, (1 + N)^r \cdot h^{t_d} \mod N^{\xi+1}\right),
\]
\[
(U_1, V_1) = \left(g^{t_d} \mod N^{\xi+1}, (1 + N)^r \cdot h^{t_d} \mod N^{\xi+1}\right),
\]
for random \(t_c, t_d \leftarrow \mathcal{U}(\mathbb{Z}_{(N-1)/4})\), and \(\pi\) a proof that
\((h, h_d, U_0, V_0, U_1, V_1) \in \mathcal{L}_{NY}\) computed using \((t_c, t_d, r)\), as described in Enc. Note however that here the public key \(h_d'\) replaces \(h_d\) in the Enc algorithm.

3. Using witness \((t_c, r)\), generate a NIZK argument \(\pi_{up}\) that
\((h, h', U_0, V_0) \in \mathcal{L}_{WFU}\), where \(\mathcal{L}_{WFU}\) is the language of Section 4.3. This argument is of the form \(\pi_{up} = (\epsilon \cdot k, r) \in \{0, 1\}^4 \times \mathbb{Z}_N^{2^1 \cdot 2^{23} \cdot B^2}, \)
with \(\epsilon \cdot (u, V', H)\) satisify (5).

Return \(\epsilon \cdot (U_0, V_0, U_1, V_1, \pi, \pi_{up})\), \(pk' = (r + 1, h')\).

- UpdateSk(pk): Given \(sk = (r, x) \in \mathbb{N} \times \mathbb{Z}\) and
\((U_0, V_0, U_1, V_1, \pi, \pi_{up})\), \(pk' = (r', h')\), return \(\perp\) if \(r' \neq r \pm 1\) otherwise.

1. Return \(\perp\) if \((U_0, V_0, U_1, V_1) \notin \mathbb{Z}_N^{2^1+1}\) or \(h' \notin \mathbb{Z}_N^{2^1+1}\).
2. Return \(\perp\) if \(r = (c, t_c, t_d, m) \in \{0, 1\}^4 \times \mathbb{Z}_N^{2^1 \cdot 2^{23} \cdot B^2}, \)
\(c_1^2 c_0^{2x} \mod N^{\xi+1}\) does not parse properly or if it is not a valid NIZK argument that
\((h, h_d', U_0, V_0, U_1, V_1) \in \mathcal{L}_{NY}\).
3. Return \(\perp\) if \(\pi_{up} = (\epsilon \cdot k, r) \in \{0, 1\}^4 \times \mathbb{Z}_N^{2^1 \cdot 2^{23} \cdot B^2}, \)
\(\epsilon \cdot (u, V', H)\) does not parse properly or if it is not a valid NIZK argument that
\((h, h', U_0, V_0) \in \mathcal{L}_{WFU}\).

4. Compute
\[
r = D \log \left( v^2 \cdot u^{-2x} \mod N^{\xi+1} \right) \cdot 2^{-1} \mod N^{\xi}.
\]
5. Let \(\tilde{r} = \min(r, N^{\xi} - r)\) and let \(b_r \in \{0, 1\}\) such that
\[
\tilde{r} = (1 - b_r) \cdot r + b_r \cdot (N^{\xi} - r) \mod N^{\xi}.
\]

Compute \(x' = x + (-1)^{\tilde{r}} \cdot r\) over \(\mathbb{Z}\).

Return \(sk' = (r + 1, x') \in \mathbb{N} \times \mathbb{Z}\).

- VerifyUpdate(\(h = pk, up, pk'\)): Let \(up = (U_0, V_0, U_1, V_1, \pi, \pi_{up})\). Return \(\perp\) if \(\pi\) is not a valid NIZK proof for the statement
\((h, h_d, (U_0, V_0, U_1, V_1)) \in \mathcal{L}_{NY}\). Then, verify that \(\pi_{up}\) is a valid NIZK argument that
\((h, pk', U_0, V_0) \in \mathcal{L}_{WFU}\), in case we return 1, else return \(\perp\).

We now prove that the scheme provides IND-CU-CCA-security in the ROM assuming that the Strong RSA assumption holds and that the scheme of Section 4.2 provides IND-CR-CCA security.

**Theorem 4.** The above construction provides IND-CU-CCA security assuming: (i) The DCR assumption holds; (ii) The NIZK proof for \(\mathcal{L}_{NY}\) provides simulation-soundness; (iii) The NIZK argument for \(\mathcal{L}_{WFU}\) provides computational soundness

**Proof.** The proof uses with a sequence of games. For each \(i, W_i\) denotes the event that the adversary \(A\) outputs \(0\) in Game\(_i\).

Game0: This is the original IND-CU-CCA game where the challenger’s bit is \(b = 0\).

Game1: We replace the proof \(\pi_{up}^*\) in the final update message \(up^* = (U_0^*, V_0^*, U_1^*, V_1^*, \pi_{up}^*)\) by a simulated NIZK proof obtained by programming \(H'\). We have \(|Pr[W_1] - Pr[W_2]| \leq 2^{-\lambda}\) as the distribution of \(\pi_{up}^*\) is statistically unchanged. We note that \(A\) can only see this simulated proof at the very end of the game, after having submitted all its update queries. Therefore, \(A\) never gets to generate a proof for \(\mathcal{L}_{WFU}\) after having seen this simulated proof.

Game2: In this game, we replace the proof \(\pi^*\) of plaintext equality in the final update \(up^* = (U_0^*, V_0^*, U_1^*, V_1^*, \pi_{up}^*)\), by a simulated proof, which is obtained by programming the random oracle \(H\). By Lemma 4, we have \(|Pr[W_3] - Pr[W_4]| \leq 2^{-\lambda}\).

Game3: We now change the \((U_1^*, V_1^*)\) components of \(up^*\) to be an encryption of 0 under the public key \(h_d\). Thanks to the standard IND-CPA security of the Elgamal-Paillier PKE scheme, these two games are indistinguishable under the DCR assumption, and we have \(|Pr[W_5] - Pr[W_6]| \leq Adv_{DCR}^E(\lambda)\).

Game4: We modify the generation of public parameters and now choose \(h_d' = g^s \mod N^{\xi+1}\), where \(s' \leftarrow \mathcal{U}(\mathbb{Z}_{(N-1)/4})\). Since \(h_d'\) remains statistically uniform in the subgroup of \(2N^{\xi}\)-th
residues, the distribution of \( pp \) is statistically close to that of Game0. We have \(|\Pr[W_4] - \Pr[W_5]| \leq 2^{-\lambda} \).

Games5: We introduce the following check: When the adversary makes an update query \((U_5, V_6, U_1, V_1, \pi_1, \pi_{upp}, pk') = ((i+1), h^i')\), the challenger uses the secret key \( sk_i \) to decrypt \((U_5, V_6)\) and check that the underlying plaintext \( r \in \mathbb{Z} \) (as decrypted at step 4 of UpdateSk) is such that \( h^i' = h_1 \cdot g^{-(r^i')^b} \mod N_2^{b+1} \), where \( h_1 \in \mathbb{Z}_{N_2}^{b+1} \) is part of \( pk_i \). It halts if it is not the case. We denote by \( E_5 \) the event that it halts. Games3 and Game4 are identical until event \( E_6 \) occurs and we have \(|\Pr[W_5] - \Pr[W_4]| \leq 0.09 \). The computational soundness of our \( \Sigma \)-protocol for \( L_{WFL} \) (together with the soundness of Fiat-Shamir in the ROM) guarantees that \( \Pr[E_5] \leq \text{Adv}_{\text{CR-SMA}}(\lambda) \). Here, note that the fact that the simulated proof \( \pi_{upp} \) is revealed only at the very end of the game plays a crucial role and allows relying on the standard soundness rather than on simulation-soundness.

Game6: This game is identical to Game3 except that, at each query to \( O_{upp}(-) \), we use \( x^*_j \) instead of the actual secret \( sk_j \) of the current epoch \( i \). Namely, when the adversary makes an update query \((U_5, V_6, U_1, V_1, \pi_1, \pi_{upp}, pk')\) that passes VerifyUpdate, the challenger does no longer use \( sk_j \) to decrypt \((U_5, V_6)\) at step 4 of UpdateSk. Instead, it uses \( x^*_j \) to decrypt \((U_1, V_1)\). Game6 proceeds identically to Game3 until the event \( E_5 \) that an \( O_{upp}(-) \)-query involves pairs \((U_5, V_6), (U_1, V_1)\) that decrypt to distinct values although \( i \) verifies. The statistical soundness of \( \pi \) (which is guaranteed in the ROM when Fiat-Shamir is applied to our \( \Sigma \)-protocol in Section 4.1) ensures that \( \Pr[E_6] \leq 2^{-\lambda} \). We insist that we do not rely on simulation-soundness here since \( E_6 \) occurs before \( A \) is allowed to see \( upp' \) (and thus before it gets to see a simulated proof).

Game7: This final game is identical to the previous game except that the challenger encrypts \( m_1 \) instead of \( m_0 \). Lemma 8 shows that these two games are indistinguishable under the IND-CR-CCA security of our construction from Section 4.2.

We have now switched the challenge ciphertext to be an encryption of \( m_1^* \) instead of \( m_0^* \), and the follow-up of Base games used in our game backwards to go back to the IND-CU-CCA security game where the challenger’s bit is \( b = 1 \).

**Lemma 8.** Game6 and Game7 are computationally indistinguishable if the scheme of Section 4.2 provides IND-CR-CCA security.

**Proof.** Assuming that there exists a PPT adversary \( A \) that can distinguish between Game6 and Game7, we build a PPT adversary \( B \) against the IND-CR-CCA security of the construction in Section 4.2. Algorithm \( B \) receives as input public parameters \( pp \) and \( pk_0 \) from its challenger. It appends \( h^*_j \) to the public parameters \( pp \), where it computes \( h^*_j = h^j \cdot g^{-(r^j)^b} \mod N_2^{b+1} \) by sampling \( x^*_j \). It forwards \((pp', \pi', \{h^*_j\}, pk_0)\) to \( A \). B has access to a random oracle \( H'' \) while \( A \) has access to two random oracles \( H, H' \). Except at the end of this reduction, \( B \) uses \( H'' \) to answer oracle \( H \)-queries while \( H' \) is honestly simulated.

During stages 2 and 4, when \( A \) makes a decryption query, \( B \) submits the same query to its decryption oracle and relays the result to \( A \). When \( A \) makes an update query \((U_5, V_6, U_1, V_1, \pi_1, \pi_{upp}, pk')\), \( B \) uses its knowledge of \( x^*_j \) to decrypt \((U_1, V_1)\). Thanks to the two tweaks introduced in Game3 and Game6, \( B \) is guaranteed to recover the correct randomness \( r \) used by \( A \) to generate its update. Then, \( B \) submits this \( r \) to its own update oracle in the IND-CR-CCA game.

At stage 3, when \( A \) submits a pair of challenge messages \((m_0^*, m_1^*)\), \( B \) submits the same pair to its IND-CR-CCA challenger. It then relays the latter’s challenge ciphertext \( c^* \) to \( A \).

Finally, at stage 6 of the IND-CR-CCA game, \( B \) obtains a tuple \((pk^*, (h^*, h'), sk^*, up^*)\), where \( up^* = (U_5^*, V_6^*) \). It then generates an encryption \((U_1^*, V_1^*)\) of 0 under key \( h'_j \). Let \( h_1^j \) be the public key at the last epoch \( j' \) before moving to \( pk^* \). \( B \) generates a simulated proof \( \pi^* \) for the statement \((h_1^j, h', U_0^*, V_0^*, U_1^*, V_1^*) \in L_{NV} \) by programming the random oracle \( H \). Note that \( H \) must be programmed on an input which is unpredictable to \( A \) (as it is chosen by \( B \)’s challenger), so that a collision on \( H \) occurs with negligible probability \( \leq Q_H \cdot 2^{-\lambda} \). Finally, \( B \) generates a simulated proof \( \pi_{upp}^* \) for the statement \((h_1^j, h', U_0^*, V_0^*) \) by programming the random oracle \( H' \). It then sends \( A \) the tuple \((pk^*, sk^*, (U_0^*, V_0^*, U_1^*, V_1^*, \pi^*, \pi_{upp}^*))\). When \( A \) halts and outputs \( b^* \in \{0, 1\} \), \( B \) outputs the same \( b^* \).

The above reduction shows \(|\Pr[W_7] - \Pr[W_6]| \leq \text{Adv}_{\text{CR-CCA}}(\lambda) + Q_H \cdot 2^{-\lambda} \), where \( \text{Adv}_{\text{CR-CCA}}(\lambda) \) denotes the advantage in the proof of Theorem 3.

5 IMPLEMENTATION AND PERFORMANCES

The main advantage of our scheme resides in that its parameters make it close to practical use, in terms of key size, ciphertext size, and running time. In this section, we compare the key, ciphertext and update sizes to the previous UPKE constructed in the standard model by Dodis et al. in [21] and to the one constructed in the ROM by Jost et al. in [34]. For the sake of completeness, we also present the running times of our "proof of concept" implementation.

5.1 Key/Ciphertext/Update Sizes

Table 1 compares the different key/ciphertext/update sizes achieved by the existing UPKE construction and their security assumptions for a 128-bit strength security. One can remark that we avoid the quadratic blowup suffered by the Dodis et al. scheme in the update sizes thanks to DCR. This allows us to construct the first IND-CR-CCA secure UPKE with both reasonable key, ciphertext and update sizes in the standard model.

Jost et al.’s scheme remains the most efficient to date. However, its IND-CR-CPA security fully relies on the ROM and it does not extend to provide the stronger security notions that we are able to achieve with DCR.

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Encryption Size</th>
<th>Decryption Size</th>
<th>Update Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>IND-CR-CPA (112 bits)</td>
<td>0.021 s</td>
<td>0.020 s</td>
<td>0.042 s</td>
</tr>
<tr>
<td>IND-CR-CPA (128 bits)</td>
<td>0.062 s</td>
<td>0.062 s</td>
<td>0.127 s</td>
</tr>
<tr>
<td>IND-CR-CCA (112 bits)</td>
<td>0.045 s</td>
<td>0.042 s</td>
<td>0.085 s</td>
</tr>
<tr>
<td>IND-CR-CCA (128 bits)</td>
<td>0.092 s</td>
<td>0.091 s</td>
<td>0.202 s</td>
</tr>
</tbody>
</table>

Table 2: Benchmarks on our implementation of our IND-CR-CPA and IND-CR-CCA schemes

5.2 Running Time

We implemented our IND-CR-CPA and IND-CR-CCA schemes in C/C++ as a proof of concept. Our implementation relies on the...
GMP\textsuperscript{3} library to handle computations on big integers. One should note that, being only a proof of concept, the implementation is heavily optimizable. No parallelization is used and a few choices were made to ease the implementation and rather harm the performances.

Our benchmarks were made on an Apple M1 CPU with 8 cores running at 3Ghz, under macOS 12. They are presented in table 2. The code was compiled with clang (clang-1205.0.22.9) with the optimization flag -O3. The running time of each function was estimated by taking the mean running time of 1000 evaluations. For the IND-CR-CPA version, encryptions, decryptions and updates sensibly have the same running time. Simply because an update consists of an encryption and a decryption, and that decryptions require the same operations as encryptions.

The first gap appears when introducing NIZKs for Naor-Yung. As updates do not require proofs, they are a bit faster than encryptions. IND-CR-CCA security comes at the cost of losing a factor 20 in efficiency in our benchmarks. We believe that those results are encouraging for future real-world implementations of UPKEs.

ACKNOWLEDGEMENTS

We thank the reviewers for constructive comments. Part of this research was funded by the French ANR ALAMiBC project (ANR-16-CE39-0006). The first author was funded by the Direction Générale de l’Armement (Pôle de Recherche CYBER).

REFERENCES


\[5\] M. Bellare and B. Yee. 2003. Forward-secure in private-key cryptography. In CT-RSA.


\[39\] M. Naor and M. Yung. 1999. Public-key cryptosystems provably secure against chosen ciphertext attacks. In TCC.


\[43\] O. Regev. 2005. On lattices, learning with errors, random linear codes, and cryptography. In STOC.

\[44\] A. Sahai. 1999. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In FOCS.

A COMPLEMENTARY DEFINITIONS:

ZERO-KNOWLEDGE PROOFS AND Σ-PROTOCOLS

We recall standard definitions of zero-knowledge proofs [29].

interactive proof system (IPS) for a language \( L \) is a two-party protocol between a prover \( P \) and a verifier \( V \), where the former wants to convince the latter that a statement \( x \) belongs to \( L \). An IPS is zero-knowledge when this can be achieved without having \( P \) leak any information beyond the fact that \( x \in L \). This is formalized by requiring the existence of a zero-knowledge simulator \( S \) that produces transcripts indistinguishable from real conversations between \( P \) and \( V \). An IPS provides soundness when even an unbounded \( P \) cannot trick \( V \) into accepting a proof of a false statement. When the soundness condition only holds for polynomial-time cheating provers, an IPS is called an argument. A zero-knowledge proof/argument system is

\( \text{GMP} \) https://gmplib.org
We require a value \( \Sigma \) associated with a relation \( R \). A 3-move interactive protocol \( \Pi = (P, V) \) is a \( \Sigma \)-protocol for \( L \) if it satisfies the following:

- **3-Move Form:** \( P \) and \( V \) proceed as follows: (i) \( P \) inputs \( (x, w) \in R \), computes \( (a, st) \leftarrow P(x, w) \) and sends \( a \) to \( V \); (ii) \( V \) sends back a random challenge \( c \); (iii) \( P \) sends a response \( z = P(x, w, a, c, st) \) to \( V \); (iv) \( P \) outputs 1 or 0.

- **Completeness:** If \( (x, w) \in R \) and \( P \) honestly computes \( (a, z) \) for a challenge \( c \), then \( V(x, (a, c, z)) \) outputs 1 with probability \( 1 - \text{negl}(\lambda) \).

- **Special soundness:** There exists a \( P \)-PTK knowledge extractor \( E \) such that, for any statement \( x \), on input of two accepting transcripts \( (a, c, z) \) and \( (a', c', z') \) with \( c \neq c' \), outputs a witness \( w' \) such that \((x, w') \in R \).

- **Special honest-verifier zero-knowledge:** There is a \( P \)-PTK simulator \( S \) that inputs \( x \in L \) and a challenge \( c \in C \). It outputs \( (a, z) \leftarrow S(x; c) \) such that \((a, c, z)\) is indistinguishable from a real transcript \((x, w) \in R \) with challenge \( c \).

The special soundness property can be relaxed to hold in the computational sense. In this case, we require that no \( P \)-PTK cheating prover be able to find two transcripts \((a, c, z)\) with \( c \neq c' \) for which the extractor fails to compute a witness.

In order to prove plaintext equalities via the Naor-Yung paradigm, we require a \( \Sigma \)-protocol satisfying the following property:

**Definition 11 ([23], Quasi-unique responses).** Let \( \lambda \in \mathbb{N} \) a security parameter. A \( \Sigma \)-protocol \((P, V)\) has quasi-unique responses if, for any \( P \)-PTK algorithm \( A \), it holds that

\[
\text{Pr}[V(x, a, c, z) = V(x, a, c, z') = 1 \mid z \neq z' \mid (x, a, c, z, z') \leftarrow A(1^\lambda)] = \text{negl}(\lambda).
\]

**Removing Interaction.** In the random oracle model, the Fiat-Shamir heuristic [24] provides an efficient way to build NIZK proofs by collapsing interactive \( \Sigma \)-protocols. The interaction between \( P \) and \( V \) is removed by replacing the latter’s challenge with a hash value \( H(x, a) \) computed by the prover, where \( H \) is a hash function modeled as a random oracle. The resulting NIZK protocol is denoted by \((P^H, V^H)\).

In the zero-knowledge simulation, the simulator \( S \) is allowed to program the random oracle at arbitrary points. As in [23], we model it as a stateful algorithm that operates in two modes: a mode \((h_i, st) \leftarrow S(1, st, q_i)\) that handles random oracle queries (typically via lazy sampling); and a second mode \((\pi, st) \leftarrow S(2, st, x)\) that simulates actual proofs. The different calls to \( S(1, \cdot, \cdot) \) and \( S(2, \cdot, \cdot) \) share a common state \( st \) which is updated after each operation.

**Definition 12 (NIZK in the ROM).** Let an NP language \( L \) associated with a relation \( R \). Let \((S_1, S_2)\) denote oracles such that \( S_1(q_i) \) returns the first output of \((h_i, st) \leftarrow S(1, st, q_i)\) and \( S_2(x, w) \) returns the first output of \((\pi, st) \leftarrow S(2, st, x)\) if \((x, w) \in R \). A non-interactive protocol \((P^H, V^H)\) is zero-knowledge for \( L \) in the random oracle model if there exists a \( PPT \) simulator \( S \) such that, for any \( PPT \) distinguisher \( D \), we have

\[
| \text{Pr}[D^{H(\cdot), P^H(\cdot)(1^\lambda)} = 1] - \text{Pr}[D^{S_1(\cdot), S_2(\cdot)(1^\lambda)} = 1] | = \text{negl}(\lambda),
\]

where oracles \( P \) and \( S_2 \) both output \( \perp \) on input of \((x, w) \notin R \).

It is known that, in the random oracle model, the Fiat-Shamir transform compiles \( \Sigma \)-protocols into NIZK protocols.

**Theorem 5.** Let \( \lambda \) be a security parameter. Consider a \( \Sigma \)-protocol \( \Pi = (P, V) \) for a language \( L \) in NP. Let \( H \) be a function that ranges over the challenge space of \( \Pi \). In the random oracle model, the proof system \( \Pi^{FS} = (P^H, V^H) \) derived from \( \Pi \) by applying the Fiat-Shamir transform is a NIZK argument system for \( L \) if the special soundness of \( \Pi \) holds (in the statistical or computational sense). Moreover, if the special honest-verifier zero-knowledge property of \( \Pi \) holds in the statistical sense (resp. computational), then \( \Pi^{FS} \) provides statistical (resp. computational) ZK.

**Simulation-soundness.** The soundness property of a NIZK proof ensures that no cheating prover \( P^* \) can come up with a convincing proof for a false statement. The notion of simulation-soundness [44] requires that soundness be preserved even if a cheating prover is allowed to see simulated proofs (generated by the NIZK simulator) for possibly false statements chosen by \( P^* \). In the random oracle model, we use a definition of simulation-soundness given in [23].

**Definition 13 (Unbounded simulation-soundness).** Let \( L \) be an NP language. Consider a proof system \((P^H, V^H)\) for \( L \) where the zero-knowledge simulator is denoted by \( S \). Denote by \((S_1, S_2)\) the oracles such that \( S_1(q_i) \) returns the first output of \((h_i, st) \leftarrow S(1, st, q_i)\) and \( S_2(x) \) returns the first output of \( S(2, st, x) \) (possibly with \( x \notin L \)). A protocol \((P^H, V^H)\) is (unbounded) simulation-sound with respect to \( S \) in the random oracle model if, for any \( PPT \) adversary \( A \), we have

\[
| \text{Pr}[(x^*, \pi^*) \leftarrow A(S_1(\cdot), S_2(\cdot)(1^\lambda)), (x^*, \pi^*) \leftarrow T] - \text{Pr}[(x^*, \pi^*) \notin T \wedge x^* \notin L \wedge V^H(x^*, \pi^*) = 1] | = \text{negl}(\lambda),
\]

where \( T \) is the list of pairs \((x_i, \pi_i)\) such that \( x_i \) was queried to the simulator and \( \pi_i \) was the latter’s response.

Fouque and Pointcheval [25] showed that, in the random oracle model, certain \( \Sigma \)-protocols are turned into simulation-sound NIZK proofs by applying the Fiat-Shamir heuristic. Their proof applies to a restricted family of IPS. In [23], Faust et al. gave a more general result that is used in our CCA-secure constructions.

**Theorem 6 ([23]).** Consider a non-trivial \( \Sigma \)-protocol \((P, V)\) for an NP language \( L \), which satisfies the quasi-unique response property of Definition 11. In the random oracle model, the proof system \((P^H, V^H)\) derived from \((P, V)\) via the Fiat-Shamir transform is a simulation-sound NIZK.

We note that, unlike the result of Fouque and Pointcheval [42], Theorem 6 does not rely on the forking lemma [42] and thus provides tighter concrete security bounds.
B DEFERRED PROOFS

B.1 Proof of Lemma 3

Proof. Let us assume that a prover can come up with two valid transcripts \((C'_0, C'_1, D'_0, D'_1, c, \{i_c, \tilde{i}_d, \tilde{i}_d', m\})\), \((C''_0, C''_1, D''_0, D''_1, c, \{i_c, \tilde{i}_d, \tilde{i}_d', \tilde{m}\})\). We show that \((h, h, C_0, C_1, D_0, D_1) \in L_{NY}\).

We define \(\Delta c = c_1 - c_2, \Delta m = \tilde{m}_1 - \tilde{m}_2 \pmod {N^k}\). Then, if we raise all members of (11) to the power \(\hat{c} = \Delta c^{-1} \pmod {N^k}\), we obtain

\[
\begin{align*}
C_0' & = g^{2^{\hat{c}}} \pmod {N^{\zeta + 1}}, \\
C_1' & = \left(1 + N\right)^{2\Delta m} \cdot h^{2^{\hat{c}} \cdot \hat{c}} \pmod {N^{\zeta + 1}}, \\
D_0' & = g^{2^{\hat{c}} \cdot \hat{d}} \pmod {N^{\zeta + 1}}, \\
D_1' & = \left(1 + N\right)^{2\Delta m} \cdot h^{2^{\hat{c}} \cdot \hat{d}} \pmod {N^{\zeta + 1}},
\end{align*}
\]

It follows that

\[
\begin{align*}
C_0 & = g^{2^{\hat{c}} \cdot \hat{c}} \pmod {N^{\zeta + 1}}, \\
C_1 & = \left(1 + N\right)^{\tilde{m}_1} \cdot h^{2^{\hat{c}} \cdot \hat{c}} \pmod {N^{\zeta + 1}}, \\
D_0 & = g^{2^{\hat{c}} \cdot \hat{d}} \pmod {N^{\zeta + 1}}, \\
D_1 & = \left(1 + N\right)^{\tilde{m}_1} \cdot h^{2^{\hat{c}} \cdot \hat{d}} \pmod {N^{\zeta + 1}},
\end{align*}
\]

Since \(|c \cdot t_c, |c \cdot t_d| < 2\beta \cdot \frac{N-1}{4}\), Lemma 2 implies

\[
\left(\bar{i}_c + c \cdot t_c, \bar{i}_d + c \cdot t_d\right) \Rightarrow U([0, 2\beta \cdot \frac{N-1}{4}]) \times U([0, 2\beta \cdot \frac{N-1}{4}])
\]

and we also have \(\tilde{m} \cdot c \pmod {N^k} \mid m \leftarrow U(\mathbb{Z}_{N^k}) \Rightarrow U(\mathbb{Z}_{N^k})\).

Since \((C'_0, C'_1, D'_0, D'_1)\) is uniquely determined by the (true) statement and the challenge-response pair \((c, \{i_c, \tilde{i}_d, \tilde{m}\})\) in the verification equations, our simulated transcript is statistically close to a real transcript.

\(\square\)

B.2 Proof of Lemma 4

Proof. In the special honest verifier setting, the simulator is given a challenge \(c\) and has to produce a properly distributed accepting transcript \((h, h, C_0, C_1, D_0, D_1) \in L_{NY}\). We can uniformly sample \(\{i_c, \tilde{i}_d\} \leftarrow U([0, 2\beta \cdot \frac{N-1}{4}])\), \(m \leftarrow U(\mathbb{Z}_{N^k})\) and then set

\[
\begin{align*}
C'_0 & = C_0^{2c} \cdot g^{2^{\hat{c}}} \pmod {N^{\zeta + 1}}, \\
C'_1 & = C_1^{2c} \cdot \left(1 + N\right)^{2\hat{m}_d} \cdot h^{2^{\hat{c}} \cdot \hat{d}} \pmod {N^{\zeta + 1}}, \\
D'_0 & = D_0^{2c} \cdot g^{2^{\hat{c}} \cdot \hat{d}} \pmod {N^{\zeta + 1}}, \\
D'_1 & = D_1^{2c} \cdot \left(1 + N\right)^{2\hat{m}_d} \cdot h^{2^{\hat{c}} \cdot \hat{d}} \pmod {N^{\zeta + 1}},
\end{align*}
\]

This is a valid transcript as it made to satisfy all conditions of the verification step. Since \((h, h, C_0, C_1, D_0, D_1) \in L_{NY}\), we know that \(\exists t_c, \tilde{t}_d \in \mathbb{Z}_{pq}, m \in \mathbb{Z}_{N^k}\) such that

\[
\begin{align*}
C_0'' & = g^{2^{\hat{c}} \cdot \hat{c}} \pmod {N^{\zeta + 1}}, \\
C_1'' & = \left(1 + N\right)^{2\hat{m}_d} \cdot h^{2^{\hat{c}} \cdot \hat{d}} \pmod {N^{\zeta + 1}}, \\
D_0'' & = g^{2^{\hat{c}} \cdot \hat{d}} \pmod {N^{\zeta + 1}}, \\
D_1'' & = \left(1 + N\right)^{2\hat{m}_d} \cdot h^{2^{\hat{c}} \cdot \hat{d}} \pmod {N^{\zeta + 1}},
\end{align*}
\]

\(\#By\ applying\ the\ same\ arguments\ as\ in\ the\ proof\ of\ Lemma\ 6,\ we\ can\ show\ that,\ unless\ the\ cheating\ prover\ is\ able\ to\ compute\ a\ non-trivial\ root\ of\ g \in \mathbb{Z}_{N^{\zeta + 1}}\, \Delta c\ divides\ both\ \Delta t_c\ and\ \Delta t_d,\ so\ that\ witnesses\ t_c\ and\ \tilde{t}_d\ are\ extractable.\ However, we\ do\ not\ rely\ on\ this\ property\ as\ we\ only\ aim\ at\ soundness\ (rather\ than\ extractability).\)
where $h_f = g^{r_f} \mod N^{c+1}$ denotes the public key at epoch $\ell$. At stage 6, $A$ obtains $(sk^*, ap^*)$, where

$$ap^* = (U^*, V^*) = \left( g^{k^*} \mod N^{c+1}, (1 + N)r^* \cdot h_f^{k^*} \mod N^{c+1} \right),$$

where $r^* \leftarrow \mathcal{U}([-B, B])$ and $h_f \in \mathbb{Z}_{N^{1/4}}$ is the public key at stage 5. Finally, $A$ outputs a bit $b^* \in \{0, 1\}$ and wins if $b^* = 0$. The latter event is called $W_0$.

Game 1: In this game, we modify the generation of public parameters and now choose $h_d$ as $h_d = g^{x_d} \mod N^{c+1}$, where $x_d \leftarrow \mathcal{U}(\mathbb{Z}(N-1)/4)$. Since $h_d$ remains statistically uniform in the subgroup of $2N^{c+1}$-th residues, the distribution of $pp$ is statistically close to that of Game 0. We have $|Pr[W_2] - Pr[W_4]| \leq 2^{-3}$.

Game 2: We change the decryption oracle. For a decryption query $ct = (C_0, C_1, D_0, D_1, \pi)$ at epoch $i$, the challenger does no longer use the current secret key $sk_i$. Instead, it uses $x_d$ to decrypt $(D_0, D_1)$ by computing

$$m = DLog \left( D_0^2 \cdot D_0^{2x_d} \mod N^{c+1} \right) \cdot 2^{-1} \mod N^c.$$ Clearly, Game 2 is perfectly indistinguishable from Game 1 until the event that $A$ queries $O_{dec}(\cdot)$ on a ciphertext for which $(C_0, C_1)$ and $(D_0, D_1)$ decrypt to distinct messages although $\pi$ is a valid proof for $L_{NY}$. However, Lemma 3 and the Fiat-Shamir heuristic ensure that, in the random oracle model, this can only occur with negligible probability: Concretely, for a false statement and any first prover message, a valid response exists for at most one challenge. Hence, we have $|Pr[W_3] - Pr[W_5]| \leq 2^{-3}$.

Game 3: We modify the challenge $ct^* = (C_0^*, C_1^*, D_0^*, D_1^*, \pi^*)$. The NIZK proof $\pi^*$ that $(h_f, h_d, C_0^*, C_1^*, D_0^*, D_1^*) \in L_{NY}$ is now simulated by programming the random oracle $H$ and using the NIZK simulator of Lemma 4. By Lemma 4, $|Pr[W_1] - Pr[W_5]| \leq Q_H \cdot 2^{-3}$ as the distribution of $\pi^*$ is statistically unchanged.

Game 4: We change the distribution of $ct^* = (C_0^*, C_1^*, D_0^*, D_1^*, \pi^*)$. In this game, the challenger computes

$$(C_0^*, C_1^*) = \left( g^{r^*_f} \mod N^{c+1}, (1 + N)^m_1 \cdot h_f^{r^*_d} \mod N^{c+1} \right),$$

$$(D_0^*, D_1^*) = \left( g^{r^*_d} \mod N^{c+1}, (1 + N)^m_1 \cdot h_f^{r^*_d} \mod N^{c+1} \right)$$

where $r_0, r_4 \leftarrow \mathcal{U}(\mathbb{Z}(N-1)/4)$ and $h_f = g^{r_f} \mod N^{c+1}$ is the public key at epoch $\ell$.

The IND-CPA security of the UPKE scheme in Section 3.2 implies - via a reduction that proceeds identically to that in the proof Theorem 2 - the indistinguishability of Game 4 and Game 3 under the DCR assumption. We have $|Pr[W_6] - Pr[W_5]| \leq \text{Adv}^{\text{DCR}}(\ell)$.

Game 5: We change again the decryption oracle. For a decryption query $ct = (C_0, C_1, D_0, D_1, \pi)$ at any epoch $i$, the challenger does no longer decrypt $(D_0, D_1)$ using $x_d$. Instead, it comes back to decrypting $(C_0, C_1)$ using the current secret key $sk_i$, by computing

$$m = DLog \left( C_0^2 \cdot C_0^{2-2sk_i} \mod N^{c+1} \right) \cdot 2^{-1} \mod N^c.$$ We note that Game 5 is perfectly indistinguishable from Game 4 until $A$ queries $O_{dec}(\cdot)$ on a ciphertext $ct$ where $(C_0, C_1)$ and $(D_0, D_1)$ decrypt to distinct messages even though $\pi$ is a verifying proof for $L_{NY}$. Here, the simulation-soundness of the NIZK construction in Section 4.1 - which holds in the random oracle model assuming that factoring is hard when we apply the Fiat-Shamir heuristic - ensures that this only occurs with negligible probability if the DCR assumption holds. We have $|Pr[W_7] - Pr[W_5]| \leq \text{Adv}^{\text{sim-sound}}(\ell)$.

Game 6: We change again the distribution of the challenge ciphertext $ct^* = (C_0^*, C_1^*, D_0^*, D_1^*, \pi^*)$. In the challenge phase, the challenger now computes

$$(C_0^*, C_1^*) = \left( g^{r^*_f} \mod N^{c+1}, (1 + N)^m_1 \cdot h_f^{r^*_d} \mod N^{c+1} \right),$$

$$(D_0^*, D_1^*) = \left( g^{r^*_d} \mod N^{c+1}, (1 + N)^m_1 \cdot h_f^{r^*_d} \mod N^{c+1} \right)$$

where $r_0, r_4 \leftarrow \mathcal{U}(\mathbb{Z}(N-1)/4)$ and $h_f = g^{r_f} \mod N^{c+1}$ is the public key at epoch $\ell$. The IND-CPA security of the Elgamal-Paillier PKE scheme ensures that $\pi^*$ is indistinguishable from $\pi$ under the DCR assumption: i.e., $|Pr[W_6] - Pr[W_5]| \leq \text{Adv}^{\text{DCR}}(\ell)$.

Game 7: In the generation of the challenge $ct^* = (C_0^*, C_1^*, D_0^*, D_1^*, \pi^*)$, we change again the generation of $\pi^*$, which is now computed as a real proof using the witnesses $r_0, r_4 \leftarrow \mathcal{U}(\mathbb{Z}(N-1)/4)$ and $h_f = g^{r_f} \mod N^{c+1}$ is the public key at epoch $\ell$. The IND-CPA security of the Elgamal-Paillier PKE scheme ensures that $\pi^*$ is indistinguishable from $\pi$ under the DCR assumption as the latter implies the simulation-soundness of our NIZK argument for $L_{NY}$.

$^7$This includes the case of a decryption query on the challenge ciphertext $ct^*$ for an epoch $i > \ell$. We included the epoch number among the inputs of the hash function $H$ to cover this case: If $A$ can come up with a valid proof $\pi^*$ for $(C_0^*, C_1^*, D_0^*, D_1^*)$ at a later epoch $i > \ell$, we have $\pi^* \neq \pi^*$ with probability $1 - 2^{-3}$. $^8$Alain: Clarifier d’où on obtient la statistical soundness (cf mon mail)