Updatable Public Key Encryption in the Standard Model

Yevgeniy Dodis[1*], Harish Karthikeyan[1], and Daniel Wichs[2†]

[1] New York University
{dodis,karthik}@cs.nyu.edu
[2] Northeastern University and NTT Research
wichs@ccs.neu.edu

Abstract. Forward security (FS) ensures that corrupting the current secret key in the system preserves the privacy or integrity of the prior usages of the system. Achieving forward security is especially hard in the setting of public-key encryption (PKE), where time is divided into periods, and in each period the receiver derives the next-period secret key from their current secret key, while the public key stays constant. Indeed, all current constructions of FS-PKE are built from hierarchical identity-based encryption (HIBE) and are rather complicated.

Motivated by applications to secure messaging, recent works of Jost et al. (Eurocrypt'19) and Alwen et al. (CRYPTO'20) consider a natural relaxation of FS-PKE, which they term *updatable PKE* (UPKE). In this setting, the transition to the next period can be initiated by any sender, who can compute a special *update ciphertext*. This ciphertext directly produces the next-period public key and can be processed by the receiver to compute the next-period secret key. If done honestly, future (regular) ciphertexts produced with the new public key can be decrypted with the new secret key, but past such ciphertexts cannot be decrypted with the new secret key. Moreover, this is true even if all other previous-period updates were initiated by untrusted senders.

Both papers also constructed a very simple UPKE scheme based on the CDH assumption in the random oracle model. However, they left open the question of building such schemes in the standard model, or based on other (e.g., post-quantum) assumptions, without using the heavy HIBE techniques. In this work, we construct two efficient UPKE schemes in the standard model, based on the DDH and LWE assumptions, respectively. Somewhat interestingly, our constructions gain their efficiency (compared to prior FS-PKE schemes from the same assumptions) by using tools from the area of circular-secure and leakage resilient public-key encryption schemes (rather than HIBE).

*Partially supported by gifts from VMware Labs and Google, and NSF grants 1619158, 1319051, 1314568.

†Partially supported by NSF grants CNS-1413964, CNS-1750795 and the Alfred P. Sloan Research Fellowship.

1 Introduction

For privacy applications, Forward Security (FS) refers to the ability to update sensitive information in a way that: (1) the system continues to be functional in the future; (2) compromise of the current secret state of the system does not affect the privacy of past secrets. For example, the famous authenticated Diffie-Hellman key agreement protocol, — where the party use long-term signing keys to authenticate the ephemeral public values g^a and g^b used to produce the shared key $k = g^{ab}$, — is forward-secure under the Decisional Diffie-Helman (DDH) assumption, even if the attacker later learns the long-term signing keys, as long as the no-longer-needed ephemeral secrets a and b were erased before this compromise.

In the symmetric-key world, forward security is also easy using a pseudorandom generator (PRG) G, provided the sender and the receiver can stay synchronized [12]. Given the current state s, the sender can produce $(r, s') \leftarrow G(s)$ to get the one-time symmetric key r, and the next state s', so that compromising s' does not affect the security of the one-time key r. One way to think about this is that PRGs allow one to produce an initial state s_0 that defines a "one-way chain" of pseudorandom states $s_0 \to s_1 \to s_2 \to \ldots$, which can only be traversed in the forward direction.

Forward-Secure PKE. Coming back to the public-key world, achieving FS for (non-interactive) Public-Key Encryption (PKE) turned out to be noticeably more complicated. The initial paper of Canetti et al. [24], — which up to this day is still essentially the state-of-the-art in the area, — defined FS-PKE as follows. The key generation outputs initial keys $(\mathsf{pk}_0, \mathsf{sk}_0)$, which implicitly defined two synchronized chains $\mathsf{pk}_0 \to \mathsf{pk}_1 \to \mathsf{pk}_2 \to \ldots$ and $\mathsf{sk}_0 \to \mathsf{sk}_1 \to \mathsf{sk}_2 \to \ldots$ which can be *independently* produced by multiple senders and a single receiver. The chains should be consistent in the sense that messages encrypted under pk_i should be decryptable by sk_i, and the secret-key chain should have the "one-wayness" property we want: exposing sk_i should not compromise the privacy of messages encrypted under prior public keys pk_j, for $j < i$.[3]

Canetti et al. [24] also showed how to build FS-PKE from any Hierarchical Identity-Based Encryption (HIBE) [35,34] scheme. As a result, as more HIBE schemes got built [14,25,15,31,30,22], we also get more FS-PKE schemes, even including purely theoretical schemes from very basic assumptions, like DDH/CDH, factoring, and super-low-noise LPN [31,30,22]. However, most of these schemes are quite complicated and inefficient (or rely on pairings/strong assumptions **??**), at least compared to many simple PKE schemes that are available today. Unfortunately, closing the efficiency/complexity gap between FS-PKE and PKE remains open until this day.

Updatable PKE. As a step towards closing this gap, and motivated by independently interesting applications in the area of secure messaging schemes, two

[3]For efficiency reasons, [24] also insisted that $\mathsf{pk}_i = (\mathsf{pk}_0, i)$, meaning one can quickly go from pk_0 to pk_i, but this point will not be important for our discussion.

recent works of Jost *et al.* [38] and Alwen *et al.* [5] defined a relaxed notion of forward-secure PKE, called *updatable PKE* (UPKE). In this setting, any sender can initiate a "key update" by sending a special update ciphertext.[4] This ciphertext updates the receiver's public key and also, once processed by the receiver, will update their secret key. A malicious sender cannot harm security by sending a malicious key update generated with bad randomness. However, an honest sender is assured that, once the receiver processes an honestly generated key update, all ciphertexts produced in the past will remain secure even if the receiver's secret key is compromised in the future.

Note that in the setting of updatable PKE, we implicitly assume that there is an ordered sequence of update ciphertexts which will be decrypted by the receiver and that each sender can see this sequence and figure out the current public key at any time. For example, this holds if all the ciphertexts are sent by the same sender [38] (similar to the symmetric-key setting), or the ciphertexts could be sent by multiple senders, but there is some outside mechanism that will anyway serialize all these ciphertexts [5].

In the extreme cases, such as a secure group messaging application of [38], every sender S will initiate a key update after *each* ciphertext that it sends to the receiver R.[5] In particular, since S "trusts itself", S can be sure that the information in this ciphertext will be secure forever, the moment that R decrypts it (and moves its secret key forward), even if R gets corrupted later on. And this is true even if other senders S' that sent messages in the past were more careless, and might not have properly generated/erased the randomness of their updates, which were used to determine the public key under which S encrypted its message. In this sense, UPKE with key updates following every encryption provides a natural public-key analog of stream ciphers extensively used in symmetric-key cryptography, despite having multiple senders who do not necessarily trust each other's randomness. To put it differently, the security of each sender only depends on the quality/secrecy of its own randomness, while the correctness relies on the serialization of update ciphertexts sent by all the senders.

UPKE Syntax and Prior Constructions. A bit more precisely,[6] in addition to standard key generation, encryption, and decryption algorithms, the UPKE schemes have two special algorithms $\mathsf{Upd\text{-}Pk}$ and $\mathsf{Upd\text{-}Sk}$. Any sender can run

[4]Of course, FS is trivial to achieve if the receiver can initiate the key update. Indeed, this type of key update is also happening in the secure messaging applications of [38,5], trivially achieving FS when the receiver "speaks" and updates its key. However, we could also be in the scenario where the receiver is non-communicating for a long period of time, while many messages are being sent to *and processed by* the receiver. For example, the receiver could be part of a large secure messaging group [5] who only reads messages, but almost never posts messages. UPKE is precisely useful in this scenario.

[5]For the sake of generality, we will not necessarily insist on updating the public key after each ciphertext, but such extreme use is certainly an option for getting higher security.

[6]We use slightly different syntax than [38,5], but all our schemes are easily converted to meet the syntax of [38,5].

$\mathsf{Upd\text{-}Pk}$ on the current public-key pk_{i-1} to produce *update ciphertext* up_i and a new public key pk_i. In turn, the receiver will run $\mathsf{Upd\text{-}Sk}$ on the current secret key sk_{i-1} and update ciphertext up_i, to produce the new secret-key sk_i.

In terms of forward security, we require that exposure of any key sk_i should not compromise the privacy of messages encrypted under prior public keys pk_j, for $j < i$, *provided at least one "good" update happened from period j to i*. Hereby "good update" we mean that the randomness used by the sender to generate this update was not compromised by the attacker. Indeed, in the secure messaging applications of [38,5], all senders were assumed to be honest, although some of their local randomnesses could be compromised by the attacker. Following [5], our definition will actually be slightly stronger in that we will allow malicious randomness for the "bad updates". See Section 3.

To validate the usefulness of this relaxation, the works of [38,5] also gave an extremely fast and simple construction, which is orders of magnitude faster than HIBE-based FS-PKE schemes and is based on the CDH assumption in the random oracle model (ROM). The encryption scheme is just the standard hashed ElGamal encryption scheme: given public key $h = g^s$, encryption of m computes $(g^r, Hash(h^r) \oplus m)$, while decryption of (c, w) outputs $w \oplus Hash(c^s)$. To update the public/secret key, the sender chooses a random exponent δ, and simply *encrypts it using the standard encryption algorithm*. The new public key is $h' = hg^\delta$, and new secret key is $s' = s + \delta$. We notice, however, the random oracle model is critically used to break the circularity between encrypting the value δ, and later leaking the value $s' = s + \delta$ which depends on the secret key.

Our Main Contribution. In this work we build two efficient UPKE schemes in the *standard model*. Our first scheme is based on the DDH assumption and is approximately "security parameter less efficient" compared to the Hashed ElGamal UPKE scheme described above. Our second construction is also quite efficient and is based on the Learning-With-Errors (LWE) assumption. In particular, it gives the first efficient UPKE based on an assumption that is believed to be post-quantum secure. This was not known even in the random oracle model.

A rough summary of efficiency, security, usability, and assumptions trade-off for our schemes when compared to previous PKE, UPKE, and FS-PKE schemes is given in Table 1. It is clear from the table that our efficiency falls between that of PKE and FS-PKE (much closer to the former), we achieve the same (resp. much stronger) forward security as FS-PKE (resp. PKE), but we do require a stronger synchronization assumption than FS-PKE.

1.1 Our Technique: Using Circular Security and Leakage-Resilience

Looking at the random-oracle-based UPKE scheme of [38,5], we observe that the attacker learns the value $s' = s + \delta$, but also an encryption of δ. Namely, the attacker simultaneously gets: (a) encryption of (some function of the) secret-key, and (b) some leakage s' on the secret-key s. Of course, in that particular scheme, the leakage in (b) was trivial, since δ was completely random and therefore s' is

Table 1. Comparison of Different Primitives. (κ is security parameter.)

Factors	**PKE**	**UPKE** [38,5] RO Model	**UPKE** (this work) Standard Model	**FS-PKE**
Efficiency	Very Efficient	$\approx$ PKE	$\approx$ PKE $\cdot\, \kappa$	Inefficient compared to PKE [7] (from HIBE)
Assumptions	DDH/CDH, Factoring, LWE	CDH	DDH, LWE	DDH/CDH, Factoring, LWE
Forward Security?	No	Yes	Yes	Yes
Synchronization	None	Strong (Updates)	Strong (Updates)	Weak (Time Periods)

completely independent of s, while (so-called) key-dependent-message (KDM) security [13] in (a) was easily handled by the random oracle.

Nevertheless, we will find the abstractions in (a)+(b) useful when going to the standard model. In particular, we will follow the same template, but rely on circular-secure encryption schemes in the standard model under DDH/LWE (e.g., [17,8]) where the key s and/or the updates δ must consist of "small" values in some larger group $\mathbb{Z}_p$ and hence the leakage $s' = s + \delta$ in part (b) will no longer be trivial.[8] Luckily, these schemes are also leakage resilient and hence this non-trivial leakage does not hurt security.

Indeed, modulo several important technicalities,[9] both of our standard model constructions will effectively build UPKE from a regular PKE, which satisfies the following three properties simultaneously:

1. *Circular-secure and leakage-resilient (CS+LR):* Given encryption of the secret key s and any bounded-entropy leakage on s, the scheme is still semantically secure. See Section 4 for the precise definition.

2. *Key-Homomorphic*: Given a public key and a ciphertext pair that corresponds to some secret key s, together with some offset δ, we can convert them into a public key and a ciphertext pair that corresponds to the secret key $s' = s + \delta$ while preserving the encrypted message.

3. *Message-Homomorphic*: Given a ciphertext encrypting some value s, and offset s', we can convert it into a ciphertext encrypting $s' - s$.

Note that, for correctness, the scheme may restrict the secret key or the encrypted messages to be "small" values in a larger group over which the homomorphism holds.

We can now build UPKE as follows. We start with the circular-secure and leakage-resilient scheme and implement the updating mechanism by simply

[7]Here we compared to PKE based on the same assumption (DDH, LWE, etc.) to make this an "apple-to-apple" comparison. However, there are HIBEs [14,15] which are relatively efficient, but rely on pairings and use relatively strong assumptions.

[8]This is true for the DDH-based scheme of [17] since circular security requires encrypting in the exponent and decryption involves solving discrete log; therefore the encrypted values must be small. This is also true for the LWE-based scheme where the secret key must be small for correctness.

[9]Which is why we present the schemes separately, and the abstraction we give below is mainly for the intuition.

encrypting a random (small) offset δ and updating the public key appropriately using the key-homomorphic property. The receiver decrypts δ and updates their secret key s to $s' = s + \delta$. Note that the original key s should still have entropy when conditioned on the updated key s', so that we can use the leakage resilience of our scheme.

Reduction Idea. To get the intuition for our security proof, we first present the simplest special case where the challenge ciphertext for UPKE is requested in the very first time period, and there is a single honest update after the challenge, followed by the reveal of the resulting secret key s' to the attacker. As we will see, in this case, we will not even need to use the key-homomorphic property, but it is easy to see how key-homomorphic property will be needed for the general case. In our reduction, we will now need to utilize our UPKE attacker $\mathcal{A}$ for this simplest case, to build our CS+LR-attacker $\mathcal{A}'$.

$\mathcal{A}'$ will start with the challenge public key pk and will forward it to $\mathcal{A}$.

$\mathcal{A}$ will select two message m_0 and m_1 and give them to $\mathcal{A}'$.

$\mathcal{A}'$ will use these messages as its own challenge and will choose a probabilistic leakage function $s' = s + \delta$ for a random (unknown!) offset δ, where s is the original secret key of the CS+LR scheme.

Upon receiving this challenge ciphertext c^* and the value s' from its challenger, $\mathcal{A}'$ can simply forward c^* to $\mathcal{A}$, and also declare s' as the final value of the secret key after the update.

However, $\mathcal{A}'$ also needs to properly simulate the update ciphertext e which was supposed to encrypt the (unknown) value δ.

This is where $\mathcal{A}'$ will use the encryption e' of the secret key s, and the message-homomorphic property of the encryption scheme, to produce an encryption e of $\delta = s' - s$.

This completes the special case of the reduction. For the general case, where several (untrusted) updates could happen before the challenge is issued, we will also need to use the key-homomorphic property: both for

(a) converting the challenge ciphertext c^* in period 1 into the ciphertext encrypting the same message during the challenge period; as well as for

(b) converting the original encryption e' of s in period 1 into correctly distributed encryption e of δ during the exposure period.

While the high-level idea above will work for both of our DDH/LWE instantiations, in both cases we need to overcome certain challenges due to the need to correctly simulate various distributions in the above-sketched reduction.

Instantiating from DDH. We show that the BHHO cryptosystem [17] constructed from the DDH assumption satisfies the properties we need. In that cryptosystem the secret key is $\boldsymbol{s} \in \mathbb{Z}_p^\ell$. Circular security holds when each component of the secret key is encrypted in the exponent, and decryption recovers the secret key by taking the discrete log. For this reason, the BHHO scheme needs to use a "short" $\boldsymbol{s} \in \{0,1\}^\ell \subseteq \mathbb{Z}_p^\ell$. In our setting, we will set the initial key to a uniformly random $\boldsymbol{s}_0 \in \{0,1\}^\ell$. Each update will choose some random offset $\boldsymbol{\delta}_i \in \{0,1\}^\ell$ and will encrypt the value $\boldsymbol{\delta}_i$ in the exponent; the updated key will be $\boldsymbol{s}_{i+1} = \boldsymbol{s}_i + \boldsymbol{\delta}_i$ where the addition is performed over $\mathbb{Z}_p$. This scheme was shown to be circular secure [17] and leakage-resilient [46]; we show that the two security properties also hold simultaneously. It is also easy to see that the scheme is key and message homomorphic. When we use this scheme as a UPKE, we rely on the fact that when $\boldsymbol{\delta}, \boldsymbol{s} \in \{0,1\}^\ell$ are both chosen randomly then giving the sum $\boldsymbol{\delta} + \boldsymbol{s}$ (with addition over $\mathbb{Z}_p$) only reduces the entropy of $\boldsymbol{s}$ by $\ell \cdot \log(4/3) \leq \ell/2$ bits.

Instantiating from LWE. We show that the dual-Regev cryptosystem [48,33] constructed from the LWE assumption also satisfies the properties we need. The proof of circular security and leakage resilience are analogous to those of BHHO. One subtle issue that, while the dual-Regev scheme is key-homomorphic, when we update the key, we no longer get the correct ciphertext distribution – in particular, the "error term" distribution is perturbed. To fix this, we need to resort to the 'noise flooding/smudging" technique, where we add some super-polynomial noise to the ciphertext to hide small polynomial differences in the error term.

Follow Up Work. Following this work, the work of [28] defined an extension of UPKE called *fast-forwardable* UPKE (FF-UPKE). FF-UPKE addresses the problem that the UPKE receiver might be offline or otherwise miss many update ciphertexts, resulting in a situation where its current secret key sk_i is considerably behind the current public key pk_j: $i \ll j$. Of course, such "stale" receiver can still get to the current key sk_j, by downloading $\Delta = (j - i)$ key update messages, and performing Δ sequential secret key updates to "catch up". The goal of FF-UPKE is to achieve such "catching up" much faster: say, but only downloading a sub-linear (and, ideally, logarithmic) in Δ number of update ciphertexts (and, similarly, doing the sub-linear amount of work). [28] also built a generic FF-UPKE from any UPKE which they call *update-homomorphic*. Interestingly, minor modifications of our UPKE constructions turn out to be update-homomorphic. In contrast, the ROM-based UPKE [38] does not appear to be update-homomorphic and does not suffice for building FF-UPKE. Thus, as an unexpected application, the homomorphic properties of our construction, — which were needed to argue security of our scheme, but were not needed for the functionality, — turned out to be useful in a setting where random oracle does not appear to help.

1.2 Additional Theoretical Contributions

In the full version of our paper, we also consider two natural strengthenings of the basic chosen-plaintext attack (CPA) security of UPKE. First, we define the chosen-ciphertext attack (CCA) variant, where the attacker also has oracle access to the decryption oracle. Second, we consider extending the capability of untrusted senders in the CPA/CCA security game to produce *arbitrary* tuple of update ciphertext e and the corresponding new public key pk', rather than limiting their ability to selecting bad randomness r, and then *honestly* using r to produce the tuple (e, pk').

As initial feasibility results, for both variants, we show a generic way — using appropriate [27] notion of non-interactive zero-knowledge (NIZK) proofs — to extend the basic CPA notion of UPKE to meet the corresponding stronger requirement. Using the existing feasibility of such NIZK proofs from DDH/LWE, we get these stronger forms of UPKE can be met, under the corresponding assumption, in the standard model. Unlike our CPA constructions described above, here we do not give an *efficient* instantiation of the resulting schemes from DDH/LWE, leaving those to future work.

1.3 Related Work

Hierarchical Identity-Based Encryption (HIBE). As mentioned, Canetti et al. [24] also showed how to build FS-PKE (and therefore also UPKE) from any Hierarchical Identity-Based Encryption (HIBE) [35,34,14,25,15,31,30,22].

By plugging in prior constructions of HIBE from DDH/CDH [31,30,22], we would get an alternate construction of UPKE from DDH/CDH in the standard model. However, this construction is mainly of theoretical interest and is hugely impractical. In particular, it relies on complex garbled circuits that perform public-key operations. In more detail, if κ is the security parameter, the construction relies on a chain of $O(\kappa)$ garbled circuits, each of which outputs $O(\kappa)$ special ciphertexts (encrypted labels for next level garbled circuit), where each ciphertext consists of at least $O(\kappa)$ group elements; the fact that this is all computed inside a garbled circuit then adds at least another $O(\kappa)$ overhead on top. Lastly, going from HIBE to FS-PKE/UPKE adds another $O(\kappa)$ overhead, for a total complexity of at least $O(\kappa^5)$. So the complexity is at least $O(\kappa^3)$ worse than our scheme, even without getting into huge concrete overheads.

By plugging in prior constructions of HIBE from LWE [25,2], we would get an alternate construction of UPKE from LWE in the standard model. The resulting schemes could potentially be piratically efficient. However, our construction is still significantly simpler and more efficient for several reasons: (1) We do not rely on lattice trapdoors or GPV style pre-image sampling [33], which makes our scheme both conceptually simpler and practically more efficient. (2) Our secret key is a single lattice vector rather than an entire lattice basis. This makes our secret keys roughly an $O(\kappa)$ factor shorter. (3) We avoid the additional $O(\kappa)$ factor overhead in the transformation from HIBE to FS-PKE/UPKE.

Forward-secure Signatures. Forward-Secure Signatures [6] are similar to FS-PKE, in that compromising the current signing key should not enable forgery of messages for previous periods. In particular, the tree-based FS-signature scheme of Bellare and Miner [11] was the inspiration for the HIBE-based FS-PKE of [24]. The above work was later extended by Malkin *et al.* [43]. Forward-secure signatures were also studied in the random oracle setting [1,36,41].

Other Key Evolving Encryption Schemes. The works of Jaeger and Stepanovs [37] and Poettering and Rössler [47] proposed two related notions of *key-updatable* PKE scheme, which provide an even stronger form of key-evolution than FS-PKE.

In these schemes, key updates can be labeled by arbitrary, possibly adversarially chosen, strings. Unsurprisingly, the schemes in these works were also built from HIBE.

Circular and KDM Secure Encryption Schemes. Circular secure schemes allow the attacker to see encryptions of the secret key of the scheme. A natural extension of this notion studies a cycle of $(\mathsf{sk}_i, \mathsf{pk}_i)$ pairs for $i = 1, \ldots, n$ where we encrypt sk_i under $\mathsf{pk}_{i \bmod n+1}$. This was defined as *key-dependent message security (KDM)* by Black *et al.* [13] and as *circular security* by Camenisch and Lysyanskaya [23]. The first cryptosystem in the standard model with a proof of KDM-security under a standard assumption was given by Boneh *et al.* [17]. Subsequently, constructions from the learning with errors (LWE) [8] and quadratic residuosity [20] assumptions were proposed. Construction for identity-based KDM-secure encryption [4] was also proposed. While the construction of Boneh *et al.* [17] was for affine functions, subsequent "KDM amplifications" transforms extended the class of functions significantly [10,21,44,7].

Leakage-Resilient Encryption Schemes. Most of the security models do not capture possible *side-channel attacks*. These attacks are designed to exploit unintended leakage that often stems from the physical environment. Akavia *et al.* [3] proposed a realistic framework that aimed to capture information about the leakage. Subsequent work by Naor and Segev [46] analyzed the resilience of public key cryptosystems to leakage. An important result was that they showed the (even slightly optimized version of the) BHHO scheme [17] was resilient to $|\mathsf{sk}|(1 - o(1))$ bits of leakage. Subsequent work [26] showed the leakage resilience of both the BHHO scheme and the dual Regev encryption scheme [48,33] in the auxiliary input model. Brakerski *et al.* [22] studied both the leakage resilience and circular security of anonymous IBE. We point to the survey of leakage resilient cryptography by Kalai and Reyzin [39] for additional work in this domain.

Different "Updatable" Encryption. With an unfortunate naming collision, there has been a different kind of "updatable encryption schemes" considered in the literature [18,32,42,40,19,16]. These are *symmetric-key* encryption schemes that aim to accomplish key rotation in the cloud, specifically moving the ciphertexts under the old key to the new key. In particular, these schemes produce multiple encryptions of the same message under different keys and aim to produce update tokens that allow the update of old ciphertexts, without leaking the message content. In contrast, updatable schemes in this paper are *public-key*, encrypt different messages, and aim to achieve forward security. Thus, the notions are very different despite the partial naming collision.

2 Preliminaries

Notation. For a distribution X, we use $x \leftarrow\!\!\$\; X$ to denote that x is a random sample drawn from the distribution X. For a set S we use $x \leftarrow\!\!\$\; S$ to denote that

x is chosen uniformly at random from the set S. We denote by U_d the uniform distribution over $\{0,1\}^d$.

Information-Theoretic Notions. The *prediction probability* is

$$\mathsf{Pred}(X) := \max_x \mathsf{P}\left[X = x\right].$$

We can also denote

$$\mathsf{Pred}(X|y) := \max_x \mathsf{P}\left[X = x|Y = y\right].$$

We define the conditional versions as

$$\mathsf{Pred}(X|Y) := \mathbf{E}_{y \leftarrow Y}\left[\mathsf{Pred}(X|y)\right].$$

The *(average-case) conditional min-entropy* is $\mathrm{H}_\infty(X|Y) = -\log(\mathsf{Pred}(X|Y))$. The *statistical distance* of X and Y is $\mathsf{SD}(X,Y) = \frac{1}{2}\sum_x |\mathsf{P}\left[X = x\right] - \mathsf{P}\left[Y = y\right]|$.

Theorem 1 (Leftover Hash Lemma). *Fix $\varepsilon > 0$. Let X be a random variable on $\{0,1\}^n$ with conditional min-entropy $\mathrm{H}_\infty(X|E) \geq k$. Let $\mathcal{H} = \{\mathcal{H}_n\}_{n\in\mathbb{N}}$ where $\mathcal{H}_n = \{h_s\}_{s\in\{0,1\}^d}$ for all n, be a universal hash family with output length $m \leq k - 2\log(1/\varepsilon)$. Then, $(h_{U_d}(X), U_d, E) \approx_\varepsilon (U_m, U_d, E)$*

Lemma 1 (Smudging Lemma [9]). *Let $B_1 = B_1(\kappa)$ and $B_2 = B_2(\kappa)$ be positive integers and let $e_1 \in [-B_1, B_1]$ be a fixed integer. Let $e_2 \leftarrow_\$ [-B_2, B_2]$ be chosen uniformly at random. Then the distribution of e_2 is statistically indistinguishable from $e_1 + e_2$ as long as $B_1/B_2 = \mathrm{negl}(\kappa)$.*

Definition 1 (The Decisional Diffie Hellman Assumption (DDH)). *Let $\mathcal{G}$ be a probabilistic polynomial-time "group generator" that, given as a parameter 1^κ where κ is the security parameter, outputs the description of a group $\mathbb{G}$ that has prime order $p = p(\kappa)$. The decisional Diffie Hellman (DDH) assumption for $\mathcal{G}$ says that the following two ensembles are computationally indistinguishable:*

$$\{(g_1, g_2, g_1^r, g_2^r) : g_i \leftarrow \mathbb{G}, r \leftarrow \mathbb{Z}_p\} \approx_c \{(g_1, g_2, g_1^{r_1}, g_2^{r_2}) : g_i \leftarrow \mathbb{G}, r_i \leftarrow \mathbb{Z}_p\}$$

A lemma of Naor and Reingold [45] generalizes the above assumption for $m > 2$ generators.

Lemma 2 ([45]). *Under the DDH assumption on $\mathcal{G}$,*

$$\{(g_1, \ldots, g_m, g_1^r, \ldots, g_m^r) : g_i \leftarrow \mathbb{G}, r \leftarrow \mathbb{Z}_p\} \approx_c \{(g_1, \ldots, g_m, g_1^{r_1}, \ldots, g_m^{r_m}) : g_i \leftarrow \mathbb{G}, r_i \leftarrow \mathbb{Z}_p\}$$

Definition 2 (Learning with Errors Assumption (LWE)). *Consider integers n, m, q and a probability distribution χ on $\mathbb{Z}_q$, typically taken to be a normal distribution that has been discretized. Then, the LWE assumption states that the following two ensembles are computationally indistinguishable:*

$$\{\boldsymbol{A}, \boldsymbol{A}^T\boldsymbol{x}+\boldsymbol{e} : \boldsymbol{A} \leftarrow_\$ \mathbb{Z}_q^{n\times m}, \boldsymbol{x} \leftarrow_\$ \mathbb{Z}_q^n, \boldsymbol{e} \leftarrow_\$ \chi^m\} \approx_c \{\boldsymbol{A}, \boldsymbol{v} : \boldsymbol{A} \leftarrow_\$ \mathbb{Z}_q^{n\times m}, \boldsymbol{v} \leftarrow_\$ \mathbb{Z}_q^m\}$$

3 Updatable Public Key Encryption (UPKE)

Jost *et al.* [38] introduced the notion of an Updatable Public Key Encryption (UPKE). This definition was later modified by the work of Alwen *et al.* [5]. Below, we present our variant of the UPKE.

Definition 3. *An updatable public key encryption (UPKE) scheme is a set of five polynomial-time algorithms* $\mathsf{UPKE} = (\mathsf{U\text{-}PKEG}, \mathsf{U\text{-}Enc}, \mathsf{U\text{-}Dec}, \mathsf{Upd\text{-}Pk}, \mathsf{Upd\text{-}Sk})$ *with the following syntax:*

- ***Key generation:*** $\mathsf{U\text{-}PKEG}$ *takes as parameter* 1^κ *where* κ *is the security parameter and outputs a fresh secret key* sk_0 *and a fresh initial public key* pk_0.
- ***Encryption:*** $\mathsf{U\text{-}Enc}$ *receives a public key* pk *and a message* m *to produce a ciphertext* c.
- ***Decryption:*** $\mathsf{U\text{-}Dec}$ *receives a secret key* sk *and a ciphertext* c *to produce message* m.
- ***Update Public Key:*** $\mathsf{Upd\text{-}Pk}$ *receives a public key* pk *to produce an update ciphertext* up *and a new public key* pk'.
- ***Update Secret Key:*** $\mathsf{Upd\text{-}Sk}$ *receives an update ciphertext* up *and secret key* sk *to produce a new secret key* sk'.

Correctness. Let (sk_0, pk_0) be the output of $\mathsf{U\text{-}PKEG}$. For any sequence of randomness $\{r_i\}_{i=1}^q$, define the sequence of public keys and secret keys $\{(\mathsf{pk}_i, \mathsf{sk}_i)\}_{i=1}^q$ as follows: $(\mathsf{up}_i, \mathsf{pk}_i) \leftarrow \mathsf{Upd\text{-}Pk}(\mathsf{pk}_{i-1}; r_i)$, $\mathsf{sk}_i \leftarrow \mathsf{Upd\text{-}Sk}(\mathsf{sk}_{i-1}, \mathsf{up}_i)$. Then, UPKE is correct if for any message m and for any $j \in [q]$,

$$\mathsf{P}\left[\mathsf{U\text{-}Dec}(\mathsf{sk}_j, \mathsf{U\text{-}Enc}(\mathsf{pk}_j, m)) = m\right] = 1 \ .$$

3.1 IND-CR-CPA Security of UPKE

In this section, we define the security game. We will called this the IND-CR-CPA Security which is meant to capture INDistinguishibility under Chosen Randomness Chosen Plaintext Attack. Largely similar to the CPA security game, this also additionally allows the adversary to choose the randomness used to update the keys which is modeled by the following oracle access:

– $\mathcal{O}_{upd}(\cdot)$: $\mathcal{A}$ provides its choice of randomness r_i. The Challenger increments the time to $i+1$. It then performs the following actions:

$$(\mathsf{up}_{i+1}, \mathsf{pk}_{i+1}) \leftarrow \mathsf{Upd\text{-}Pk}(\mathsf{pk}_i; r_i);\ \mathsf{sk}_{i+1} \leftarrow \mathsf{Upd\text{-}Sk}(\mathsf{sk}_i, \mathsf{up}_{i+1}) \ .$$

For any adversary $\mathcal{A}$ with running time t we consider the IND-CR-CPA security game:

– Sample $(\mathsf{sk}_0, \mathsf{pk}_0) \leftarrow \mathsf{U\text{-}PKEG}(1^\kappa)$, $b \leftarrow_\$ \{0,1\}$.
– $(m_0^*, m_1^*, state) \leftarrow_\$ \mathcal{A}^{\mathcal{O}_{upd}(\cdot)}(\mathsf{pk}_0)$

– Compute $c^* \leftarrow\$ \mathsf{U\text{-}Enc}(\mathsf{pk}_{q'}, m_b^*)$ where q' is the current time period.

– $state \leftarrow\$ \mathcal{A}^{\mathcal{O}_{upd}(\cdot)}(c^*, state)$

– Choose uniformly random r^* and then compute

$$(\mathsf{up}^*, \mathsf{pk}^*) \leftarrow \mathsf{Upd\text{-}Pk}(\mathsf{pk}_q; r^*);\ \mathsf{sk}^* \leftarrow \mathsf{Upd\text{-}Sk}(\mathsf{sk}_q, \mathsf{up}^*)\ .$$

where q is the current time period.

– $b' \leftarrow\$ \mathcal{A}(\mathsf{pk}^*, \mathsf{sk}^*, \mathsf{up}^*, state)$.

– $\mathcal{A}$ wins the game if $b = b'$. The advantage of $\mathcal{A}$ in winning the above game is denoted by $\mathrm{Adv}_{\mathrm{crcpa}}^{\mathsf{UPKE}}(\mathcal{A}) = |\mathsf{P}[b = b'] - \frac{1}{2}|$.

Definition 4. *An updatable public-key encryption scheme* UPKE *is IND-CR-CPA -secure if for all PPT attackers $\mathcal{A}$, its advantage $\mathrm{Adv}_{\mathrm{crcpa}}^{\mathsf{UPKE}}(\mathcal{A})$ is negligible.*

Remark 1 (Comparison of the Security Models.). The work of Jost *et al.* [38] defined a notion which had an update procedure not specific to any public key. This was designed to support multiple instances, i.e. multiple key pairs, and where the offset generated by the public update could be applied to many public keys. While we consider the simpler setting of only one instance, which is also reflected in our syntax, we believe that our constructions trivially satisfy the stronger security model proposed by [38]. Our model also allows for $q \neq q'$, i.e., for the adversary to issue a challenge in one time period and corrupt in another time period. However, without loss of generality, we give the attacker the final secret key sk^* immediately following the honest post-challenge key update (at period q'), as this gives the most amount of information to the attacker.

Our definition is a generalization of the model proposed by Alwen *et al.* [5]: their notion forced an update of the keys after every encryption query, while ours separates the two processes for more flexibility.

4 Key-Dependent-Message-Secure Encryption Scheme

Let us recall the definition of a public-key encryption scheme.

Definition 5. *An encryption scheme is a set of three polynomial-time algorithms $\mathcal{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ with the following syntax:*

- ***Key generation:*** Gen *receives 1^κ where κ is the security parameter and outputs a fresh secret* sk *and outputs a fresh public key* pk*.*
- ***Encryption:*** Enc *receives a public key* pk *and a message m to produce a ciphertext c.*
- ***Decryption:*** Dec *receives a secret key* sk *and a ciphertext c to produce message m.*

Correctness. The correctness of an encryption scheme is such that $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\kappa)$, $\forall m \in \mathcal{M}$,

$$\mathsf{P}[\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, m)) = m] = 1$$

CS+LR Security. For any PPT adversary $\mathcal{A}$ we consider the following security game:

- Sample $(\mathsf{sk}, \mathsf{pk}) \leftarrow_{\$} \mathsf{Gen}(1^{\kappa}), b \leftarrow_{\$} \{0,1\}$.
- $L, f, m_0, m_1 \leftarrow_{\$} \mathcal{A}(\mathsf{pk})$ where L is the leakage function chosen by $\mathcal{A}$, m_0, m_1 are the challenge messages, and f is the function of the secret key that $\mathcal{A}$ wants to receive as encryption. L defines the leakage resilience and f defines the KDM security.
- Compute $C \leftarrow_{\$} \mathsf{Enc}(\mathsf{pk}, m_b)$, $C' \leftarrow_{\$} \mathsf{Enc}(\mathsf{pk}, f(\mathsf{sk}))$[10].
- $b' \leftarrow_{\$} \mathcal{A}(c_0, c_1, L(\mathsf{sk}))$.
- $\mathcal{A}$ wins the game if $b = b'$. The advantage of $\mathcal{A}$ in winning the above game is denoted by $\mathrm{Adv}^{\mathcal{E}}_{\mathrm{KDM}}(\mathcal{A}) = |\mathsf{P}[b = b'] - \frac{1}{2}|$.

Definition 6. *A public-key encryption scheme $\mathcal{E}$ is λ-CS+LR-secure if for all PPT attackers $\mathcal{A}$, and leakage functions L such that $\mathrm{H}_{\infty}(\mathsf{sk}|L(\mathsf{sk})) \geq |\mathsf{sk}| - \lambda$, its advantage $\mathrm{Adv}^{\mathcal{E}}_{\mathrm{cs+lr}}(\mathcal{A})$ is negligible.*

5 DDH Based Construction

This section presents construction from the DDH Assumption. We begin by presenting a slightly modified version of the PKE Scheme proposed by Boneh *et al.* [17] in section 5.1. This scheme was shown to be independently circular secure and leakage resilient. We also show that the scheme is CS+LR secure in section 5.2. We then present our construction of a UPKE scheme (section 5.3), extended from the PKE scheme. We finally prove that the UPKE scheme is IND-CPA secure in section 5.4.

5.1 The BHHO Cryptosystem

In this section, we present a modified version of the original BHHO Cryptosystem. This is presented as Construction 1.

Correctness. Let $m \in \mathbb{G}$. $\mathsf{Enc}(\mathsf{pk}, m) = (f_1 = g_1^r, \ldots, f_\ell = g_\ell^r, c = h^r \cdot m)$. Now, $\mathsf{Dec}(\mathsf{sk}, f_1, \ldots, f_\ell, c)$ outputs: $c \cdot (\prod_{i=1}^{\ell} f_i^{s_i})^{-1} = h^r \cdot m(\prod_{i=1}^{\ell} f_i^{s_i})^{-1} = (\prod_{i=1}^{\ell} g_i^{s_i})^r \cdot m \cdot (\prod_{i=1}^{\ell} (g_i^r)^{s_i})^{-1} = m$.

5.2 CS+LR Security of BHHO Cryptosystem

In this section, we provide proof of the combined circular security and leakage resilience of the BHHO Cryptosystem. Formally, we will prove the following theorem:

Theorem 2. *Under the DDH Assumption, Construction 1 is λ-CS+LR secure for leakage $\lambda = \ell - 2\log p - \omega(\log \kappa)$.*

[10]In our security proofs, the function f will be applied to each bit of the secret key.

Protocol BHHO Cryptosystem

```
$\mathsf{Gen}(1^\kappa)$
    Sample $\boldsymbol{s} = (s_1, \dots, s_\ell) \leftarrow\$ \{0,1\}$ and
    $g_1, \dots, g_\ell \leftarrow\$ \mathbb{G}$.
    Compute $h = \prod_{i=1}^{\ell} g_i^{s_i}$.
    return $\mathsf{sk} = \boldsymbol{s} \in \mathbb{Z}_p^\ell,\ \mathsf{pk} = (g_1, \dots, g_\ell, h) \in \mathbb{G}^{\ell+1}$ .

$\mathsf{Enc}(\mathsf{pk}, m \in \mathbb{G})$
    Parse $\mathsf{pk} = (g_1, \dots, g_\ell, h)$
    Sample $r \leftarrow\$ \mathbb{Z}_p$
    for $i = 1, \dots, \ell$ do
        Compute $f_i = g_i^r$
    return $C = (f_1, \dots, f_\ell, c = h^r \cdot m) \in \mathbb{G}^{\ell+1}$

$\mathsf{Dec}(\mathsf{sk}, C)$
    Parse $C = (f_1, \dots, f_\ell, c = h^r \cdot m)$ and $\mathsf{sk} = \boldsymbol{s} = (s_1, \dots, s_\ell) \in \mathbb{Z}_p^\ell$
    Compute $m' = c \cdot \left(\prod_{i=1}^{\ell} f_i^{s_i}\right)^{-1}$
    return $m'$
```

Construction 1. A modified version of the BHHO Cryptosystem where the bits of the secret key are not encoded as group elements. Let κ be the the security parameter. Let $\mathcal{G}$ be a probabilistic polynomial-time "group generator" that takes as input 1^κ and outputs the description of a group $\mathbb{G}$ with prime order $p = p(\kappa)$ and g is a fixed generator of $\mathbb{G}$.

However, before we can prove the theorem, we will prove that there exists an algorithm $\mathsf{Enc}'(\mathsf{pk}, i)$ such that $(\mathsf{pk}, \mathsf{Enc}(\mathsf{pk}, g^{s_i}), \boldsymbol{s}) \approx_c (\mathsf{pk}, \mathsf{Enc}'(\mathsf{pk}, i), \boldsymbol{s})$. Consider the following definition of Enc':

$$\mathsf{Enc}'(\mathsf{pk}, i) = (f_1 = g_1^r, \dots, f_{i-1} = g_{i-1}^r, f_i = g_i^r/g, f_{i+1} = g_{i+1}^r, \dots, f_\ell = g_\ell^r, h^r)$$

We will first show that this ciphertext decrypts correctly to g^{s_i}.

$$\mathsf{Dec}(\boldsymbol{s}, f_1, \dots, f_\ell, c = h^r) = h^r \cdot \left(\prod_{i=1}^{\ell} f_i^{s_i}\right)^{-1} = h^r \left(\prod_{i=1}^{\ell} {g_i}^{s_i}\right)^{-r} g^{s_i} = h^r h^{-r} g^{s_i} = g^{s_i}$$

Lemma 3. *Under the DDH Assumption,* $(\mathsf{pk}, \mathsf{Enc}(\mathsf{pk}, g^{s_i}), \boldsymbol{s}) \approx_c (\mathsf{pk}, \mathsf{Enc}'(\mathsf{pk}, i), \boldsymbol{s})$ *where* $(\mathsf{pk}, \boldsymbol{s}) \leftarrow\$ \mathsf{Gen}(1^\kappa)$

Proof. We will prove the lemma through a sequence of hybrids, outlined in Table 2.

Hybrid D_0. This is when Enc is used to encrypt g^{s_i}. It corresponds to the distribution:

$$(\mathsf{pk}, g_1^r, \dots, g_\ell^r, h^r \cdot g^{s_i}, \boldsymbol{s} : r \leftarrow\$ \mathbb{Z}_p)$$

Hybrid D_1. This is same as Hybrid D_0 where we replace h^r by the steps of the decryption algorithm. It corresponds to the distribution

$$\left(\mathsf{pk}, f_1 = g_1^r, \dots, f_\ell = g_\ell^r, \prod_{j=1}^{\ell} f_j^{s_j} \cdot g^{s_i}, \boldsymbol{s} : r \leftarrow\$ \mathbb{Z}_p\right)$$

The distributions D_0 and D_1 are identical for the same value of $r \leftarrow\$ \mathbb{Z}_p$. Therefore, there is no distinguishing advantage for any adversary $\mathcal{A}$.

Table 2. Proof Outline for Lemma 3

Hybrid	Hybrid Definition	Security
D_0	Enc is used to encrypt g^{s_i}	Identical
D_1	D_0 except $h^r \cdot g^{s_i}$ replaced with $\prod_{j=1}^{\ell} f_j^{s_j} \cdot g^{s_i}$	DDH
D_2	D_1 except each $f_j \leftarrow\!\!\$\ \mathbb{G}$	Identical
D_3	D_2 except f_i is replaced by f_i/g where $f_i \leftarrow\!\!\$\ \mathbb{G}$	DDH
D_4	D_3 except $f_j = g_j^r$ where $r \leftarrow\!\!\$\ \mathbb{Z}_p$	Identical
D_5	Enc' is used to encrypt g^{s_i}	

Hybrid D_2. In this case, we sample each $f_i \leftarrow\!\!\$\ \mathbb{G}$. This corresponds to the distribution:

$$\left(\mathsf{pk}, f_1, \ldots, f_\ell, \prod_{j=1}^{\ell} f_j^{s_j} \cdot g^{s_i}, \boldsymbol{s} : f_1, \ldots, f_\ell \leftarrow\!\!\$\ \mathbb{G}\right)$$

Claim. If DDH (as defined in Lemma 2) is hard for $\mathbb{G}$, then for every PPT $\mathcal{A}$, the advantage in distinguishing Hybrids D_1 and D_2 is negligible.

Proof. We will use an adversary $\mathcal{A}$ capable of distinguishing between the two distributions to create an adversary $\mathcal{B}$ that can win against the DDH Game. After receiving input from the challenger $(g_1, \ldots, g_\ell, f_1, \ldots, f_\ell)$, $\mathcal{B}$ generates $(\mathsf{pk}, \mathsf{sk} = \boldsymbol{s})$ and returns to $\mathcal{A}$: $(f_1, \ldots, f_\ell, \prod_{j=1}^{\ell} f_j^{s_j} \cdot g^{s_i}, \boldsymbol{s})$. It is easy to see that $\mathcal{B}$ perfectly simulates one of the hybrids based on the input it receives. This concludes the proof that $\mathcal{A}$ has negligible advantage in distinguishing the two hybrids. □

Hybrid D_3. The same distribution as Hybrid 2, except that f_i is replaced by f_i/g.

$$\left(\mathsf{pk}, f_1, \ldots, f_{i-1}, f_i/g, f_{i+1}, \ldots, f_\ell, \prod_{j=1}^{\ell} f_j^{s_j} \cdot g^{s_i}, \boldsymbol{s} : f_1, \ldots, f_\ell \leftarrow\!\!\$\ \mathbb{G}\right)$$

We know that for fixed g, f_i/g is indistinguishable from f_i where $f_i \leftarrow\!\!\$\ \mathbb{G}$. Therefore, the distributions are identical and $\mathcal{A}$ has no advantage in distinguishing the two distributions.

Hybrid D_4. This is corresponding to the distribution where $f_j = g_j^r$ where $r \leftarrow\!\!\$\ \mathbb{Z}_p$.

$$\left(\mathsf{pk}, g_1^r, \ldots, g_{i-1}^r, g_i^r/g, g_{i+1}^r, \ldots, g_\ell^r, \prod_{j=1}^{\ell} f_j^{s_j} \cdot g^{s_i}, \boldsymbol{s} : r \leftarrow\!\!\$\ \mathbb{Z}_p\right)$$

Claim. If DDH (as defined in Lemma 2) is hard for $\mathbb{G}$, then for every PPT $\mathcal{A}$, the advantage in distinguishing Hybrids D_3 and D_4 is negligible.

The proof of this claim is similar to the proof of the earlier claim.

Table 3. Proof Outline for Theorem 2

Hybrid	Hybrid Definition	Security
D_0	The Original CS+LR Security Game, Enc is used	
D_1	D_0 except Enc' is used	Corollary 1
D_2	D_1 except except $h^r \cdot m_b$ replaced with $\prod_{j=1}^{\ell} f_j^{s_j} \cdot m_b$	Identical
D_3	D_2 except each f_i is replaced by $f_i \leftarrow\$ \mathbb{G}$	DDH
D_4	D_3 except $\prod_{j=1}^{\ell} f_j^{s_j} \cdot m_b$ replaced by $U \leftarrow\$ \mathbb{G}$	Leftover Hash Lemma

Hybrid 5. This is corresponding to the distribution $(\mathsf{Enc}'(\mathsf{pk}, i), \boldsymbol{s})$, $f_i = g_i^r/g$ where $r \leftarrow\$ \mathbb{Z}_p$.

$$(\mathsf{pk}, f_1 = g_1^r, \ldots, f_{i-1} = g_{i-1}^r, f_i = g_i^r/g, f_{i+1} = g_{i+1}^r, \ldots, f_\ell = g_\ell^r, h^r, \boldsymbol{s} : r \leftarrow\$ \mathbb{Z}_p)$$

It is clear that the input distribution in Hybrids D_4 and D_5 are identical for the same r and $\mathcal{A}$ has no advantage in distinguishing the two distributions. This is because: $\prod_{j=1}^{\ell} f_j^{s_j} \cdot g^{s_i} = \prod_{j \neq i} g_j^{rs_j} \cdot g_i^{rs_i}/g^{s_i} \cdot g^{s_i} = (\prod_{j=1}^{\ell} g_j^{s_j})^r = h^r$.

Therefore, we have shown that $(\mathsf{Enc}(\mathsf{pk}, g^{s_i}), \boldsymbol{s}) \approx_c (\mathsf{Enc}'(\mathsf{pk}, i), \boldsymbol{s})$. □

Further, note that each s_i is independently chosen. Additionally, each encryption/fake-encryption chooses its own independent randomness r. Therefore, we can independently replace each $\mathsf{Enc}(\mathsf{pk}, g^{s_i})$ with $\mathsf{Enc}'(\mathsf{pk}, i)$, and the resulting encryption of secret key is computationally indistinguishable from the one computed by Enc'. This proof can be shown by a sequence of hybrids, replacing one encryption at a time. Therefore, as a corollary we get that:

Corollary 1. *Under the DDH Assumption,*

$$(\mathsf{pk}, \mathsf{Enc}(\mathsf{pk}, g^{s_1}), \ldots, \mathsf{Enc}(\mathsf{pk}, g^{s_\ell}), \boldsymbol{s}) \approx_c (\mathsf{pk}, \mathsf{Enc}'(\mathsf{pk}, 1), \ldots, \mathsf{Enc}(\mathsf{pk}, \ell), \boldsymbol{s})$$

With this corollary, we can prove the original theorem:

Theorem 2. *Under the DDH Assumption, Construction 1 is λ-CS+LR secure for leakage $\lambda = \ell - 2\log p - \omega(\log \kappa)$.*

Proof. We will prove the same through a sequence of hybrids, summarized in Table 3. Note that each of our hybrid distribution contains pk and $L(\mathsf{sk} = \boldsymbol{s})$ in its definition. We drop these terms from the definition for simplicity and merely focus on the two ciphertexts which undergo the bulk of the changes.

Hybrid D_0. The original CS+LR Game. In this hybrid, $\mathcal{A}$ receives the following distribution:

$$(C = (f_1 = g_1^r, \ldots, f_\ell = g_\ell^r, h^r \cdot m_b), C' = (\mathsf{Enc}(\mathsf{pk}, g^{s_1}), \ldots, \mathsf{Enc}(\mathsf{pk}, g^{s_\ell})) : r \leftarrow\$ \mathbb{Z}_p)$$

Hybrid D_1. The CS+LR Game but with C' consisting of the "fake encryption" algorithm. This corresponds to the distribution:

$$\big(C = (f_1 = g_1^r, \dots, f_\ell = g_\ell^r, h^r \cdot m_b), C' = (\mathsf{Enc}'(\mathsf{pk}, 1), \dots, \mathsf{Enc}'(\mathsf{pk}, \ell)) : r \leftarrow\!\!\$\, \mathbb{Z}_p\big)$$

In Corollary 1 we showed that the two distribution were indistinguishable even when conditioned on the secret key $\boldsymbol{s}$. However, in the definition of D_0, D_1, we only provide partial leakage $L(\boldsymbol{s})$, and hence $\mathcal{A}$ has negligible advantage in distinguishing the two distributions.

Hybrid D_2. It is similar to hybrid D_1, but with $h^r \cdot m_b$ replaced by $\prod_{j=1}^{\ell} f_j^{s_j} \cdot m_b$. This is the following distribution:

$$\left(C = \left(f_1 = g_1^r, \dots, f_\ell = g_\ell^r, \prod_{j=1}^{\ell} f_j^{s_j} \cdot m_b\right), C' : r \leftarrow\!\!\$\, \mathbb{Z}_p\right)$$

For the same r, the distributions from Hybrids D_2 and D_3 are identical. Therefore, $\mathcal{A}$ has no advantage in distinguishing the two hybrids.

Hybrid D_3. Similar to hybrid D_2, except each $f_i \leftarrow\!\!\$\, \mathbb{G}$. This is the following distribution:

$$\left(C = \left(f_1, \dots, f_\ell, \prod_{j=1}^{\ell} f_j^{s_j} \cdot m_b\right), C' : f_1, \dots, f_\ell \leftarrow\!\!\$\, \mathbb{G}\right)$$

Claim. If DDH is hard for $\mathbb{G}$, then for every PPT $\mathcal{A}$, the advantage in distinguishing hybrids D_2 and D_3 is negligible.

Proof. We will use an adversary $\mathcal{A}$ capable of distinguishing hybrids D_2 and D_3 to create $\mathcal{B}$ that can win against the DDH Game. $\mathcal{B}$ receives from the DDH Challenger: $(g_1, \dots, g_\ell, f_1, \dots, f_\ell)$. It chooses $\boldsymbol{s} \leftarrow\!\!\$\, \{0,1\}^\ell$ and sets $\mathsf{pk} = (g_1, \dots, g_\ell, h)$ where $h = \prod_{i=1}^{\ell} g_i^{s_i}$ and sets $\mathsf{sk} = \boldsymbol{s}$. It then sends to $\mathcal{A}$: $(\mathsf{pk}, L(\mathsf{sk} = \boldsymbol{s}), C = (f_1, \dots, f_\ell, \prod_{j=1}^{\ell} f_j^{s_j} \cdot m_b), C' = (\mathsf{Enc}'(\mathsf{pk}, 1), \dots, \mathsf{Enc}'(\mathsf{pk}, \ell)))$. It is easy to see that $\mathcal{B}$ perfectly simulates the distributions of hybrids D_2 and D_3 based on the input it receives. It merely forwards $\mathcal{A}$'s guess as its own. This concludes the proof that $\mathcal{A}$ has a negligible advantage in distinguishing hybrids D_2 and D_3. $\square$

Hybrid D_4. Replace $\prod_{j=1}^{\ell} f_j^{s_j}$ with a random value $U \leftarrow\!\!\$\, \mathbb{G}$. This gives the distribution:

$$\big(C = (f_1, \dots, f_\ell, U \cdot m_b), C' = (\mathsf{Enc}'(\mathsf{pk}, 1), \dots, \mathsf{Enc}'(\mathsf{pk}, \ell)) : U, f_1, \dots, f_\ell \leftarrow\!\!\$\, \mathbb{G}\big)$$

Claim. Hybrids D_3 and D_4 are statistically indistinguishable .

Proof. We can represent $f_i \leftarrow_\$ \mathbb{G}$ as g^{r_i} for random $r_i \leftarrow_\$ \mathbb{Z}_p$. Therefore, the term $\prod_{j=1}^{\ell} f_j^{s_j} = g^{\langle \boldsymbol{r}, \boldsymbol{s} \rangle}$ where $\boldsymbol{r} = (r_1, \ldots, r_\ell)$. Now, note that distinguishing hybrids D_3 and D_4 is at least as hard as distinguishing the following two ensembles:

$$(r_1, \ldots, r_\ell, \langle \boldsymbol{r}, \boldsymbol{s} \rangle, C' : r_1, \ldots, r_\ell \leftarrow_\$ \mathbb{Z}_p); (r_1, \ldots, r_\ell, u, C' : u, r_1, \ldots, r_\ell \leftarrow_\$ \mathbb{Z}_p)$$

If one could distinguish the second pair of distributions, then they can efficiently calculate the value of g^{r_i} and $g^{\langle \boldsymbol{r}, \boldsymbol{s} \rangle}$, thereby distinguishing the original pair of distributions.

We will now complete the proof by showing that the second pair of distributions are statistically indistinguishable. To this end, we will use LHL, as defined in Theorem 1. We have that

$$\mathrm{H}_\infty(\boldsymbol{s}|C', L(\boldsymbol{s}), \mathsf{pk}) = \mathrm{H}_\infty(\boldsymbol{s}|L(\boldsymbol{s}), \mathsf{pk}) \geq \mathrm{H}_\infty(\boldsymbol{s}|L(\boldsymbol{s})) - \log p \geq \ell - \lambda - \log p.$$

This is because C' is independent of the sk conditioned on pk, the value pk's component of h comes from a domain of size p, and L was a leakage function that satisfied $\mathrm{H}_\infty(\boldsymbol{s}|L(\boldsymbol{s})) = \ell - \lambda$. Now, consider, the hash function family $\mathcal{H}$ consisting of $h_{\boldsymbol{r}}(\boldsymbol{s}) = \langle \boldsymbol{r}, \boldsymbol{s} \rangle \mod p$. The output length is $m = \log p$. This is a universal hash family. To apply LHL we need, $m = k - 2\log(1/\varepsilon)$. Here $k = \ell - \lambda - \log p$. Therefore, $\log p = \ell - \lambda - \log p - 2\log(1/\varepsilon)$. Or if $\lambda \leq \ell - 2\log p - 2\log(1/\varepsilon)$, for some negligible ε then the latter two distributions are statistically indistinguishable. $\square$

It follows from the above claim that $\mathcal{A}$ has a negligible advantage in distinguishing hybrids D_3 and D_4. Further, in Hybrid 4, the message is masked by a random value and therefore $\mathcal{A}$ has no advantage in Hybrid D_4.

Combining the different hybrid arguments together, we get that any PPT algorithm $\mathcal{A}$ has a negligible advantage in the CS+LR security game. $\square$

5.3 UPKE Construction

In this section, we present our construction of an updatable public key encryption based on the BHHO Cryptosystem. This is presented in Construction 2.

Correctness. Informally, correctness requires that any message m encrypted by an updated public key decrypts with the help of the corresponding updated secret key to the same message m, always.

- Let $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{U\text{-}PKEG}(1^\kappa)$. Here, $\mathsf{sk} = \boldsymbol{s} = (s_1, \ldots, s_\ell) \leftarrow_\$ \{0,1\}^\ell$, and $\mathsf{sk} = (g_1, \ldots, g_\ell, \prod_{j=1}^{\ell} g_i^{s_i})$.
- Let r be the randomness used for the $\mathsf{Upd\text{-}Sk}$ procedure. Let $\boldsymbol{\delta} = (\delta_1, \ldots, \delta_\ell)$ be the first ℓ bits of r. We have $(\mathsf{pk}', \mathsf{up}) \leftarrow \mathsf{Upd\text{-}Pk}(\mathsf{pk})$. Here, $\mathsf{pk}' = (g_1, \ldots, g_\ell, h \cdot \prod_{j=1}^{\ell} g_i^{\delta_i})$. $\mathsf{up} = (g_1^{r_1}, \ldots, g_\ell^{r_1}, h^{r_1} \cdot g^{\delta_1}), \ldots, (g_1^{r_\ell}, \ldots, g_\ell^{r_\ell}, h^{r_\ell} \cdot g^{\delta_\ell})$.
- We will look at $\mathsf{Upd\text{-}Sk}$ now. It is easy to verify that $\mathsf{Upd\text{-}Sk}$ correctly decrypts each ciphertext in up to corresponding g^{δ_i}. This is either 1 when $\delta_i = 0$ or non-identity if $\delta = 1$. It then updates $\boldsymbol{s}' = \boldsymbol{s} + \boldsymbol{\delta}$. Interestingly, while $\boldsymbol{s}$ was initialized to be a bit string, each element grows slowly over $\mathbb{Z}_p$.

Protocol DDH-Based UPKE

```
U-PKEG($1^\kappa$)
    Sample $\boldsymbol{s} = (s_1, \dots, s_\ell) \leftarrow\$ \{0,1\}$ and
    $g_1, \dots, g_\ell \leftarrow\$ \mathbb{G}$.
    Compute $h = \prod_{i=1}^{\ell} g_i^{s_i}$.
    return $\mathsf{sk} = \boldsymbol{s} \in \mathbb{Z}_p^\ell,\ \mathsf{pk} = (g_1, \dots, g_\ell, h) \in \mathbb{G}^{\ell+1}$ .

U-Enc($\mathsf{pk}, m \in \mathbb{G}$)
    Parse $\mathsf{pk} = (g_1, \dots, g_\ell, h)$
    Sample $r \leftarrow\$ \mathbb{Z}_p$
    for $i = 1, \dots, \ell$ do
        Compute $f_i = g_i^r$
    return $C = (f_1, \dots, f_\ell, c = h^r \cdot m) \in \mathbb{G}^{\ell+1}$

U-Dec($\mathsf{sk}, C$)
    Parse $C = (f_1, \dots, f_\ell, c = h^r \cdot m)$ and $\mathsf{sk} = \boldsymbol{s} =$
    $(s_1, \dots, s_\ell) \in \mathbb{Z}_p^\ell$
    Compute $m' = c \cdot \left(\prod_{i=1}^{\ell} f_i^{s_i}\right)^{-1}$
    return $m'$

Upd-Pk($\mathsf{pk}$)
    Parse $\mathsf{pk} = (g_1, \dots, g_\ell, h)$
    Sample $\boldsymbol{\delta} = (\delta_1, \dots, \delta_m) \leftarrow\$ \{0,1\}^\ell$
    Compute $h' = h \cdot \left(\prod_{i=1}^{\ell} g_i^{\delta_i}\right)$
    Encrypt $\delta$ bit-by-bit, i.e., $\mathsf{up} =$
    $\left(\mathsf{U\text{-}Enc}(\mathsf{pk}, g^{\delta_1}), \dots, \mathsf{U\text{-}Enc}(\mathsf{pk}, g^{\delta_\ell})\right)$.
    return $(\mathsf{up}, \mathsf{pk}' = (g_1, \dots, g_\ell, h'))$

Upd-Sk($\mathsf{sk}, \mathsf{up}$)
    Parse $\mathsf{up} = (c_1, \dots, c_\ell)$
    for $i = 1, \dots, \ell$ do
        Compute $u_i = \mathsf{U\text{-}Dec}(\mathsf{sk}, c_i)$
        if $u_i = 1$ then
            Set $\delta_i = 0$
        else
            Set $\delta_i = 1$
    Compute $\boldsymbol{s}' = \boldsymbol{s} + \boldsymbol{\delta}$ where $\boldsymbol{\delta} = (\delta_1, \dots, \delta_\ell)$ and addition
    is element-by-element over $\mathbb{Z}_p$.
    return $\mathsf{sk}' = (\boldsymbol{s}')$
```

Construction 2. DDH Based Construction. Let κ be the the security parameter. Let $\mathcal{G}$ be a probabilistic polynomial-time "group generator" that takes as input 1^κ and outputs the description of a group $\mathbb{G}$ with prime order $p = p(\kappa)$ and g is a fixed generator of $\mathbb{G}$. Set $\ell = \lceil 5 \log p \rceil$.

- Consider $\mathsf{U\text{-}Enc}(\mathsf{pk}', m)$. The resulting ciphertext is $(g_1^u, \dots, g_\ell^u, h'^u \cdot m)$ for $u \leftarrow\$ \mathbb{Z}_p$.
- Consider $\mathsf{U\text{-}Dec}(\mathsf{sk}', g_1^u, \dots, g_\ell^u, h'^u \cdot m))$. The decryption algorithm returns:

$$
\begin{aligned}
h'^u \cdot m \cdot (\prod_{j=1}^{\ell} (g_j^u)^{s'_j})^{-1} &= (h \cdot \prod_{j=1}^{\ell} g_j^{\delta_j})^u \cdot m \cdot (\prod_{j=1}^{\ell} (g_j^u)^{s'_j})^{-1} \\
&= (\prod_{j=1}^{\ell} g_j^{s_j} \cdot \prod_{j=1}^{\ell} g_j^{\delta_j})^u \cdot m \cdot (\prod_{j=1}^{\ell} (g_j^u)^{s_j+\delta_j})^{-1} = m
\end{aligned}
$$

- The same can be extended to additional updates. The key point to note is that the algorithms do not need $\boldsymbol{s}$ to be a bit string and therefore can, and indeed will grow.

5.4 Security of the UPKE Construction

Theorem 3. *Under the DDH Assumption, Construction 2 is IND-CR-CPA secure UPKE.*

Proof. We proved in Theorem 2 that Construction 1 is CS+LR secure with $\lambda = \ell - 2\log p - \omega(\log \kappa)$, under the DDH Assumption. We will use this as the starting point and use an adversary $\mathcal{A}$ against the IND-CPA game of the UPKE construction to construct an adversary $\mathcal{B}$ against the CS+LR Security game of the PKE Scheme.

- The reduction $\mathcal{B}$ receives from the challenger the public key pk_0 corresponding to some secret key $\boldsymbol{s}_0$.
- It has a time period counter t initialized to 0
- $\mathcal{B}$ provides pk_0 to the adversary $\mathcal{A}$.
- $\mathcal{B}$ responds as follows to the oracle queries to $\mathcal{O}_{upd}(\cdot)$ as follows:

  For each input invocation, it increments the counter t to i and records the $\boldsymbol{\delta}_i$ it receives as input.
- $\mathcal{B}$ then receives the challenge messages m_0^*, m_1^*.
- $\mathcal{B}$ then provides the *randomized* leakage function $L(\mathsf{sk}; \boldsymbol{\delta}^*) = \boldsymbol{s}_0 + \boldsymbol{\delta}^*$ where the addition is element-by-element over $\mathbb{Z}_p$. Looking ahead, $\boldsymbol{\delta}^*$ will correspond to the randomness for the fresh update before the secret key is provided to the $\mathcal{A}$. It also sets m_0^*, m_1^* as its challenge messages.
- $\mathcal{B}$ sends to its challenger the leakage function L, m_0^*, m_1^*. It also specifies the function f to be the encryption of each bit of the secret key in the exponent.
- In response, $\mathcal{B}$ receives C which is an encryption of m_b^* under pk_0, C' which is a encryption of $\boldsymbol{s}_0$, bit-by-bit in the exponent, under pk_0, and a leakage $\boldsymbol{z}$ on $\boldsymbol{s}_0$ defined by $\boldsymbol{z} = \boldsymbol{s}_0 + \boldsymbol{\delta}^*$ for *unknown* $\boldsymbol{\delta}^* \leftarrow_{\$} \{0,1\}^\ell$. More formally,

$$C = \mathsf{U\text{-}Enc}(\mathsf{pk}_0, m_b^*); C' = (\mathsf{U\text{-}Enc}(\mathsf{pk}_0, g^{s_1}), \ldots, \mathsf{U\text{-}Enc}(\mathsf{pk}_0, g^{s_\ell}))$$

- At this point, let the time period be q'. Now, $\mathcal{A}$ expects $c^* = \mathsf{U\text{-}Enc}(\mathsf{pk}_{q'}, m_b^*)$. So $\mathcal{B}$ does the following to compute c^*:
  - $\mathcal{B}$ has $C = (\mathsf{U\text{-}Enc}(\mathsf{pk}_0, m_b^*))$ or $C = (f_1, \ldots, f_\ell, c = h^r \cdot m_b^*)$.
  - It computes $\boldsymbol{\Delta}' = \sum_{i=1}^{q'} \boldsymbol{\delta}_i$. $\boldsymbol{\Delta} = (\Delta_1', \ldots, \Delta_\ell')$
  - To convert it into a public key corresponding to $\boldsymbol{s}_{q'} = \boldsymbol{s}_0 + \boldsymbol{\Delta}'$, we do the following:
$$c^* = \left(f_1, \ldots, f_\ell, c \cdot \prod_{j=1}^{\ell} f_j^{\Delta_j'} \right)$$

    This is where we employ the key homomorphism property.
- $\mathcal{B}$ sends to $\mathcal{A}$ the value of c^*.
- $\mathcal{B}$ continues to respond to $\mathcal{O}_{upd}(\cdot)$ queries as before. When $\mathcal{A}$ finally stops, let q be the time period. Now, $\mathcal{B}$ does the following:
  - To compute $\boldsymbol{s}^*$:
    * It sets $\boldsymbol{\Delta} = \sum_{i=1}^{q} \boldsymbol{\delta}_i$. Again, the operation is element-by-element addition over $\mathbb{Z}_p$.

Let $\boldsymbol{\Delta} = (\Delta_1, \ldots, \Delta_\ell)$.

- $*$ With the knowledge of $\boldsymbol{z}$ and $\boldsymbol{\Delta}$, $\mathcal{B}$ sets $\boldsymbol{s}^* = \boldsymbol{s}_{q+1} = \boldsymbol{z} + \boldsymbol{\Delta}$. Recall that $\boldsymbol{z} = \boldsymbol{s}_0 + \boldsymbol{\delta}^*$ for random $\boldsymbol{\delta}^*$. In other words, $\mathcal{B}$ implicitly sets $\boldsymbol{\delta}_{q+1} = \boldsymbol{\delta}^*$, corresponding to the final secure update.

- To compute pk^*: With the knowledge of $\boldsymbol{s}^*$ it is also easy to generate the corresponding public key pk^* by merely computing the value of $h^* = \prod_{i=1}^{\ell} g_i^{s_i^*}$ where $\boldsymbol{s}^* = (s_1^*, \ldots, s_\ell^*)$. Therefore, $\mathsf{pk}^* = (g_1, \ldots, g_\ell, h^*)$
- To compute up^*:
  - $*$ Recall that $\mathsf{up}^* = \big(\mathsf{U\text{-}Enc}(\mathsf{pk}_q, g^{\delta_1}), \ldots \mathsf{U\text{-}Enc}(\mathsf{pk}_q, g^{\delta_\ell}\big)$ where $\boldsymbol{\delta}^* = (\delta_1, \ldots, \delta_\ell))$.
  - $*$ $\mathcal{B}$ has $C' = (\mathsf{U\text{-}Enc}(\mathsf{pk}_0, g^{s_1}), \ldots, \mathsf{U\text{-}Enc}(\mathsf{pk}_0, g^{s_\ell}))$ where $\boldsymbol{s}_0 = (s_1, \ldots, s_\ell))$
  - $*$ Note that for all $i = 1, \ldots, \ell$, by definition, $\delta_i = z_i - s_i \in \mathbb{Z}_p$.
  - $*$ Let $\boldsymbol{ct}_i = \mathsf{Enc}(\mathsf{pk}_0, g^{s_i}) = (f_1, \ldots, f_\ell, c = h^r \cdot g^{s_i})$
  - $*$ For $i = 1, \ldots \ell$, then we transform each $\boldsymbol{ct}_i$ into $\boldsymbol{ct}'_i$ where

$$\boldsymbol{ct}'_i = \left(f'_1 = f_1^{-1}, \ldots, f'_\ell = f_\ell^{-1}, c' = \left(c \cdot g^{-z_i} \cdot \prod_{j=1}^{\ell} f_j^{-\Delta_j} \right)^{-1} \right)$$

  Now, $\mathsf{up}^* = (\boldsymbol{ct}'_1, \ldots, \boldsymbol{ct}'_\ell)$

- Send $(\mathsf{pk}^*, \mathsf{sk}^*, \mathsf{up}^*)$ to $\mathcal{A}$.
- $\mathcal{B}$ forwards $\mathcal{A}$'s guess as its own.

Analysis of Reduction. We first show that the leakage function defined here has sufficiently small entropy loss.

Claim. $\mathrm{H}_\infty(\boldsymbol{s}_0|\boldsymbol{z}) = \ell - \lambda$ where $\lambda = \ell(1 - \log_2(4/3))$

Proof. First note that the components of $\boldsymbol{z} = (z_1, \ldots, z_\ell)$ and $\boldsymbol{s}_0 = (s_1, \ldots, s_\ell)$ are independent of each other so $\mathrm{H}_\infty(\boldsymbol{s}_0|z) = \sum_i \mathrm{H}_\infty(s_i|z_i)$. Now, the distribution of z_i is given by

$$\mathsf{P}[z_i = 0] = \mathsf{P}[z_i = 2] = 1/4, \mathsf{P}[z_i = 1] = 1/2$$

Further,

$$\mathsf{P}[s_i = 0|z_i = 0] = 1, \mathsf{P}[s_i = 0|z_i = 2] = 0, \mathsf{P}[s_i = 0|z_i = 1] = \frac{1}{2}$$

Therefore, $\mathrm{H}_\infty(s_i|z_i) = -\log_2(1 \cdot \mathsf{P}[z_i = 0] + 1 \cdot \mathsf{P}[z_i = 2] + \frac{1}{2}\mathsf{P}[z_i = 1]) = -\log(3/4)$ and $\mathrm{H}_\infty(\boldsymbol{s}_0|\boldsymbol{z}) = \ell \cdot \log_2(4/3)$. □

We now show that the distribution of ciphertext is correct. We will show it is correct for any i. We have $\boldsymbol{c}_i = (f_1, \ldots, f_\ell, c = h^r \cdot g^{s_i})$. First, we first transform it into a cipher text of $\boldsymbol{z} - \boldsymbol{s}_0$, under pk_0. This is message homomorphism. We then transform this ciphertext, under pk_0 to a ciphertext encrypting the same message under pk_q. This is the property of key homomorphism.

- Multiplying c with g^{-z_i} gives us a valid encryption of $g^{s_i - z_i}$. However, we have that $z_i - s_i = d_i$ where $\boldsymbol{\delta}^* = (d_1, \ldots, d_\ell)$.
- To obtain the encryption of $g^{z_i - s_i}$ we merely take the inverse of all elements, and then multiply the last element by g^{z_i}. Therefore,

$$\boldsymbol{c}'_i = (f'_1 = g_1^{r'} = f_1^{-1}, \ldots, f'_\ell = g_\ell^{r'} = f_\ell^{-1}, c' = c^{-1} \cdot g^{z_i} = h^{r'} \cdot g^{-s_i} \cdot g^{z_i})$$

  with $r' = -r$.
- Now, note that $\mathsf{sk}_q = \mathsf{sk}_0 + \boldsymbol{\Delta} = (s_1 + \Delta_1, \ldots, s_\ell + \Delta_\ell)$. The public key is therefore $\mathsf{pk}_q = (g_1, \ldots, g_\ell, h_q)$ where $h_q = h \cdot \prod_{j=1}^{\ell} g_j^{\Delta_j}$. In order to transform a ciphertext $c'_i = (f'_1, \ldots, f'_\ell, c')$ under pk_0 to a ciphertext under pk_q we modify the last component, c' as $c' \cdot \prod_{j=1}^{\ell} f_j'^{\Delta_j} = (c \cdot g^{-z_i} \cdot \prod_{j=1}^{\ell} f_j^{\Delta_j})^{-1}$.

Under this reduction, it is easy to see that $\mathcal{B}$ perfectly simulates the IND-CR-CPA game for $\mathcal{A}$. The advantage of $\mathcal{A}$ against the IND-CR-CPA is the same as the advantage of $\mathcal{B}$. □

Choice of Parameters. We have from Theorem 2 that $\lambda \leq \ell - 2\log p - \omega(\log \kappa)$. We have also shown that our reduction needs $\ell - \lambda = \ell \cdot \log_2(4/3)$. Therefore, we have that $\ell \geq \frac{2}{\log_2(4/3)} \log p + \omega(\log \kappa)$. Or, $\ell = \lceil 5 \log p \rceil$.

6 Constructions based on LWE

This section presents construction from the LWE Assumption. We begin by presenting a slightly modified version of the dual-Regev PKE Scheme [48,33] in section 6.1. We show that the scheme is CS+LR secure in section 6.2. We then present our construction of a UPKE scheme (section 6.3), extended from the PKE scheme. We finally prove that the UPKE scheme is IND-CR-CPA secure in section 6.4.

6.1 The Dual Regev or GPV Cryptosystem

The construction is presented as Construction 3.

Correctness. We show that the decryption algorithm is correct with overwhelming probability (over the choice of the randomness of $\mathsf{Gen}, \mathsf{Enc}$). The decryption algorithm computes:

$$\langle \boldsymbol{r}, \boldsymbol{t} \rangle = \langle \boldsymbol{r}, \boldsymbol{A}^T \boldsymbol{x} + \boldsymbol{e} \rangle = \langle \boldsymbol{r}, \boldsymbol{A}^T \boldsymbol{x} \rangle + \langle \boldsymbol{r}, \boldsymbol{e} \rangle = \langle \boldsymbol{x}, \boldsymbol{u} \rangle + \langle \boldsymbol{r}, \boldsymbol{e} \rangle$$

$$pad - \langle \boldsymbol{r}, \boldsymbol{t} \rangle = \langle \boldsymbol{x}, \boldsymbol{u} \rangle + e' + b \left\lfloor \frac{p}{2} \right\rfloor - \langle \boldsymbol{r}, \boldsymbol{t} \rangle = b \left\lfloor \frac{p}{2} \right\rfloor + (e' - \langle \boldsymbol{e}, \boldsymbol{r} \rangle)$$

Now, note that $e' - \langle e, r \rangle$ is small in comparison to p. Therefore, the computed value is closer to $\lfloor p/2 \rfloor$ when $b = 0$ and the opposite when $b = 1$.

Protocol Dual Regev or GPV Cryptosystem

```
Gen(1^κ)
    Sample $\boldsymbol{A} \leftarrow_\$ \mathbb{Z}_p^{n \times m}$
    Sample $r \leftarrow_\$ \{0,1\}^m$
    Compute $\boldsymbol{u} = \boldsymbol{A}\boldsymbol{r}$
    return $(\mathsf{pk} = (\boldsymbol{A}, \boldsymbol{u}), \mathsf{sk} = (\boldsymbol{r}))$

Enc(pk, $b \in \{0,1\}$)
    Parse $\mathsf{pk} = (\boldsymbol{A}, \boldsymbol{u})$
    Sample $\boldsymbol{x} \leftarrow_\$ \mathbb{Z}_p^n, \boldsymbol{e} \leftarrow_\$ \chi^m, e' \leftarrow \chi'$
    Compute $\boldsymbol{t} = \boldsymbol{A}^T\boldsymbol{x} + \boldsymbol{e}$
    Compute $pad = \langle \boldsymbol{x}, \boldsymbol{u} \rangle + e' + b\lfloor p/2 \rfloor$
    return $c = (\boldsymbol{t}, pad)$

Dec(sk, $c$)
    Parse $c = (\boldsymbol{t}, pad)$ and $\mathsf{sk} = \boldsymbol{r}$
    Compute $b' = (pad - \langle \boldsymbol{r}, \boldsymbol{t} \rangle) \in \mathbb{Z}_p$
    return 0 if $m'$ is closer to 0 than to $\lfloor p/2 \rfloor$ and 1 otherwise.
```

Construction 3. The Dual Regev or GPV Cryptosystem. Let n, m, p be integer parameters of the scheme. We will assume that LWE holds where p is super-polynomial and χ is polynomially bounded. Then, we set χ' to be uniformly random over (say) $[-p/8, p/8]$.

6.2 CS+LR Security of the dual-Regev Cryptosystem

In this section, we provide proof of the combined circular security and leakage resilience of the dual-Regev Cryptosystem. Formally, we will prove the following theorem:

Theorem 4. *Under the LWE Assumption, Construction 3 is λ-CS+LR secure with leakage $\lambda = m - (n+1)\log p - \omega(\log \kappa)$.*

Before we can prove the above theorem, we show the existence of an encryption algorithm Enc' such that

$$(\mathsf{Enc}'(\mathsf{pk}, i), \mathsf{sk}) \approx_c (\mathsf{Enc}(\mathsf{pk}, r_i), \mathsf{sk}).$$

Consider: $\mathsf{Enc}'(\mathsf{pk}, i) := (\boldsymbol{t}', pad')$ where:

- Let $\boldsymbol{x} \leftarrow_\$ \mathbb{Z}_p^n, \boldsymbol{e} \leftarrow_\$ \chi^m$ and $\boldsymbol{d} = (d_1 = 0, \ldots, d_{i-1} = 0, d_i = -\lfloor p/2 \rfloor, d_{i+1} = 0, \ldots, d_m = 0)$. Then, $\boldsymbol{t}' = \boldsymbol{A}^T\boldsymbol{x} + \boldsymbol{e} + \boldsymbol{d}$.
- $pad' = \langle \boldsymbol{x}, \boldsymbol{u} \rangle + e'$ where e' is chosen from a distribution χ' such that $e' + B$ is statistically indistinguishable from e' where $B \in \mathbb{Z}_p$.

Lemma 4. *Under the LWE Assumption,* $(\mathsf{pk}, \mathsf{Enc}'(\mathsf{pk}, i), \mathsf{sk}) \approx_c (\mathsf{pk}, \mathsf{Enc}(\mathsf{pk}, r_i), \mathsf{sk})$. *where* $(\mathsf{pk}, \mathsf{sk}) \leftarrow_\$ \mathsf{Gen}(1^\kappa)$

Proof Sketch. We will prove through a sequence of hybrids, summarized in Table 4. The complete proof of this Lemma can be found in the full version of the paper [29].

Table 4. Proof Outline for Lemma 4

Hybrid	Hybrid Definition	Security
D_0	Enc is used to encrypt r_i	Lemma 1
D_1	D_0 except $\langle \boldsymbol{x}, \boldsymbol{u} \rangle$ replaced with $\langle \boldsymbol{r}, \boldsymbol{t} \rangle$	LWE
D_2	D_1 except $\boldsymbol{t} = \boldsymbol{A}^T\boldsymbol{x} + \boldsymbol{e}$ replaced with $\boldsymbol{t} \leftarrow\!\$\ \mathbb{Z}_p^n$	Identical
D_3	D_2 except $\boldsymbol{t}$ replaced with $\boldsymbol{t} + \boldsymbol{d}$ where $\boldsymbol{d} = (0, \ldots, 0, d_i = -\lfloor p/2 \rfloor, 0, \ldots, 0)$	LWE
D_4	D_3 except $\boldsymbol{t} = \boldsymbol{A}^T\boldsymbol{x} + \boldsymbol{e}$ where $\boldsymbol{x} \leftarrow\!\$\ \mathbb{Z}_p^n, \boldsymbol{e} \leftarrow\!\$\ \chi^m$	Lemma 1
D_5	Enc' is used to encrypt r_i	

Further, note that $\boldsymbol{r}$ is independently chosen, bit by bit. In addition, each Enc, Enc' has independently chosen randomness. Therefore, as a corollary we get that:

Corollary 2. *Under the LWE Assumption,*

$$(\mathsf{pk}, \mathsf{Enc}(\mathsf{pk}, r_1), \ldots, \mathsf{Enc}(\mathsf{pk}, r_m), \mathsf{sk}) \approx_c (\mathsf{pk}, \mathsf{Enc}'(\mathsf{pk}, 1), \ldots, \mathsf{Enc}'(\mathsf{pk}, m), \mathsf{sk})$$

We can now prove the original theorem:

Theorem 4. *Under the LWE Assumption, Construction 3 is λ-CS+LR secure with leakage $\lambda = m - (n+1)\log p - \omega(\log \kappa)$.*

Proof Sketch. We prove this similar to the proof of Theorem 2. This is done through a sequence of hybrids, summarized in Table 5. The complete proof of this Theorem can be found in the full version of the paper [29].

Table 5. Proof Outline for Theorem 4

Hybrid	Hybrid Definition	Security
D_0	The Original CS+LR Security Game, Enc is used	Corollary 2
D_1	D_0 except Enc' is used	Identical
D_2	D_1 except $\langle \boldsymbol{x}, \boldsymbol{u} \rangle$ replaced with $\langle \boldsymbol{r}, \boldsymbol{t} \rangle$	LWE
D_3	D_2 except $\boldsymbol{t} = \boldsymbol{A}^T\boldsymbol{x} + \boldsymbol{e}$ replaced with $\boldsymbol{t} \leftarrow\!\$\ \mathbb{Z}_p^n$.	Leftover Hash Lemma
D_4	D_3 except $\langle \boldsymbol{r}, \boldsymbol{t} \rangle$ replaced with $U \leftarrow \mathbb{Z}_p$	

6.3 UPKE Construction

In this section, we present our construction of an updatable public key encryption based on the dual-Regev cryptosystem. This is presented in Construction 4.

Protocol LWE-Based UPKE

```
$\mathsf{U\text{-}PKEG}(1^\kappa)$
    Sample $\boldsymbol{A} \leftarrow_\$ \mathbb{Z}_p^{n \times m}$
    Sample $\boldsymbol{r} \leftarrow_\$ \{0,1\}^m$
    Compute $\boldsymbol{u} = \boldsymbol{A}\boldsymbol{r}$
    return $(\mathsf{pk} = (\boldsymbol{A}, \boldsymbol{u}), \mathsf{sk} = (\boldsymbol{r}))$

$\mathsf{U\text{-}Enc}(\mathsf{pk}, b \in \{0,1\})$
    Parse $\mathsf{pk} = (\boldsymbol{A}, \boldsymbol{u})$
    Sample $\boldsymbol{x} \leftarrow_\$ \mathbb{Z}_p^n, \boldsymbol{e} \leftarrow_\$ \chi^m, e' \leftarrow \chi'$
    Compute $\boldsymbol{t} = \boldsymbol{A}^T\boldsymbol{x} + \boldsymbol{e}$, $pad = \langle \boldsymbol{x}, \boldsymbol{u} \rangle + e' + b\lfloor p/2 \rfloor$
    return $c = (\boldsymbol{t}, pad)$

$\mathsf{U\text{-}Dec}(\mathsf{sk}, c)$
    Parse $c = (\boldsymbol{t}, pad)$ and $\mathsf{sk} = \boldsymbol{r}$
    Compute $b' = (pad - \langle \boldsymbol{r}, \boldsymbol{t} \rangle) \in \mathbb{Z}_p$
    return 0 if $m'$ is closer to 0 than to $\lfloor p/2 \rfloor$ and 1
    otherwise.

$\mathsf{Upd\text{-}Pk}(\mathsf{pk})$
    Parse $\mathsf{pk} = (\boldsymbol{A}, \boldsymbol{u})$
    Sample $\boldsymbol{\delta} = (\delta_1, \ldots, \delta_m) \leftarrow_\$ \{0,1\}^m$
    Compute $\boldsymbol{u}' = \boldsymbol{u} + \boldsymbol{A}\boldsymbol{\delta}$
    Encrypt $\delta$ bit-by-bit, i.e., $\mathsf{up} =$
    $(\mathsf{U\text{-}Enc}(\mathsf{pk}, \delta_1), \ldots, \mathsf{U\text{-}Enc}(\mathsf{pk}, \delta_m))$.
    return $(\mathsf{up}, \mathsf{pk}' = (\boldsymbol{A}, \boldsymbol{u}'))$

$\mathsf{Upd\text{-}Sk}(\mathsf{sk}, \mathsf{up})$
    Parse $\mathsf{up} = (c_1, \ldots, c_m)$
    for $i = 1, \ldots, m$ do
        $\delta_i = \mathsf{U\text{-}Dec}(\mathsf{sk}, c_i)$
    Compute $\boldsymbol{r}' = \boldsymbol{r} + \boldsymbol{\delta}$ where $\boldsymbol{\delta} = (\delta_1, \ldots, \delta_m)$
    return $\mathsf{sk}' = (\boldsymbol{r}')$
```

Construction 4. LWE Based Construction. Let n, m, p be integer parameters of the scheme. We will assume that LWE holds where p is super-polynomial and χ is polynomially bounded. Then, we set χ' to be uniformly random over (say) $[-p/8, p/8]$. Further, we have that $m \geq \frac{(n+1)}{\log_2(4/3)} \log p + \omega(\log \kappa)$.

Correctness. The property of correctness of UPKE requires that a bit b encrypted by an updated public key decrypts to the same bit b when the corresponding updated secret key is used.

– $(\mathsf{pk} = (\boldsymbol{A}, \boldsymbol{u} = \boldsymbol{A}\boldsymbol{r}), \mathsf{sk} = \boldsymbol{r}) \leftarrow \mathsf{U\text{-}PKEG}(1^\kappa)$
– We have the update bit $\boldsymbol{\delta} \leftarrow_\$ \{0,1\}^m$. We have the updated public key $\mathsf{pk}' = (\boldsymbol{A}, \boldsymbol{u}')$ where $\boldsymbol{u}' = \boldsymbol{u} + \boldsymbol{A}\boldsymbol{\delta}$. We also have $\mathsf{sk}' = \boldsymbol{r}' = \boldsymbol{r} + \boldsymbol{\delta}$
– Let us look at $\mathsf{U\text{-}Enc}(\mathsf{pk}', b)$. It produces ciphertext $(\boldsymbol{t}, pad)$ where $\boldsymbol{t} = \boldsymbol{A}^T\boldsymbol{x}+\boldsymbol{e}$, $pad = \langle \boldsymbol{x}, \boldsymbol{u}' \rangle + e' + b\lfloor p/2 \rfloor$.
– Now, let us look at $\mathsf{U\text{-}Dec}(\boldsymbol{r}', (\boldsymbol{t}, pad))$. It computes

$$\begin{aligned} pad - \langle \boldsymbol{r}', \boldsymbol{t} \rangle &= \langle \boldsymbol{x}, \boldsymbol{u}' \rangle + e' + b\lfloor p/2 \rfloor - \langle \boldsymbol{r} + \boldsymbol{\delta}, \boldsymbol{A}^T\boldsymbol{x} + \boldsymbol{e} \rangle \\ &= \langle \boldsymbol{x}, \boldsymbol{A}\boldsymbol{r} + \boldsymbol{A}\boldsymbol{\delta} \rangle + e' + b\lfloor p/2 \rfloor - \langle \boldsymbol{r} + \boldsymbol{\delta}, \boldsymbol{A}^T\boldsymbol{x} + \boldsymbol{e} \rangle \\ &= \langle \boldsymbol{x}, \boldsymbol{A}(\boldsymbol{r} + \boldsymbol{\delta}) \rangle - \langle \boldsymbol{r} + \boldsymbol{\delta}, \boldsymbol{A}^T, \boldsymbol{x} \rangle + e' - \langle \boldsymbol{e}, \boldsymbol{r} \rangle + b\lfloor p/2 \rfloor \\ &= e' - \langle \boldsymbol{e}, \boldsymbol{r} \rangle + b\lfloor p/2 \rfloor \end{aligned}$$

– Now, note that $e' - \langle e, r \rangle$ is small in comparison to p. Therefore, the computed value is closer to $\lfloor p/2 \rfloor$ when $b = 0$ and the opposite when $b = 1$.

6.4 Security of the UPKE Construction

Theorem 5. *Under the LWE Assumption, Construction 4 is IND-CR-CPA secure UPKE.*

Proof. The proof is very similar to the proof of Theorem 3. We proved in Theorem 4 that the PKE scheme was CS+LR secure with $\lambda \leq m - (n+1)\log p - \omega(\log \kappa)$, under the LWE assumption. We will use this to construct $\mathcal{B}$ against the CS+LR game by using $\mathcal{A}$ against the IND-CPA Game.

- The reduction $\mathcal{B}$ receives from the challenger the public key pk_0 corresponding to some secret key $\boldsymbol{s}_0$.
- It has a time period counter t initialized to 0
- $\mathcal{B}$ provides pk_0 to the adversary $\mathcal{A}$.
- $\mathcal{B}$ responds as follows to the oracle queries to $\mathcal{O}_{upd}(\cdot)$ as follows:

  For each input invocation, it increments the counter t to i and records the $\boldsymbol{\delta}_i$ it receives as input.
- $\mathcal{B}$ then receives the challenge messages m_0^*, m_1^*.
- $\mathcal{B}$ then provides the *randomized* leakage function $L(\mathsf{sk}; \boldsymbol{\delta}^*) = \boldsymbol{s}_0 + \boldsymbol{\delta}^*$ where the addition is element-by-element over $\mathbb{Z}_p$. Looking ahead, $\boldsymbol{\delta}^*$ will correspond to the randomness for the fresh update before the secret key is provided to the $\mathcal{A}$. It also sets m_0^*, m_1^* as its challenge messages.
- $\mathcal{B}$ sends to its challenger the leakage function L, m_0^*, m_1^*. It also specifies the function f to be the encryption of each bit of the secret key.
- In response, $\mathcal{B}$ receives C which is an encryption of m_b^* under pk_0, C' which is a encryption of $\boldsymbol{s}_0$, bit-by-bit, under pk_0, and a leakage $\boldsymbol{z}$ on $\boldsymbol{r}_0$ defined by $\boldsymbol{z} = \boldsymbol{r}_0 + \boldsymbol{\delta}^*$ for *unknown* $\boldsymbol{\delta}^* \leftarrow_{\$} \{0,1\}^m$. More formally,

$$C = \mathsf{U\text{-}Enc}(\mathsf{pk}_0, m_b^*); C' = (\mathsf{U\text{-}Enc}(\mathsf{pk}_0, r_1), \ldots, \mathsf{U\text{-}Enc}(\mathsf{pk}_0, r_m))$$

- At this point, let the time period be q'. Now, $\mathcal{A}$ expects $c^* = \mathsf{U\text{-}Enc}(\mathsf{pk}_{q'}, m_b^*)$. So $\mathcal{B}$ does the following to compute c^*:
  - $\mathcal{B}$ has $C = \mathsf{U\text{-}Enc}(\mathsf{pk}_0, m_b^*)$ or $C = \Big(\boldsymbol{t} = \boldsymbol{A}^T\boldsymbol{x} + \boldsymbol{e}, pad = \langle \boldsymbol{x}, \boldsymbol{u}\rangle + e' + m_b^* \lfloor p/2 \rfloor\Big)$.
  - It computes $\boldsymbol{\Delta}' = \sum_{i=1}^{q'} \boldsymbol{\delta}_i$.
  - It computes $pad^* = pad + \langle \boldsymbol{\Delta}', \boldsymbol{t}\rangle$ and sets $\boldsymbol{t}^* = \boldsymbol{t}$.
  - Now, $c^* = (\boldsymbol{t}^*, pad^*)$
- $\mathcal{B}$ sends to $\mathcal{A}$ the value of c^*.
- $\mathcal{B}$ continues to respond to $\mathcal{O}_{upd}(\cdot)$ queries as before. When $\mathcal{A}$ finally stops, let q be the time period. Now, $\mathcal{B}$ does the following:
  - To compute $\mathsf{sk}^* = \boldsymbol{r}^*$:
    - Set $\boldsymbol{\Delta} = \sum_{i=1}^{q} \boldsymbol{\delta}_i$.
    - With the knowledge of $\boldsymbol{z}, \boldsymbol{\Delta}$, $\mathcal{B}$ sets $\boldsymbol{r}^* = \boldsymbol{r}_{q+1} = \boldsymbol{z} + \boldsymbol{\Delta}$.

- To compute pk^*: With the knowledge of $\boldsymbol{A}, \boldsymbol{r}^*$, $\mathcal{B}$ computes $\boldsymbol{u}^* = \boldsymbol{A}\boldsymbol{r}^*$. It sets $\mathsf{pk}^* = (\boldsymbol{A}, \boldsymbol{u}^*)$.
- To compute up^*: $\mathcal{B}$ has bit-by-bit encryption of $\boldsymbol{r}_0$. It needs to compute the bit-by-bit encryption of $\boldsymbol{\delta}^* = \boldsymbol{z} - \boldsymbol{r}_0$. For simplicity, assume that $\boldsymbol{z}$ is a trit, i.e., taking value 0, 1, 2. Let $\boldsymbol{r}_0 = (r_1, \ldots, r_m)$, $\boldsymbol{\delta}^* = (d_1, \ldots, d_m)$ and $\boldsymbol{z} = (z_1, \ldots, z_m)$. Recall that $r_i, d_i \in \{0, 1\}$ while $z_i \in \{0, 1, 2\}$.

  We will first look at how to transform $\mathsf{U\text{-}Enc}(\mathsf{pk}_0, r_i)$ to $\mathsf{U\text{-}Enc}(\mathsf{pk}_0, d_i)$.

  * If $z_i = 2$, then we have that $r_i = d_i = 1$. Therefore, $\mathsf{U\text{-}Enc}(\mathsf{pk}_0, r_i) = \mathsf{U\text{-}Enc}(\mathsf{pk}_0, d_i)$ and we do not need to do anything.
  * Similarly if $z_i = 0$, then we have that $r_i = d_i = 0$. Once again, $\mathsf{U\text{-}Enc}(\mathsf{pk}_0, r_i) = \mathsf{U\text{-}Enc}(\mathsf{pk}_0, d_i)$ and we do not need to do anything.
  * If $z_i = 1$, then we merely need $\mathsf{U\text{-}Enc}(\mathsf{pk}_0, r_i)$ to be modified to $\mathsf{U\text{-}Enc}(\mathsf{pk}_0, 1 - r_i)$. To achieve this we merely add $\lfloor p/2 \rfloor$ to the second term in the ciphertext.

  To convert $\mathsf{U\text{-}Enc}(\mathsf{pk}_0, d_i)$ to $\mathsf{U\text{-}Enc}(\mathsf{pk}_q, d_i)$ we do the following:

  * Note that $\mathsf{U\text{-}Enc}(\mathsf{pk}_0, d_i) = (\boldsymbol{t}_0, pad_0)$ where $\boldsymbol{t}_0 = \boldsymbol{A}^T\boldsymbol{x} + \boldsymbol{e}$ and $pad_0 = \langle \boldsymbol{x}, \boldsymbol{u}_0 \rangle + e' + b\lfloor p/2 \rfloor$. Further, $\boldsymbol{u}_q = \boldsymbol{u}_0 + \boldsymbol{A}\boldsymbol{\Delta}$
  * Let $t_q = t_0$, then $pad_q = pad_0 + \langle \boldsymbol{\Delta}, \boldsymbol{t}_0 \rangle$. with the choice

– Send $(\mathsf{pk}^*, \mathsf{sk}^*, \mathsf{up}^*)$ to $\mathcal{A}$.

– $\mathcal{B}$ forwards $\mathcal{A}$'s guess as its own.

Analysis of the Reduction. We first show that the leakage function has sufficiently small entropy loss.

Claim. $\mathrm{H}_\infty(\boldsymbol{r}_0 | \boldsymbol{z}) = m - \lambda$, where $\lambda = m(1 - \log_2(4/3))$

The above is identical to Claim 5.4 in the proof of Theorem 3.

We then need to show that the distribution of ciphertext is correct. Specifically, the distribution of the update ciphertext. We have $t_0 = t_q$. We will show that pad_q is correctly distributed. By definition we have that: $pad_q = \langle \boldsymbol{x}, \boldsymbol{u}_q \rangle + e' + b\lfloor p/2 \rfloor$. Here, we compute pad_q as follows:

$$\begin{aligned} pad_q = pad_0 + \langle \boldsymbol{\Delta}, \boldsymbol{t}_0 \rangle &= \langle \boldsymbol{x}, \boldsymbol{u}_0 \rangle + e' + b\lfloor p/2 \rfloor + \langle \boldsymbol{\Delta}, \boldsymbol{t}_0 \rangle \\ &= \langle \boldsymbol{x}, \boldsymbol{u}_0 \rangle + e' + b\lfloor p/2 \rfloor + \langle \boldsymbol{\Delta}, \boldsymbol{A}^T\boldsymbol{x} + \boldsymbol{e} \rangle \\ &= \langle \boldsymbol{x}, \boldsymbol{u}_0 \rangle + \langle \boldsymbol{A}\boldsymbol{\Delta}, \boldsymbol{x} \rangle + e' + \langle \boldsymbol{\Delta}, \boldsymbol{e} \rangle + b\lfloor p/2 \rfloor \\ &= \langle \boldsymbol{x}, \boldsymbol{u}_q \rangle + e' + \langle \boldsymbol{\Delta}, \boldsymbol{e} \rangle + b\lfloor p/2 \rfloor \end{aligned}$$

We can now use the definition of distribution e' and Lemma 1 to show that the computed distribution is statistically indistinguishable from the actual distribution.

Under this reduction, it is easy to see that $\mathcal{B}$ perfectly simulates the IND-CPA game for $\mathcal{A}$. The advantage of $\mathcal{A}$ against the IND-CPA is the same as the advantage of $\mathcal{B}$. □

Choice of Parameters. From Theorem 4, we have that $m - \lambda \geq (n+1)\log p + \omega(\log \kappa)$. Further, we have from the above claim that $m - \lambda = m \log_2(4/3)$. Putting the two together, we get $m \geq \frac{n+1}{\log_2(4/3)} \log p + \omega(\log \kappa)$.

7 Towards Stronger Security

In this section, we begin by presenting the CCA extension of the CPA security game presented in Section 3. We later extend the CPA and CCA security to a stronger definition. Due to space constraints, we invite the readers to refer to the full version of this paper [29] for constructions that satisfy these definitions and proofs of security.

IND-CR-CCA Security of UPKE. In Section 3, we defined a CPA based security for an updatable public-key encryption. However, a natural extension is to consider CCA based security of the UPKE. We will call this IND-CR-CCA which is the abbreviation of INDistinguishability under Chosen Randomness Chosen Ciphertext Attack. In this setting, the adversary is given access to also the decryption oracle where the adversary can ask for decryption of a ciphertext under the current secret key on a ciphertext of its choice or creation. To model this access, we define two oracles:

– $\mathcal{O}_{upd}(\cdot)$: The challenger on receiving the randomness r_i from the adversary does the following:

$$(\mathsf{up}_i, \mathsf{pk}_i) \leftarrow \mathsf{Upd\text{-}Pk}(\mathsf{pk}_{i-1}; r_i);\ \mathsf{sk}_i \leftarrow \mathsf{Upd\text{-}Sk}(\mathsf{sk}_{i-1}, \mathsf{up}_i)\ .$$

– $\mathcal{D}(\cdot)$: The challenger on receiving ciphertext c as input, returns $\mathsf{U\text{-}Dec}(sk_i, c)$ where i is the current epoch and sk_i is the secret key of the current epoch.

IND-CR-CCA Security. For any adversary $\mathcal{A}$ with running time t and we consider the
IND-CR-CCA security game:

- Sample $(\mathsf{sk}_0, \mathsf{pk}_0) \leftarrow \mathsf{U\text{-}PKEG}(1^\kappa)$, $b \leftarrow\$ \{0,1\}$.
- $(m_0^*, m_1^*, state) \leftarrow\$ \mathcal{A}^{\mathcal{O}_{upd}(\cdot), \mathcal{O}_{dec}(\cdot)}(\mathsf{pk}_0)$
- Compute $c^* \leftarrow\$ \mathsf{U\text{-}Enc}(\mathsf{pk}_{q'}, m_b^*)$ where q' is the current time period.
- Compute $state \leftarrow\$ \mathcal{A}^{\mathcal{O}_{upd}(\cdot), \mathcal{O}_{dec}(\cdot)}(c^*, state)$.
  - Here $\mathcal{A}$ is not allowed to query its $\mathcal{O}_{dec}(\cdot)$ oracle on the challenge ciphertext c^* until $\mathcal{A}$ makes at least one (arbitrary) query to its $\mathcal{O}_{upd}(\cdot)$ oracle.
- Choose uniformly random r^* and then compute

$$(\mathsf{up}^*, \mathsf{pk}^*) \leftarrow \mathsf{Upd\text{-}Pk}(\mathsf{pk}_q; r^*);\ \mathsf{sk}^* \leftarrow \mathsf{Upd\text{-}Sk}(\mathsf{sk}_q, \mathsf{up}^*)\ .$$

  where q is the current time period.

- $b' \leftarrow_{\$} \mathcal{A}(\mathsf{pk}^*, \mathsf{sk}^*, \mathsf{up}^*, \mathit{state})$. Note that the adversary is not given access to the decryption query as with knowledge of sk^*, it can perform the decryption on its own.
- $\mathcal{A}$ wins the game if $b = b'$. The advantage of $\mathcal{A}$ in winning the above game is denoted by $\mathrm{Adv}^{\mathsf{UPKE}}_{\mathrm{crcca}}(\mathcal{A}) = |\mathsf{P}[b = b'] - \frac{1}{2}|$.

Definition 7. *An updatable public-key encryption scheme* UPKE *is IND-CR-CCA - secure if for all PPT attackers $\mathcal{A}$, its advantage $\mathrm{Adv}^{\mathsf{UPKE}}_{\mathrm{crcca}}(\mathcal{A})$ is negligible.*

Stronger CPA, CCA Security. In the definition of both the IND-CR-CPA and the IND-CR-CCA security, we allowed the adversary to provide bad randomness and the challenger honestly updated the public key based on this bad randomness. However, one can consider a stronger attack where the attacker could provide an arbitrary update ciphertext up and the new public key pk'. In other words, the adversary chooses the full *output* $(\mathsf{up}, \mathsf{pk}')$ of the public key update algorithm $\mathsf{Upd\text{-}Pk}$, rather than its *input* r. The challenger will then check — using a special new algorithm (see below) $\mathsf{Verify\text{-}Upd}(\mathsf{pk}, \mathsf{up}, \mathsf{pk}')$ — that the supplied values are "consistent". If so, it will update the secret key using $\mathsf{sk} \leftarrow \mathsf{Upd\text{-}Sk}(\mathsf{sk}, \mathsf{up})$, as before. Otherwise, it will ignore this query of the attacker. With this intuition in mind, we formalize the changes in the syntax and security of UPKE.

Syntactic Changes. We introduce a new algorithm $\mathsf{Verify\text{-}Upd}(\mathsf{pk}, \mathsf{up}, \mathsf{pk}')$ where pk is the old public key, up is the update ciphertext and pk' is the updated public key. This algorithm outputs 1 iff pk' is consistently produced by up and pk.

Security Game Changes. We also change the definition of $\mathcal{O}_{upd}(\cdot)$. The new definition is as follows:

– $\mathcal{O}_{upd}(\cdot, \cdot)$: This takes as input two values up and pk'. It then runs $\tau \leftarrow \mathsf{Verify\text{-}Upd}(\mathsf{pk}, \mathsf{up}, \mathsf{pk}')$. If $\tau = 1$, it runs $\mathsf{sk}' \leftarrow \mathsf{Upd\text{-}Sk}(\mathsf{sk}, \mathsf{up})$, else it returns $\perp$.

References

1. Abdalla, M., Reyzin, L.: A new forward-secure digital signature scheme. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 116–129. Springer, Heidelberg (Dec 2000). https://doi.org/10.1007/3-540-44448-3˙10
2. Agrawal, S., Boneh, D., Boyen, X.: Efficient lattice (H)IBE in the standard model. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 553–572. Springer, Heidelberg (May / Jun 2010). https://doi.org/10.1007/978-3-642-13190-5˙28
3. Akavia, A., Goldwasser, S., Vaikuntanathan, V.: Simultaneous hardcore bits and cryptography against memory attacks. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 474–495. Springer, Heidelberg (Mar 2009). https://doi.org/10.1007/978-3-642-00457-5˙28
4. Alperin-Sheriff, J., Peikert, C.: Circular and KDM security for identity-based encryption. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 334–352. Springer, Heidelberg (May 2012). https://doi.org/10.1007/978-3-642-30057-8˙20

5. Alwen, J., Coretti, S., Dodis, Y., Tselekounis, Y.: Security analysis and improvements for the IETF MLS standard for group messaging. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Part I. LNCS, vol. 12170, pp. 248–277. Springer, Heidelberg (Aug 2020). https://doi.org/10.1007/978-3-030-56784-2˙9
6. Anderson, R.: Invited lecture. Fourth Annual Conference on Computer and Communications Security, ACM (1997)
7. Applebaum, B.: Key-dependent message security: Generic amplification and completeness. Journal of Cryptology **27**(3), 429–451 (Jul 2014). https://doi.org/10.1007/s00145-013-9149-6
8. Applebaum, B., Cash, D., Peikert, C., Sahai, A.: Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 595–618. Springer, Heidelberg (Aug 2009). https://doi.org/10.1007/978-3-642-03356-8˙35
9. Asharov, G., Jain, A., López-Alt, A., Tromer, E., Vaikuntanathan, V., Wichs, D.: Multiparty computation with low communication, computation and interaction via threshold FHE. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 483–501. Springer, Heidelberg (Apr 2012). https://doi.org/10.1007/978-3-642-29011-4˙29
10. Barak, B., Haitner, I., Hofheinz, D., Ishai, Y.: Bounded key-dependent message security. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 423–444. Springer, Heidelberg (May / Jun 2010). https://doi.org/10.1007/978-3-642-13190-5˙22
11. Bellare, M., Miner, S.K.: A forward-secure digital signature scheme. In: Wiener, M.J. (ed.) CRYPTO'99. LNCS, vol. 1666, pp. 431–448. Springer, Heidelberg (Aug 1999). https://doi.org/10.1007/3-540-48405-1˙28
12. Bellare, M., Yee, B.S.: Forward-security in private-key cryptography. In: Joye, M. (ed.) CT-RSA 2003. LNCS, vol. 2612, pp. 1–18. Springer, Heidelberg (Apr 2003). https://doi.org/10.1007/3-540-36563-X˙1
13. Black, J., Rogaway, P., Shrimpton, T.: Encryption-scheme security in the presence of key-dependent messages. In: Nyberg, K., Heys, H.M. (eds.) SAC 2002. LNCS, vol. 2595, pp. 62–75. Springer, Heidelberg (Aug 2003). https://doi.org/10.1007/3-540-36492-7˙6
14. Boneh, D., Boyen, X.: Efficient selective-ID secure identity based encryption without random oracles. In: Cachin, C., Camenisch, J. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 223–238. Springer, Heidelberg (May 2004). https://doi.org/10.1007/978-3-540-24676-3˙14
15. Boneh, D., Boyen, X., Goh, E.J.: Hierarchical identity based encryption with constant size ciphertext. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 440–456. Springer, Heidelberg (May 2005). https://doi.org/10.1007/11426639˙26
16. Boneh, D., Eskandarian, S., Kim, S., Shih, M.: Improving speed and security in updatable encryption schemes (2020), to appear in Asiacrypt 2020
17. Boneh, D., Halevi, S., Hamburg, M., Ostrovsky, R.: Circular-secure encryption from decision Diffie-Hellman. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 108–125. Springer, Heidelberg (Aug 2008). https://doi.org/10.1007/978-3-540-85174-5˙7
18. Boneh, D., Lewi, K., Montgomery, H.W., Raghunathan, A.: Key homomorphic PRFs and their applications. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 410–428. Springer, Heidelberg (Aug 2013). https://doi.org/10.1007/978-3-642-40041-4˙23

19. Boyd, C., Davies, G.T., Gjøsteen, K., Jiang, Y.: Fast and secure updatable encryption. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Part I. LNCS, vol. 12170, pp. 464–493. Springer, Heidelberg (Aug 2020). https://doi.org/10.1007/978-3-030-56784-2˙16
20. Brakerski, Z., Goldwasser, S.: Circular and leakage resilient public-key encryption under subgroup indistinguishability - (or: Quadratic residuosity strikes back). In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 1–20. Springer, Heidelberg (Aug 2010). https://doi.org/10.1007/978-3-642-14623-7˙1
21. Brakerski, Z., Goldwasser, S., Kalai, Y.T.: Black-box circular-secure encryption beyond affine functions. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 201–218. Springer, Heidelberg (Mar 2011). https://doi.org/10.1007/978-3-642-19571-6˙13
22. Brakerski, Z., Lombardi, A., Segev, G., Vaikuntanathan, V.: Anonymous IBE, leakage resilience and circular security from new assumptions. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part I. LNCS, vol. 10820, pp. 535–564. Springer, Heidelberg (Apr / May 2018). https://doi.org/10.1007/978-3-319-78381-9˙20
23. Camenisch, J., Lysyanskaya, A.: An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 93–118. Springer, Heidelberg (May 2001). https://doi.org/10.1007/3-540-44987-6˙7
24. Canetti, R., Halevi, S., Katz, J.: A forward-secure public-key encryption scheme. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 255–271. Springer, Heidelberg (May 2003). https://doi.org/10.1007/3-540-39200-9˙16
25. Cash, D., Hofheinz, D., Kiltz, E., Peikert, C.: Bonsai trees, or how to delegate a lattice basis. Journal of Cryptology **25**(4), 601–639 (Oct 2012). https://doi.org/10.1007/s00145-011-9105-2
26. Dodis, Y., Goldwasser, S., Kalai, Y.T., Peikert, C., Vaikuntanathan, V.: Public-key encryption schemes with auxiliary inputs. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 361–381. Springer, Heidelberg (Feb 2010). https://doi.org/10.1007/978-3-642-11799-2˙22
27. Dodis, Y., Haralambiev, K., López-Alt, A., Wichs, D.: Efficient public-key cryptography in the presence of key leakage. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 613–631. Springer, Heidelberg (Dec 2010). https://doi.org/10.1007/978-3-642-17373-8˙35
28. Dodis, Y., Jost, D., Karthikeyan, H.: Forward-secure encryption with fast forwarding. Manuscript (2021)
29. Dodis, Y., Karthikeyan, H., Wichs, D.: Updatable public key encryption in the standard model. `https://cs.nyu.edu/~dodis/ps/upke.pdf` (2021)
30. Döttling, N., Garg, S.: From selective IBE to full IBE and selective HIBE. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017, Part I. LNCS, vol. 10677, pp. 372–408. Springer, Heidelberg (Nov 2017). https://doi.org/10.1007/978-3-319-70500-2˙13
31. Döttling, N., Garg, S.: Identity-based encryption from the Diffie-Hellman assumption. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part I. LNCS, vol. 10401, pp. 537–569. Springer, Heidelberg (Aug 2017). https://doi.org/10.1007/978-3-319-63688-7˙18
32. Everspaugh, A., Paterson, K.G., Ristenpart, T., Scott, S.: Key rotation for authenticated encryption. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part III. LNCS, vol. 10403, pp. 98–129. Springer, Heidelberg (Aug 2017). https://doi.org/10.1007/978-3-319-63697-9˙4

33. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Ladner, R.E., Dwork, C. (eds.) 40th ACM STOC. pp. 197–206. ACM Press (May 2008). https://doi.org/10.1145/1374376.1374407
34. Gentry, C., Silverberg, A.: Hierarchical ID-based cryptography. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 548–566. Springer, Heidelberg (Dec 2002). https://doi.org/10.1007/3-540-36178-2˙34
35. Horwitz, J., Lynn, B.: Toward hierarchical identity-based encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 466–481. Springer, Heidelberg (Apr / May 2002). https://doi.org/10.1007/3-540-46035-7˙31
36. Itkis, G., Reyzin, L.: Forward-secure signatures with optimal signing and verifying. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 332–354. Springer, Heidelberg (Aug 2001). https://doi.org/10.1007/3-540-44647-8˙20
37. Jaeger, J., Stepanovs, I.: Optimal channel security against fine-grained state compromise: The safety of messaging. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part I. LNCS, vol. 10991, pp. 33–62. Springer, Heidelberg (Aug 2018). https://doi.org/10.1007/978-3-319-96884-1˙2
38. Jost, D., Maurer, U., Mularczyk, M.: Efficient ratcheting: Almost-optimal guarantees for secure messaging. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019, Part I. LNCS, vol. 11476, pp. 159–188. Springer, Heidelberg (May 2019). https://doi.org/10.1007/978-3-030-17653-2˙6
39. Kalai, Y.T., Reyzin, L.: A survey of leakage-resilient cryptography. Cryptology ePrint Archive, Report 2019/302 (2019), https://eprint.iacr.org/2019/302
40. Klooß, M., Lehmann, A., Rupp, A.: (R)CCA secure updatable encryption with integrity protection. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019, Part I. LNCS, vol. 11476, pp. 68–99. Springer, Heidelberg (May 2019). https://doi.org/10.1007/978-3-030-17653-2˙3
41. Kozlov, A., Reyzin, L.: Forward-secure signatures with fast key update. In: Cimato, S., Galdi, C., Persiano, G. (eds.) SCN 02. LNCS, vol. 2576, pp. 241–256. Springer, Heidelberg (Sep 2003). https://doi.org/10.1007/3-540-36413-7˙18
42. Lehmann, A., Tackmann, B.: Updatable encryption with post-compromise security. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part III. LNCS, vol. 10822, pp. 685–716. Springer, Heidelberg (Apr / May 2018). https://doi.org/10.1007/978-3-319-78372-7˙22
43. Malkin, T., Micciancio, D., Miner, S.K.: Efficient generic forward-secure signatures with an unbounded number of time periods. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 400–417. Springer, Heidelberg (Apr / May 2002). https://doi.org/10.1007/3-540-46035-7˙27
44. Malkin, T., Teranishi, I., Yung, M.: Efficient circuit-size independent public key encryption with KDM security. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 507–526. Springer, Heidelberg (May 2011). https://doi.org/10.1007/978-3-642-20465-4˙28
45. Naor, M., Reingold, O.: Number-theoretic constructions of efficient pseudo-random functions. In: 38th FOCS. pp. 458–467. IEEE Computer Society Press (Oct 1997). https://doi.org/10.1109/SFCS.1997.646134
46. Naor, M., Segev, G.: Public-key cryptosystems resilient to key leakage. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 18–35. Springer, Heidelberg (Aug 2009). https://doi.org/10.1007/978-3-642-03356-8˙2
47. Poettering, B., Rösler, P.: Towards bidirectional ratcheted key exchange. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part I. LNCS, vol. 10991, pp. 3–32. Springer, Heidelberg (Aug 2018). https://doi.org/10.1007/978-3-319-96884-1˙1

48. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: Gabow, H.N., Fagin, R. (eds.) 37th ACM STOC. pp. 84–93. ACM Press (May 2005). https://doi.org/10.1145/1060590.1060603