Leakage-Resilient Public-Key Encryption from Obfuscation

Dana Dachman-Soled* S. Dov Gordon† Feng-Hao Liu‡ Adam O'Neill§
Hong-Sheng Zhou¶

July 25, 2016

Abstract

The literature on leakage-resilient cryptography contains various leakage models that provide different levels of security. In this work, we consider the *bounded leakage* and the *continual leakage* models. In the bounded leakage model (Akavia et al. – TCC 2009), it is assumed that there is a fixed upper bound L on the number of bits the attacker may leak on the secret key in the entire lifetime of the scheme. Alternatively, in the continual leakage model (Brakerski et al. – FOCS 2010, Dodis et al. – FOCS 2010), the lifetime of a cryptographic scheme is divided into "time periods" between which the scheme's secret key is updated. Furthermore, in its attack the adversary is allowed to obtain some bounded amount of leakage on the current secret key during each time period.

In the continual leakage model, a challenging problem has been to provide security against *leakage on key updates*, that is, leakage that is a function not only of the current secret key but also the *randomness used to update it*. We propose a new, modular approach to overcome this problem. Namely, we present a compiler that transforms any public-key encryption or signature scheme that achieves a slight strengthening of continual leakage resilience, which we call *consecutive* continual leakage resilience, to one that is continual leakage resilient with leakage on key updates, assuming *indistinguishability obfuscation* (Barak et al. — CRYPTO 2001, Garg et al. – FOCS 2013). Under the stronger assumption of *public-coin differing-inputs obfuscation* (Ishai et al. – TCC 2015) the leakage rate tolerated by our compiled scheme is essentially as good as that of the starting scheme. Our compiler is obtained by making a new connection between the problems of leakage on key updates and so-called "sender-deniable" encryption (Canetti et al. – CRYPTO 1997), which was recently realized for the first time by Sahai and Waters (STOC 2014).

In the bounded leakage model, we develop a new approach to constructing leakage-resilient encryption from obfuscation, based upon the public-key encryption scheme from $i\mathcal{O}$ and punctured pseudorandom functions due to Sahai and Waters (STOC 2014). In particular, we achieve leakage-resilient public key encryption tolerating L bits of leakage for any L from $i\mathcal{O}$ and one-way functions. We build on this to achieve leakage-resilient public key encryption with optimal leakage rate of $1 - o(1)$ based on public-coin differing-inputs obfuscation and collision-resistant hash functions. Such a leakage rate is not known to be achievable in a generic way based on public-key encryption alone. We then develop entirely new techniques to construct a new public key encryption scheme that is secure under (consecutive) continual leakage resilience (under appropriate assumptions), which we believe is of independent interest.

*Dept. of Electrical and Computer Engineering, University of Maryland. Email: danadach@ece.umd.edu. This work was done in part while the author was visiting the Simons Institute for the Theory of Computing, supported by the Simons Foundation and by the DIMACS/Simons Collaboration in Cryptography through NSF grant #CNS-1523467.

†Dept. of Computer Science, George Mason University. Email: crypto@dovgordon.com. This work was done when the author was a research scientist at Applied Communication Sciences.

‡Dept. of Computer Science, Florida Atlantic University. Email: fenghao.liu@fau.edu. This work was done when the author was a postdoc at the University of Maryland.

§Dept. of Computer Science, Georgetown University. Email: adam@cs.georgetown.edu.

¶Dept. of Computer Science, Virginia Commonwealth University. Email: hszhou@vcu.edu.

Contents

1 Introduction

1.1 Background and Motivation

In recent years, researchers have uncovered a variety of ways to capture cryptographic keys through *side-channel attacks*: physical measurements, such as execution time, power consumption, and even sound waves generated by the processor. This has prompted cryptographers to build models for these attacks and to construct *leakage resilient* schemes that remain secure in the face of such attacks. Of course, if the adversary can leak the entire secret key, security becomes impossible, and so the *bounded leakage model* was introduced (cf. [2, 45, 38, 11]). Here, it is assumed that there is a fixed upper bound, L on the number of bits the attacker may leak, regardless of the parameters of the scheme, or, alternatively, it is assumed that the attacker is allowed to leak $L = \lambda \cdot |\mathsf{sk}|$ total number of bits, where the amount of leakage increases as the size of the secret key increases. Various works constructed public key encryption and signature schemes with optimal leakage rate of $\lambda = 1 - o(1)$, from specific assumptions (cf. [45, 11]). Hazay et al. [35] constructed a leakage resilient public key encryption scheme in this model, assuming only the existence of some standard public key encryption scheme; the tradeoff is that they tolerate a leakage rate of only $O(\log(\kappa)/|sk|)$, where $|sk|$ is the size of the secret key when using security parameter κ.

Surprisingly, it is possible to do better; an interesting strengthening of the model — the *continual leakage model*[1] — allows the adversary to request *unbounded* leakage. This model was introduced by Brakerski et al. [12] and Dodis et al. [21], who constructed continual-leakage resilient (CLR) public-key encryption and signature schemes. Intuitively, the CLR model divides the lifetime of the attack, which may be unbounded, into time periods and: (1) allows the adversary to obtain the output of a "bounded" leakage function in each time period, and (2) allows the secret key (but not the public key!) to be updated between time periods. So, while the adversary's leakage in each round is bounded, the total leakage is unbounded.

Note that the algorithm used by any CLR scheme to update the current secret key to the next one must be *randomized*, since otherwise the adversary can obtain some future secret key, bit-by-bit, via its leakage in each time period. While the CLR schemes of [12, 21] were able to tolerate a remarkable $1 - o(1)$ leakage rate (the ratio of the allowed number of bits leaked per time period to the length of the secret key) handling leakage *during the update procedure itself* — that is, produced as a function of the randomness used by the update algorithm as well as the current secret key — proved to be much more challenging. The first substantial progress on this problem of "leakage on key updates" was made by Lewko et al. [42], with their techniques being considerably refined and generalized by Dodis et al. [24]. In particular, they give encryption and signature schemes that are CLR with leakage on key updates tolerating a constant leakage rate, using "dual-system" techniques (cf. [47]) in bilinear groups.

1.2 Overview of Our Results

Our first main contribution is to show how to compile *any* public-key encryption or signature scheme that satisfies a slight strengthening of CLR (which we call "consecutive" CLR or 2CLR) *without* leakage on key updates to one that is CLR *with* leakage on key updates. Our compiler is based on a new connection we make between the problems of leakage on key updates and "sender-deniability" [13] for encryption schemes. In particular, our compiler uses program obfuscation — either indistinguishability obfuscation (iO) [5, 29] or the public-coin differing-inputs obfuscation [37][2] — and adapts and extends techniques recently developed by Sahai and Waters [46] to achieve sender-deniable encryption. This demonstrates the applicability of the

[1] Here "continual" refers to the fact that the total amount of leakage obtained by the adversary is unbounded. Additionally, the model is more accurately called the continual *memory* leakage model to contrast with schemes constructed under an assumption that "only computation leaks" [44].

[2] To the best of our knowledge, no impossibility results are known for public-coin differing-inputs obfuscation. Indeed, the impossibility results of Garg et al. [30] do not apply to this setting. In either case, current constructions rely on multilinear maps, whose first candidate construction was given by [28].

techniques of [46] to other seemingly unrelated contexts.[3] We then show that the existing CLR encryption scheme of Brakerski et al. [12] can be extended to meet the stronger notion of 2CLR that we require for our compiler. Additionally, we show all our results carry over to signatures as well. In particular, we show that 2CLR PKE implies 2CLR signatures (via the intermediate notion of CLR "one-way relations" of Dodis et al. [21]), and observe that our compiler also upgrades 2CLR signatures to ones that are CLR with leak on updates.

Our second main contributions concerns constructions of leakage-resilient public-key encryption directly from obfuscation. In particular, we show that the approach of Sahai and Waters to achieve public-key encryption from iO and punctured pseudorandom functions [46] can be extended to achieve leakage-resilience in the bounded-leakage model. Specifically, we achieve (1) leakage-resilient public key encryption tolerating L bits of leakage for any L from iO and one-way functions, (2) leakage-resilient public key encryption with optimal leakage rate of $1 - o(1)$ based on public-coin differing-inputs obfuscation and collision-resistant hash functions. (3) (consecutive) CLR public key encryption with constant (although not optimal, on the order of one over several hundred) leakage rate from differing inputs obfuscation (not public coin) and standard assumptions. Extending the construction from (2) to achieve continual leakage-resilience, without these additional assumptions, is an interesting open problem.

In summary, we provide a thorough study of the connection between program obfuscation and leakage resilience. We define a new notion of leakage-resilience (2CLR), and demonstrate new constructions of 2CLR secure encryption and signature schemes from program obfuscation. Also using program obfuscation, we construct a compiler that lifts 2CLR-secure schemes to CLR with leakage on updates; together with our new constructions, this provides a unified and modular method for constructing CLR with leakage on key updates. Under appropriate assumptions (namely, the ones used by Brakerski et al. [12] in their construction), this approach allows us to achieve a leakage rate of $1/4 - o(1)$, a large improvement over prior work, where the best leakage rate was $1/258 - o(1)$ [42]. Our result nearly matches the trivial upper-bound of $1/2 - o(1)$.[4] In the bounded leakage model, we show that it is possible to achieve optimal-rate leakage-resilient public key encryption from obfuscation and generic assumptions.

Finally, as we have mentioned above, Hazay et al. [35] constructed bounded leakage resilient public key encryption in this model from a far weaker generic assumption, albeit with a far worse leakage rate. In addition to offering a tradeoff between the strength of the assumption and the leakage rate, the value of our result in the bounded leakage model is that it provides direct insight into the connection between program obfuscation and leakage resilience. We are hopeful that our techniques might lead to future improvements in the continual-leakage models.

1.3 Details and Techniques

Part I: The Leak-on-Update Compiler. As described above, in the model of continual leakage-resilience (CLR) [12, 21] for public-key encryption or signature schemes, the secret key can be updated periodically (according to some algorithm Update) and the adversary can obtain bounded leakage between any two updates. Our compiler applies to schemes that satisfy a slight strengthening of CLR we call *consecutive* CLR, where the adversary can obtain bounded leakage as a *joint* function of any two consecutive keys. More formally, let $\mathsf{sk}_0, \mathsf{sk}_1, \mathsf{sk}_2, \ldots, \mathsf{sk}_t, \ldots$ be the secret keys at each time period, where $\mathsf{sk}_i = \mathsf{Update}(\mathsf{sk}_{i-1}, r_i)$, and each r_i denotes fresh random coins used at that round. For leakage functions $f_1, \ldots, f_t, \ldots$ (chosen adaptively by the adversary), consider the following two leakage models:

[3] We note that the techniques of [46] have been shown useful in adaptively secure two-party and multiparty computation [31, 14, 19] and "only computation leaks" (OCL) circuits without trusted hardware [20]. We note that this work precedes the work of [19].

[4] Unlike the case of CLR without leakage on key updates, observe that any scheme that is CLR with leakage on key updates can leak at most $1/2 \cdot |\mathsf{sk}|$-bits per time period, since otherwise the adversary can recover an entire secret key. As a consequence, the optimal leakage rate for a scheme that is CLR with leakage on key updates is at most $\frac{1/2 \cdot |\mathsf{sk}|}{|\mathsf{sk}| + |r_{up}|} < 1/2$, where $|\mathsf{sk}|$ is the secret key length and $|r_{up}|$ is the length of the randomness needed by the update algorithm.

(1) For **consecutive CLR (2CLR)**, the adversary obtains leakage

$$f_1(\mathsf{sk}_0, \mathsf{sk}_1), f_2(\mathsf{sk}_1, \mathsf{sk}_2), \ldots, f_t(\mathsf{sk}_{t-1}, \mathsf{sk}_t), \ldots \ .$$

(2) For **CLR with leakage on key updates**, the adversary obtains leakage

$$f_1(\mathsf{sk}_0, r_1), f_2(\mathsf{sk}_1, r_2), \ldots, f_t(\mathsf{sk}_{t-1}, r_t), \ldots \ .$$

Our compiler from 2CLR to CLR with leakage on key updates produces a slightly different Update algorithm for the compiled scheme depending on whether we assume indistinguishability-obfuscation (iO) [5, 29] or public-coin differing-inputs obfuscation [37]. In both cases, if we start with an underlying scheme that is consecutive two-key CLR while allowing μ-bits of leakage, then our compiled scheme is CLR with leakage on key updates with leakage rate

$$\frac{\mu}{|\mathsf{sk}| + |r_{up}|} \ ,$$

where $|r_{up}|$ is the length of the randomness required by Update. When using iO, we obtain $|r_{up}| = 6|\mathsf{sk}|$, where $|\mathsf{sk}|$ is the secret key length for the underlying 2CLR scheme, whereas using public-coin differing-input obfuscation we obtain $|r_{up}| = |\mathsf{sk}|$. Thus:

- Assuming iO, the compiled scheme is CLR with leakage on key updates with leakage rate $\frac{\mu}{7 \cdot |\mathsf{sk}|}$.
- Assuming public-coin differing-input obfuscation, the compiled scheme is CLR with leakage on key updates with leakage rate $\frac{\mu}{2 \cdot |\mathsf{sk}|}$.

Thus, if the underlying 2CLR scheme tolerates the optimal number of bits of leakage ($\approx 1/2 \cdot |\mathsf{sk}|$), then our resulting public-coin differing-inputs based scheme achieves leakage rate $1/4 - o(1)$.

Our compiler is obtained by adapting and extending the techniques developed by [46] to achieve sender-deniable PKE from any PKE scheme. In sender-deniable PKE, a sender, given a ciphertext and *any* message, is able to produce coins that make it appear that the ciphertext is an encryption of that message. Intuitively, the connection we make to leakage on key updates is that the simulator in the security proof faces a similar predicament to the coerced sender in the case of deniable encryption; it needs to come up with some randomness that "explains" a current secret key as the update of an old one. Our compiler makes any two such keys explainable in a way that is similar to how Sahai and Waters make any ciphertext and message explainable. Intuitively, this is done by "encoding" a secret key in the explained randomness in a special way that can be detected only by the (obfuscated) Update algorithm. Once detected, the Update algorithm outputs the encoded secret key, instead of running the normal procedure.

However, in our context, naïvely applying their techniques would result in the randomness required by our Update algorithm being very long, which, as described above, affects the leakage rate of our resulting CLR scheme with leakage on key updates in a crucial way (we would not even be able to get a constant leakage rate). We decrease the length of this randomness in two steps. First, we note that the sender-deniable encryption scheme of Sahai and Waters encrypts a message bit-by-bit and "explains" each message-bit individually. This appears to be necessary in their context in order to allow the adversary to choose its challenge messages *adaptively* depending on the public key. For our setting, this is not the case, since the secret key is chosen honestly (not by the adversary), so "non-adaptive" security is in fact sufficient in our context and we can "explain" a secret key all at once.This gets us to $|r_{up}| = 6 \cdot |\mathsf{sk}|$ and thus $1/14 - o(1)$ leakage rate assuming the underlying 2CLR scheme can tolerate the optimal leakage. Second, we observe that by switching assumptions from iO to the public-coin differing-inputs obfuscation we can replace some instances of sk in the explained randomness with its value under a collision-resistant hash, which gets us to $|r_{up}| = \mathsf{sk}$ and thus $1/4 - o(1)$ leakage rate in this case.

A natural question is whether the upper bound of $1/2 - o(1)$ leakage rate for CLR with leakage on key updates, can be attained via our techniques (if at all). We leave this as an intriguing open question, but note that the *only* way to do so would be to further decrease $|r_{up}|$ so that $|r_{up}| < |\mathsf{sk}|$.

Part II: Constructions against Two-key Consecutive Continual Leakage. We revisit the existing CLR public-key encryption scheme of [12] and show that a suitable modification of it achieves 2CLR[5] with optimal $1/4 - o(1)$ leakage rate[6], under the same assumption used by [12] to achieve optimal leakage rate in the basic CLR setting (namely the symmetric external Diffie-Hellman (SXDH) assumption in bilinear groups; smaller leakage rates can be obtained under weaker assumptions). Our main technical tool here is a new generalization of the Crooked Leftover Hash Lemma [26, 6] that generalizes the result of [12], which shows that "random subspaces are leakage resilient," showing that random subspaces are in fact resilient to "consecutive leakage." Our claim also leads to a simpler analysis of the scheme than appears in [12].

Finally, we also show (via techniques from learning theory) that 2CLR public-key encryption generically implies 2CLR one-way relations. Via a transformation of Dodis et al. [21], this then yields 2CLR signatures with the same leakage rate as the starting encryption scheme. Therefore, all the above results translate to the signature setting as well. We also show a direct approach to constructing 2CLR one-way relations following [21] based on the SXDH assumption in bilinear groups, although we are not able to achieve as good of a leakage rate this way (only $1/8 - o(1)$).

Part III: Exploring the relationship between (bounded and continual) leakage resilience and obfuscation. Note that, interestingly, even the strong notion of VBB obfuscation does not immediately lead to constructions of leakage resilient public-key encryption. In particular, if we replace the secret key of a public key encryption scheme with a VBB obfuscation of the decryption algorithm, it is not clear that we gain anything: E.g., the VBB obfuscation may output a circuit of size $|C|$, where only $\sqrt{|C|}$ number of the gates are "meaningful" and the remaining gates are simply "dummy" gates, in which case we cannot hope to get a leakage bound better than $L = \sqrt{|C|}$, and a leakage rate of $1/\sqrt{|C|}$. Nevertheless, we are able to show that the PKE scheme of Sahai and Waters (SW) [46], which is built from iO and "punctured pseudorandom functions (PRFs)," can naturally be made leakage resilient. To give some brief intuition, a ciphertext in our construction is of the form $(r, w, \mathsf{Ext}(\mathsf{PRF}(k; r), w) \oplus m)$, where Ext is a strong extractor, r and w are random values[7], and the PRF key k is embedded in obfuscated programs that are used in both encryption and decryption. In the security proof, we "puncture" the key k at the challenge point, t^*, and hardcode the mapping $t^* \to y$, where $y = \mathsf{PRF}(k; t^*)$, in order to preserve the input/output behavior. As in SW, we switch the mapping to $t^* \to y^*$ for a random y^* via security of the puncturable PRF. But now observe we have that the min-entropy of y^* is high even after leakage, so the output of the extractor is close to uniform. To achieve optimal leakage rate, we further modify the scheme to separate $t^* \to y^*$ from the obfuscated program and store only an encryption of $t^* \to y^*$ in the secret key.

Note that the last change lends itself to achieving (consecutive) CLR, since the secret key can be refreshed by re-randomizing the encryption. However, the information theoretic argument above about the entropy remaining in y^* no longer holds, since additional entropy is lost in every round, and, eventually, y^* might be recovered in full. To address this issue, we must prevent the attacker from directly leaking on y^* in each round. Instead of embedding an encryption of $t^* \to y^*$ in the secret key, we embed an encryption of a tuple $(s_i, \alpha_i, H(t^*)) \to y^*$ using a fresh s_i in each round i, subject to the constraint that $\alpha_i = \langle s_i, t^* \rangle$. In order to determine whether to output y^* on some input t, our obfuscated circuit decrypts and checks whether $H(t^*) = H(t) \wedge \langle s_i, t \rangle = \alpha_i$, where H is a collision resistant hash function. We rely on the following facts to ensure that y^* remains indistinguishable from random given the adversary's view: a) the adversary must form his leakage queries before learning t^*, b) very little information about t^* is contained in the secret key, and c) due to the previous facts, and since the inner-product is a good two-source extractor, $\langle s_i, t^* \rangle$ remains very close to uniform, even under the leakage. It follows that we can switch, even under leakage, to a random α^*, uncorrelated with s_i, t^*. Since it is now hard to find inputs satisfying $H(t^*) = H(t) \wedge \langle s_i, t \rangle = \alpha^*$, we can,

[5] Note that [12] also constructs such a signature scheme, but, as discussed below, such a signature scheme can in fact be generically obtained, and therefore for simplicity we do not consider their direct construction here.

[6] In the 2CLR model, the maximum amount of leakage is roughly $1/2 \cdot |\mathsf{sk}|$, so the optimal rate is roughly $\frac{1/2 \cdot |\mathsf{sk}|}{|\mathsf{sk}| + |\mathsf{sk}|} = 1/4$.

[7] Technically, we actually use pseudo-random value r, just as SW do. We omit this here to make the explanation a little more clear.

using security of the diO, *ignore* this conditional statement and replace y^*, with a 0 string in the secret key, while still using y^* in the challenge ciphertext.

In the above discussion above we omitted some additional technical challenges due to lack of space. Most notably, we also require that the encryption scheme used for encrypting the tuple in the secret key satisfies a notion of "diO-compatible RCCA-secure re-randomizability," which we introduce (see Section 5.3.2), and show that the "controlled malleable" RCCA-secure PKE due to Chase et al. [17] based on the decision linear assumption in bilinear groups schemes satisfies it, giving us a constant leakage rate for our (2)CLR scheme. For an in-depth technical overview and complete proof, see Section 5.3.

1.4 Related Work

Leakage resilient cryptography. We discuss various types of memory leakage attacks that have been studied in the literature. Memory attacks are a strong type of attack, where all secrets in memory are subject to leakage, whether or not they are actively being computed on. Memory leakage attacks are motivated by the cold-boot attack of Halderman et al. [34], who showed that for some time after power is shut down, partial data can be recovered from random access memory (DRAM and SRAM). Akavia et al. [2] introduced the model of bounded memory attacks, where arbitrary leakage on memory is allowed, as long as the output size of the leakage function is bounded. Additional models introduced by [16], [27] and [23] allow unbounded-length noisy leakage, unbounded-length leakage under restricted leakage functions, or unbounded-length hard-to-invert leakage, respectively. The works of [12] and [21] introduced the notion of "continual memory leakage" for public key primitives where the secret key is updated while the public key remains the same. This model allows bounded memory leakage between key refreshes. Finally, [12, 21, 42, 24] considered the model of continual memory leakage with leak on update, where leakage can occur while the secret key is being updated. In this work, we consider bounded memory attacks, continual memory leakage and continual memory leakage with leak on update.

There is a long line of constructions of leakage-resilient cryptographic primitives, including public key encryption that are leakage resilient (LR) against bounded memory attacks [2, 45]; public key encryption that is continual leakage resilient (CLR) without leak on update [12]; public key encryption that is CLR with leak on update [42]; digital signature schemes that are leakage resilient (LR) against bounded memory attacks [38]; digital signature schemes that are LR against bounded memory attacks on both secret key and random coins for signing [38, 11, 43]; digital signature schemes that are CLR without leak on update [21]; digital signature schemes that are CLR with leak on update [42].

Obfuscation and its applications. Since the breakthrough result of Garg et al. [29], demonstrating the first candidate of indistinguishability obfuscation (iO) for all circuits, a myriad of uses for iO in cryptography have been found. Among these results, the puncturing methodology by Sahai and Waters [46] has been found very useful. Related notions such as differing-inputs obfuscation (diO) [4] have been studied [9, 3, 37]. Please refer to [49, 48]for new constructions, applications, and limitations of obfuscations.

2 Definitions and Preliminaries

2.1 Security Definitions for Leakage Resilient Public Key Encryption

In this section, we first present the definitions of various leakage resilient PKE. The definitions in the first two subsections are from the literature. We present our new definition for consecutive continual leakage resilience (2CLR) in Subsection 3.1. We will present definitions for other primitives, such as one-way relations and signatures in Sections 6 and 7.

We present definitions for obfuscation and puncturable PRFs in Sections 2.2 and 2.3.

2.1.1 One-time Leakage Model

A public key encryption scheme PKE consists of three algorithms: $(\mathsf{PKE.Gen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$.

- $\mathsf{PKE.Gen}(1^\kappa) \to (\mathsf{pk}, \mathsf{sk})$. The key generation algorithm takes in the security parameter and outputs a public key pk and a secret key sk.
- $\mathsf{PKE.Enc}(\mathsf{pk}, m) \to c$. The encryption algorithm takes in a public key pk and a message m. It outputs a ciphertext c.
- $\mathsf{PKE.Dec}(\mathsf{sk}, c) \to m$. The decryption algorithm takes in a ciphertext c and a secret key sk. It outputs a message m.

Correctness. The PKE scheme satisfies correctness if $\mathsf{PKE.Dec}(\mathsf{sk}, c) = m$ with all but negligible probability whenever pk, sk is produced by $\mathsf{PKE.Gen}$ and c is produced by $\mathsf{PKE.Enc}(\mathsf{pk}, m)$.

Security. We define one-time leakage-resilient security for PKE schemes in terms of the following game between a challenger and an attacker (this extends the usual notion of semantic security to our leakage setting). We let κ denote the security parameter, and the parameter μ controls the amount of leakage allowed.

Setup Phase. The game begins with a setup phase. The challenger calls $\mathsf{PKE.Gen}(1^\kappa)$ to create the initial secret key sk and public key pk. It gives pk to the attacker. No leakage is allowed in this phase.

Query Phase. The attacker specifies an efficiently computable leakage function f, whose output is at most μ bits. The challenger returns $f(\mathsf{sk})$ to the attacker.

Challenge Phase. The attacker chooses two messages m_0, m_1 which it gives to the challenger. The challenger chooses a random bit $b \in \{0, 1\}$, encrypts m_b, and gives the resulting ciphertext to the attacker. The attacker then outputs a guess b' for b. The attacker wins the game if $b = b'$. We define the advantage of the attacker in this game as $|\frac{1}{2} - \Pr[b' = b]|$.

Definition 2.1 (One-time Leakage Resilience) *We say a Public Key Encryption scheme is μ-leakage resilient against one-time key leakage if any probabilistic polynomial time attacker only has a negligible advantage (negligible in κ) in the above game.*

2.1.2 Continual Leakage Model

In the continual leakage setting we require an additional algorithm $\mathsf{PKE.Update}$ which updates the secret key. Specifically, the update algorithm takes in a secret key sk_{i-1} and some randomness r_i, and produces a new secret key sk_i for the *same* public key. Thus, scheme PKE consists of four algorithms: $(\mathsf{PKE.Gen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec}, \mathsf{PKE.Update})$.

- $\mathsf{PKE.Gen}(1^\kappa) \to (\mathsf{pk}, \mathsf{sk}_0)$. The key generation algorithm takes in the security parameter and outputs a public key pk and a secret key sk_0.
- $\mathsf{PKE.Enc}(\mathsf{pk}, m) \to c$. The encryption algorithm takes in a public key pk and a message m. It outputs a ciphertext c.
- $\mathsf{PKE.Dec}(\mathsf{sk}_i, c) \to m$. The decryption algorithm takes in a ciphertext c and a secret key sk_i. It outputs a message m.
- $\mathsf{PKE.Update}(\mathsf{sk}_{i-1}) \to \mathsf{sk}_i$. The update algorithm takes in a secret key sk_{i-1} and produces a new secret key sk_i for the *same* public key. Here some randomness r_i is used in the update algorithm.

Correctness. The PKE scheme satisfies correctness if $\mathsf{PKE.Dec}(\mathsf{sk}_i, c) = m$ with all but negligible probability whenever pk, sk is produced by $\mathsf{PKE.Gen}$, sk_i is obtained by calles to $\mathsf{PKE.Update}$ on previously obtained secret keys (starting with sk_0), and c is produced by $\mathsf{PKE.Enc}(\mathsf{pk}, m)$.

Security. We define continual leakage-resilient security for PKE schemes in terms of the following game between a challenger and an attacker (this extends the usual notion of semantic security to our leakage setting). We let κ denote the security parameter, and the parameter μ controls the amount of leakage allowed.

Setup Phase. The game begins with a setup phase. The challenger calls $\mathsf{PKE.Gen}(1^\kappa)$ to create the initial secret key sk_0 and public key pk. It gives pk to the attacker. No leakage is allowed in this phase.

Query Phase. In this phase, the attacker launches a polynomial number of leakage queries. Each time, say in the i-th query, the attacker provides an efficiently computable leakage function f_i whose output is at most μ bits, and the challenger chooses randomness r_i, updates the secret key from sk_{i-1} to sk_i, and gives the attacker the leakage response ℓ_i. In the *regular continual leakage* model, the leakage attack is applied on a single secret key, and the leakage response $\ell_i = f_i(\mathsf{sk}_{i-1})$. In the *continual leak-on-update* model, the leakage attack is applied on the current secret key and the randomness used for updating the secret key, i.e., $\ell_i = f_i(\mathsf{sk}_{i-1}, r_i)$.

Challenge Phase. The attacker chooses two messages m_0, m_1 which it gives to the challenger. The challenger chooses a random bit $b \in \{0, 1\}$, encrypts m_b, and gives the resulting ciphertext to the attacker. The attacker then outputs a guess b' for b. The attacker wins the game if $b = b'$. We define the advantage of the attacker in this game as $|\frac{1}{2} - \Pr[b' = b]|$.

Definition 2.2 (Continual Leakage Resilience) *We say a Public-Key Encryption scheme is μ-CLR secure (or μ-CLR secure with leakage on key updates) if any* PPT *attacker only has a negligible advantage (negligible in κ) in the above game.*

Statistical Indistinguishability. The statistical distance between two random variables X, Y is defined by

$$\Delta(X, Y) = \frac{1}{2} \sum_x |\Pr[X = x] - \Pr[Y = x]|$$

We write $X \overset{\text{s}}{\approx} Y$ to denote that the statistical distance is negligible in the security parameter, and we say that X, Y are statistically indistinguishable.

2.2 Obfuscation

Indistinguishability Obfuscation. A uniform PPT machine iO is called an indistinguishable obfucastor [4, 5, 33, 29], for a circuit family $\{\mathcal{C}_\kappa\}$, if the following conditions hold:

- (Correctness) For all $\kappa \in \mathbb{N}$, for all $C \in \mathcal{C}_\kappa$, for all inputs x, we have

$$\Pr\left[C'(x) = C(x) \ : \ C' \leftarrow \mathsf{iO}(\kappa, C)\right] = 1$$

- For any uniform or non-uniform PPT distinguisher D, for all security parameter $\kappa \in \mathbb{N}$, for all pairs of circuits $C_0, C_1 \in \mathcal{C}_\kappa$ such that $C_0(x) = C_1(x)$ for all inputs x, then

$$|\Pr\left[D(\mathsf{iO}(\kappa, C_0)) = 1\right] - \Pr\left[D(\mathsf{iO}(\kappa, C_1)) = 1\right]| \leq \mathsf{negl}(\kappa)$$

For simplicity, when the security parameter κ is clear, we write $\mathsf{iO}(C)$ in short.

Public-Coin Differing-inputs Obfuscation for Circuits. Barak et al. [4, 5] defined the notion of differing-inputs obfuscation, which was later re-formulated in the works of Ananth et al. and Boyle et. al [3, 9]. In our work, we use a weaker notion known as *public-coin differing inputs obfuscation*, due to Ishai et al. [37]. To the best of our knowledge, unlike the case of differing-inputs obfuscation, there are no impossibility results for public-coin differing-inputs obfuscation. Below, we closely follow the definitions presented in [37].

Definition 2.3 (Public-Coin Differing-Inputs Sampler for Circuits) *An efficient non-uniform sampling algorithm* $\mathsf{Samp} = \{\mathsf{Samp}_\kappa\}$ *is called a* public-coin differing-inputs sampler *for the parameterized collection of circuits* $\mathcal{C} = \{\mathcal{C}_\kappa\}$ *if the output of* Samp_κ *is distributed over* $\mathcal{C}_\kappa \times \mathcal{C}_\kappa$ *and for every efficient non-uniform algorithm* $\mathcal{A} = \{\mathcal{A}_\kappa\}$ *there exists negligible function* negl *such that for all* $\kappa \in \mathbb{N}$*:*

$$\Pr_r[C_0(x) \neq C_1(x) : (C_0, C_1) \leftarrow \mathsf{Samp}_\kappa(r), x \leftarrow \mathcal{A}_\kappa(r)] \leq \mathsf{negl}(\kappa).$$

Note that in the above definition the sampler and attacker circuits both receive the same random coins as input.

Definition 2.4 (Public-Coin Differing-inputs Obfuscator for Circuits) *A uniform* PPT *machine* diO *is called a Public-Coin Differing-inputs Obfuscator for the parameterized collection of circuits* $\mathcal{C} = \{\mathcal{C}_\kappa\}$ *if the following conditions are satisfied:*

- *(Correctness): For all security parameter* κ*, all* $C \in \mathcal{C}_\kappa$*, all inputs* x*, we have*

$$\Pr[C'(x) = C(x) : C' \leftarrow \mathsf{diO}(\kappa, C)] = 1.$$

- *(Polynomial slowdown): There exists a universal polynomial* p *such that for any circuit* C*, we have* $|C'| \leq p(|C|)$ *for all* $C' = \mathsf{diO}(\kappa, C)$ *under all random coins.*

- *(Differing-inputs): For every public-coin differing-inputs samplers* $\mathsf{Samp} = \{\mathsf{Samp}_\kappa\}$ *for the collection* $\mathcal{C}$*, for every (not necessarily uniform)* PPT *distinguisher* D*, there exists a negligible function* negl *such that for all security parameters* κ*:*

$$\left| \begin{array}{r} \Pr[D_\kappa(r, C') = 1 : (C_0, C_1) \leftarrow \mathsf{Samp}_\kappa(r), C' \leftarrow \mathsf{diO}(\kappa, C_0)] - \\ \Pr[D_\kappa(r, C') = 1 : (C_0, C_1) \leftarrow \mathsf{Samp}_\kappa(r), C' \leftarrow \mathsf{diO}(\kappa, C_1)] \end{array} \right| \leq \mathsf{negl}(\kappa),$$

  where the probability is taken over r *and the coins of* diO*.*

2.3 Puncturable Pseudorandom Functions.

Puncturable family of PRFs are a special case of constrained PRFs [8, 10, 40], where the PRF is defined on all input strings except for a set of size polynomial in the security parameter. Below we recall their definition, as given by [46].

A puncturable family of PRFs PRF is defined by a tuple of efficient algorithms $(\mathsf{Gen}, \mathsf{Eval}, \mathsf{Punct})$ and a pair of polynomials $n()$ and $m()$:

- **Key Generation** $\mathsf{Gen}(1^\kappa)$ is a PPT algorithm that takes as input the security parameter κ and outputs a PRF key K.

- **Punctured Key Generation** $\mathsf{Punct}(K, S)$ is a PPT algorithm that takes as input a PRF key K, a set $S \subset \{0,1\}^{n(\kappa)}$ and outputs a punctured key K_S.

- **Evaluation** $\mathsf{Eval}(K, x)$ is a deterministic algorithm that takes as input a key K (punctured key or PRF key), a string $x \in \{0,1\}^{n(\kappa)}$ and outputs $y \in \{0,1\}^{m(\kappa)}$

Definition 2.5 *A family of PRFs* $(\mathsf{Gen}, \mathsf{Eval}, \mathsf{Punct})$ *is puncturable if it satisfies the following properties*

- **Functionality preserved under puncturing.** *Let* $K \leftarrow \mathsf{Gen}(1^\kappa)$ *and* $K_S \leftarrow \mathsf{Punct}(K, S)$. *Then for all* $x \notin S$, $\mathsf{Eval}(K, x) = \mathsf{Eval}(K_S, x)$.
- **Pseudorandom at punctured points.** *For every* PPT *adversary* $(\mathcal{A}_1, \mathcal{A}_2)$ *such that* $\mathcal{A}_1()$ *outputs a set* $S \subset \{0,1\}^{n(\kappa)}$ *and* $x \in S$, *consider an experiment* $K \leftarrow \mathsf{Gen}(1^\kappa)$ *and* $K_S \leftarrow \mathsf{Punct}(K, S)$. *Then*

$$\left|\Pr[\mathcal{A}_2(K_S, x, \mathsf{Eval}(K, x)) = 1] - \Pr[\mathcal{A}_2(K_S, x, U_{m(\kappa)}) = 1]\right| \leq \mathsf{negl}(\kappa)$$

where $U_{m(\kappa)}$ *denotes the the uniform distribution over* $m(\kappa)$ *bits.*

Theorem 2.6 ([32, 8, 10, 40]) *If one-way functions exist, then for all polynomial* $n()$ *and* $m()$, *there exists a puncturable PRF family that maps* $n()$ *bits to* $m()$ *bits.*

Next we consider families of PRFs that are with high probability injective:

Definition 2.7 *A statistically injective (puncturable) PRF family with failure probability* $\epsilon()$ *is a family of (puncturable) PRFs such that with probability* $1 - \epsilon()$ *over the random choice of key* $K \leftarrow \mathsf{Gen}(1^\kappa)$, *we have that* $\mathsf{Eval}(K, \cdot)$ *is injective.*

If the failure probability function $\epsilon()$ *is not specified, then* $\epsilon()$ *is a negligible function.*

Theorem 2.8 ([46]) *If one-way functions exist, then for all efficiently computable functions* $n(\kappa)$, $m(\kappa)$, *and* $e(\kappa)$ *such that* $m(\kappa) > 2n(\kappa) + e(\kappa)$ *here exists a puncturable statistically injective PRF family with failure probability* $2^{-e(\kappa)}$ *that maps* $n(\kappa)$ *bits to* $m(\kappa)$ *bits.*

Finally, we consider PRFs that are also (strong) extractors over their inputs:

Definition 2.9 *An extracting (puncturable) PRF family with error* $\epsilon()$ *for min-entropy* $k(\kappa)$ *is a family of (puncturable) PRFs mapping* $n(\kappa)$ *bits to* $m(\kappa)$ *bits such that for all* κ, *if* X *is any distribution over* $n(\kappa)$ *bits with min-entropy greater than* $k(\kappa)$ *then the statistical distance between* $(K \leftarrow \mathsf{Gen}(1^\kappa), \mathsf{Eval}(K, X))$ *and* $(K \leftarrow \mathsf{Gen}(1^\kappa), U_{m(\kappa)})$ *is at most* $\epsilon()$, *where* U_ℓ *denotes the uniform distribution over* ℓ *bit strings.*

Theorem 2.10 ([46]) *If one-way functions exist, then for all efficiently computable functions* $n(\kappa)$, $m(\kappa)$, $k(\kappa)$ *and* $e(\kappa)$ *such that* $n(\kappa) > k(\kappa) > m(\kappa) + 2e(\kappa) + 2$ *there exists an extracting puncturable PRF family that maps* $n(\kappa)$ *bits to* $m(\kappa)$ *bits with error* $2^{-e(\kappa)}$ *for min-entropy* $k(\kappa)$

For ease of presentation, for a puncturable family of PRFs PRF, we often write $\mathsf{PRF}(K, x)$ to represent $\mathsf{PRF}.\mathsf{Eval}(K, x)$.

3 Compiler from 2CLR to Leakage on Key Updates

In this section, we present a compiler that upgrades any scheme for public key encryption (PKE), digital signature (SIG), or one-way relation (OWR) that is consecutive two-key leakage resilient, into one that is secure against leak on update. We first introduce a notion of *explainable update transformation*, which is a generalization of the idea of universal deniable encryption by Sahai and Waters [46]. We show how to use such a transformation to upgrade a scheme (PKE, SIG, or OWR) that is secure in the consecutive two-key leakage model to one that is secure in the leak-on-update model (Section 3.2). Finally, we show two instantiations of the explainable update transformation: one based on indistinguishability obfuscation, and the other on differing-inputs obfuscation (Section 3.3). For clarity of exposition, the following sections will focus on constructions of PKE, but we remark that the same results can be translated to SIG and OWR.

3.1 Consecutive Continual Leakage Resilience (2CLR)

In this section, we present a new notion of *consecutive continual leakage resilience* for public-key encryption (PKE). We remark that this notion can be easily extended to different cases, such as signatures or leakage resilient one-way relations [21]. For simplicity and concreteness, we only present the PKE version. Let κ denote the security parameter, and μ be the leakage bound between two updates. Let $\mathsf{PKE} = \{\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Update}\}$ be an encryption scheme with update.

Setup Phase. The game begins with a setup phase. The challenger calls $\mathsf{PKE.Gen}(1^\kappa)$ to create the initial secret key sk_0 and public key pk. It gives pk to the attacker. No leakage is allowed in this phase.

Query Phase. The attacker specifies an efficiently computable leakage function f_1, whose output is at most μ bits. The challenger updates the secret key (changing it from sk_0 to sk_1), and then gives the attacker $f_1(\mathsf{sk}_0, \mathsf{sk}_1)$. The attacker then repeats this a polynomial number of times, each time supplying an efficiently computable leakage function f_i whose output is at most μ bits. Each time, the challenger updates the secret key from sk_{i-1} to sk_i according to $\mathsf{Update}(\cdot)$, and gives the attacker $f_i(\mathsf{sk}_{i-1}, \mathsf{sk}_i)$.

Challenge Phase. The attacker chooses two messages m_0, m_1 which it gives to the challenger. The challenger chooses a random bit $b \in \{0, 1\}$, encrypts m_b, and gives the resulting ciphertext to the attacker. The attacker then outputs a guess b' for b. The attacker wins the game if $b = b'$. We define the advantage of the attacker in this game as $|\frac{1}{2} - \Pr[b' = b]|$.

Definition 3.1 (Continual Consecutive Leakage Resilience) *We say a public-key encryption scheme is μ-leakage resilient against consecutive continual leakage (or μ-2CLR) if any probabilistic polynomial time attacker only has a negligible advantage (negligible in κ) in the above game.*

3.2 Explainable Key-Update Transformation

Now we introduce a notion of *explainable key-update transformation*, and show how it can be used to upgrade security of a PKE scheme from 2CLR to CLR with leakage on key updates. Informally, an encryption scheme has an "explainable" update procedure if given both sk_{i-1} and $\mathsf{sk}_i = \mathsf{Update}(\mathsf{sk}_{i-1}, r_i)$, there is an efficient way to come up with some explained random coins $\hat{r}_i$ such that no adversary can distinguish the real coins r_i from the explained coins $\hat{r}_i$. Intuitively, this gives a way to handle leakage on random coins given just leakage on two consecutive keys.

We start with any encryption scheme PKE that has some key update procedure, and we introduce a transformation that produces a scheme PKE' with an *explainable* key update procedure.

Definition 3.2 (Explainable Key Update Transformation) *Let $\mathsf{PKE} = \mathsf{PKE}.\{\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Update}\}$ be an encryption scheme with key update. An explainable key update transformation for PKE is a* PPT *algorithm $\mathsf{TransformGen}$ that takes input security parameter 1^κ, an update circuit C_{Update} (that implements the key update algorithm $\mathsf{PKE.Update}(1^\kappa, \cdot; \cdot)$), a public key pk of PKE, and outputs two programs $\mathcal{P}_{\mathsf{update}}, \mathcal{P}_{\mathsf{explain}}$ with the following syntax:*

Let $(\mathsf{pk}, \mathsf{sk})$ be a pair of public and secret keys of the encryption scheme

- *$\mathcal{P}_{\mathsf{update}}$ takes inputs sk, random coins r, and $\mathcal{P}_{\mathsf{update}}(\mathsf{sk}; r)$ outputs a updated secret key sk';*
- *$\mathcal{P}_{\mathsf{explain}}$ takes inputs $(\mathsf{sk}, \mathsf{sk}')$, random coins $\bar{v}$, and $\mathcal{P}_{\mathsf{explain}}(\mathsf{sk}, \mathsf{sk}'; \bar{v})$ outputs a string r.*

Given a public key pk, we define $\Pi_{\mathsf{pk}} = \bigcup_{j=0}^{\mathrm{poly}(\kappa)} \Pi_j$, where $\Pi_0 = \{\mathsf{sk} : (\mathsf{pk}, \mathsf{sk}) \in \mathsf{PKE.Gen}\}$, $\Pi_i = \{\mathsf{sk} : \exists \mathsf{sk}' \in \Pi_{i-1}, \mathsf{sk} \in \mathsf{Update}(\mathsf{sk}')\}$ for $i = 1, 2, \ldots, \mathrm{poly}(\kappa)$. In words, Π_{pk} is the set of all secret keys sk such that either $(\mathsf{pk}, \mathsf{sk})$ is in the support of $\mathsf{PKE.Gen}$ or sk can be obtained by the update procedure Update (up to polynomially many times) with an initial $(\mathsf{pk}, \mathsf{sk}') \in \mathsf{PKE.Gen}$.

We say the transformation is secure if:

(a) For any pk*, all* $\mathsf{sk} \in \Pi_{\mathsf{pk}}$*, any* $\mathcal{P}_{\mathsf{update}} \in \mathsf{TransformGen}(1^\kappa, \mathsf{PKE.Update}, \mathsf{pk})$*, the following two distributions are statistically close:* $\{\mathcal{P}_{\mathsf{update}}(\mathsf{sk})\} \approx \{\mathsf{PKE.Update}(\mathsf{sk})\}$*. Note that the circuit* $\mathcal{P}_{\mathsf{update}}$ *and the update algorithm* $\mathsf{PKE.Update}$ *might have different spaces for random coins, but the distributions can still be statistically close.*

(b) For any public key pk *and secret key* $\mathsf{sk} \in \Pi_{\mathsf{pk}}$*, the following two distributions are computationally indistinguishable:*

$$\{(\mathcal{P}_{\mathsf{update}}, \mathcal{P}_{\mathsf{explain}}, \mathsf{pk}, \mathsf{sk}, u)\} \approx \{(\mathcal{P}_{\mathsf{update}}, \mathcal{P}_{\mathsf{explain}}, \mathsf{pk}, \mathsf{sk}, e)\},$$

where $(\mathcal{P}_{\mathsf{update}}, \mathcal{P}_{\mathsf{explain}}) \leftarrow \mathsf{TransformGen}(1^\kappa, \mathsf{PKE.Update}, \mathsf{pk})$*,* $u \leftarrow U_{\mathrm{poly}(\kappa)}$*,* $\mathsf{sk}' = \mathcal{P}_{\mathsf{update}}(\mathsf{sk}; u)$*,* $e \leftarrow \mathcal{P}_{\mathsf{explain}}(\mathsf{sk}, \mathsf{sk}')$*, and* $U_{\mathrm{poly}(\kappa)}$ *denotes the uniform distribution over a polynomial number of bits.*

Let $\mathsf{PKE} = \mathsf{PKE}.\{\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Update}\}$ be a public key encryption scheme and $\mathsf{TransformGen}$ be an explainable key update transformation for PKE as above. We define the following transformed scheme $\mathsf{PKE}' = \mathsf{PKE}'.\{\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Update}\}$ as follows:

- $\mathsf{PKE}'.\mathsf{Gen}(1^\kappa)$: compute $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PKE.Gen}(1^\kappa)$.
  Then compute $(\mathcal{P}_{\mathsf{update}}, \mathcal{P}_{\mathsf{explain}}) \leftarrow \mathsf{TransformGen}(1^\kappa, \mathsf{PKE.Update}, \mathsf{pk})$.
  Finally, output $\mathsf{pk}' = (\mathsf{pk}, \mathcal{P}_{\mathsf{update}}, \mathcal{P}_{\mathsf{explain}})$ and $\mathsf{sk}' = \mathsf{sk}$.
- $\mathsf{PKE}'.\mathsf{Enc}(\mathsf{pk}', m)$: parse $\mathsf{pk}' = (\mathsf{pk}, \mathcal{P}_{\mathsf{update}}, \mathcal{P}_{\mathsf{explain}})$. Then output $c \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, m)$.
- $\mathsf{PKE}'.\mathsf{Dec}(\mathsf{sk}', c)$: output $m = \mathsf{PKE.Dec}(\mathsf{sk}', c)$.
- $\mathsf{PKE}'.\mathsf{Update}(\mathsf{sk}')$: sample $\mathsf{sk}'' \leftarrow \mathcal{P}_{\mathsf{update}}(\mathsf{sk}')$ and overwrite the old key, i.e. $\mathsf{sk}' := \mathsf{sk}''$.

Then we are able to show the following theorem for the upgraded scheme PKE'.

Theorem 3.3 *Let* $\mathsf{PKE} = \mathsf{PKE}.\{\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Update}\}$ *be a public key encryption scheme that is* μ*-2CLR (without leakage on update), and* $\mathsf{TransformGen}$ *a secure explainable key update transformation for* PKE*. Then the transformed scheme* $\mathsf{PKE}' = \mathsf{PKE}'.\{\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Update}\}$ *described above is* μ*-CLR with leakage on key updates.*

Proof:

Assume towards contradiction that there is a PPT adversary $\mathcal{A}$ and a non-negligible $\epsilon(\cdot)$ such that for infinitely many values of κ, $\mathsf{Adv}_{\mathcal{A},\mathsf{PKE}'} \geq \epsilon(\kappa)$ in the leak-on-update model. Then we show that there exists $\mathcal{B}$ that breaks the security of the underlying PKE (in the consecutive two-key leakage model) with probability $\epsilon(\kappa) - \mathsf{negl}(\kappa)$. This is a contradiction.

For notionally simplicity, we will use $\mathsf{Adv}_{\mathcal{A},\mathsf{PKE}'}$ to denote the advantage of the adversary $\mathcal{A}$ attacking the scheme PKE' (according to leak-on-update attacks), and $\mathsf{Adv}_{\mathcal{B},\mathsf{PKE}}$ to denote the advantage of the adversary $\mathcal{B}$ attacking the scheme PKE (according to consecutive two-key leakage attacks).

We define $\mathcal{B}$ in the following way: $\mathcal{B}$ internally instantiates $\mathcal{A}$ and participates externally in a continual consecutive two-key leakage experiment on public key encryption scheme PKE'. Specifically, $\mathcal{B}$ does the following:

- Upon receiving pk^* externally, $\mathcal{B}$ runs
  $(\mathcal{P}_{\mathsf{update}}, \mathcal{P}_{\mathsf{explain}}) \leftarrow \mathsf{TransformGen}(1^\kappa, \mathsf{PKE.Update}, \mathsf{pk}^*)$. Note that by the properties of the transformation, this can be done given only pk^*. $\mathcal{B}$ sets $\mathsf{pk}' = (\mathsf{pk}^*, \mathcal{P}_{\mathsf{update}}, \mathcal{P}_{\mathsf{explain}})$ to be the public key for the PKE' scheme and forwards pk' to $\mathcal{A}$.
- When $\mathcal{A}$ asks for a leakage query $f(\mathsf{sk}'_{i-1}, r_i)$, $\mathcal{B}$ asks for the following leakage query on $(\mathsf{sk}_{i-1}, \mathsf{sk}_i)$: $f'(\mathsf{sk}_{i-1}, \mathsf{sk}_i) = f(\mathsf{sk}_{i-1}, \mathcal{P}_{\mathsf{explain}}(\mathsf{sk}_{i-1}, \mathsf{sk}_i))$ and forwards the response to $\mathcal{A}$. Note that the output lengths of f and f' are the same.

- At some point $\mathcal{A}$ submits m_0, m_1 and $\mathcal{B}$ forwards them to its external experiment.
- Upon receiving the challenge ciphertext c^*, $\mathcal{B}$ forwards it to $\mathcal{A}$ and outputs whatever $\mathcal{A}$ outputs.

Now we would like to analyze the advantage of $\mathcal{B}$. It is easy to see that $\mathcal{B}$ has the same advantage as $\mathcal{A}$, however there is a subtlety such that $\mathcal{A}$ does not necessarily have advantage $\epsilon(\kappa)$: the simulation of leakage queries provided by $\mathcal{B}$ is not identical to the distribution in the real game that $\mathcal{A}$ would expect. Recall that in the security experiment of the scheme PKE', the secret keys are updated according to $\mathcal{P}_{\mathsf{update}}$. In the above experiment (where $\mathcal{B}$ set up), the secret keys were updated using the Update externally, and the random coins were simulated by the $\mathcal{P}_{\mathsf{explain}}$ algorithm.

Our goal is to show that actually $\mathcal{A}$ has essentially the same advantage in this modified experiment as in the original experiment. We show this by the following lemma:

Lemma 1 *For any polynomial n, the following two distributions are computationally indistinguishable.*

$$D_1 \equiv (\mathcal{P}_{\mathsf{update}}, \mathcal{P}_{\mathsf{explain}}, \mathsf{pk}, \mathsf{sk}_0, r_1, \mathsf{sk}_1, \ldots, \mathsf{sk}_{n-1}, r_n, \mathsf{sk}_n) \approx$$
$$D_2 \equiv (\mathcal{P}_{\mathsf{update}}, \mathcal{P}_{\mathsf{explain}}, \mathsf{pk}, \mathsf{sk}_0, \widehat{r}_1, \widehat{\mathsf{sk}}_1, \ldots, \widehat{\mathsf{sk}}_{n-1}, \widehat{r}_n, \widehat{\mathsf{sk}}_n),$$

where the initial $\mathsf{pk}, \mathsf{sk}_0$ and $\mathsf{TransformGen}(1^\kappa, \mathsf{pk})$ are sampled identically in both experiment; in D_1 $\mathsf{sk}_{i+1} = \mathcal{P}_{\mathsf{update}}(\mathsf{sk}_i; r_{i+1})$, and r_{i+1}'s are uniformly random; in D_2, $\widehat{\mathsf{sk}}_{i+1} \leftarrow \mathsf{Update}(\widehat{\mathsf{sk}}_i)$, $\widehat{r}_{i+1} \leftarrow \mathcal{P}_{\mathsf{explain}}(\widehat{\mathsf{sk}}_i, \widehat{\mathsf{sk}}_{i+1})$. (Note $\widehat{\mathsf{sk}}_0 = \mathsf{sk}_0$).

Proof: To show the lemma, we consider the following hybrids: for $i \in [n]$ define

$$H^{(i)} = (\mathcal{P}_{\mathsf{update}}, \mathcal{P}_{\mathsf{explain}}, \mathsf{pk}, \mathsf{sk}_0, \widehat{r}_1, \widehat{\mathsf{sk}}_1, \ldots, \widehat{\mathsf{sk}}_{i-1}, r_i, \mathsf{sk}_i, r_{i+1}, \mathsf{sk}_{i+1}, r_{i+2}, \ldots, \mathsf{sk}_n),$$

where the experiment is identical to D_2 for up to $\widehat{\mathsf{sk}}_{i-1}$. Then it samples a uniformly random r_i, sets $\mathsf{sk}_i = \mathcal{P}_{\mathsf{update}}(\widehat{\mathsf{sk}}_{i-1}; r_i)$, and proceeds as D_1.

$$H^{(i.5)} = (\mathcal{P}_{\mathsf{update}}, \mathcal{P}_{\mathsf{explain}}, \mathsf{pk}, \mathsf{sk}_0, \widehat{r}_1, \widehat{\mathsf{sk}}_1, \ldots, \widehat{\mathsf{sk}}_{i-1}, \widehat{r}_i, \mathsf{sk}_i, r_{i+1}, \mathsf{sk}_{i+1}, r_{i+2}, \ldots, \mathsf{sk}_n),$$

where the experiment is identical to $H^{(i)}$ for up to $\widehat{\mathsf{sk}}_{i-1}$, and then it samples $\mathsf{sk}_i \leftarrow \mathcal{P}_{\mathsf{update}}(\widehat{\mathsf{sk}}_{i-1})$, and $\widehat{r}_i \leftarrow \mathcal{P}_{\mathsf{explain}}(\widehat{\mathsf{sk}}_{i-1}, \mathsf{sk}_i)$. The experiment is identical to D_1 for the rest.

Then we establish the following lemmas, and the lemma follows directly.

Lemma 2 *For $i \in [n-1]$, $H^{(i.5)}$ is statistically close to $H^{(i+1)}$.*

Lemma 3 *For $i \in [n]$, $H^{(i)}$ is computationally indistinguishable from $H^{(i.5)}$.*

This first lemma follows directly from the property (a) of Definition 3.2. We now prove Lemma 3.

Proof: Suppose there exists a (polysized) distinguisher $\mathcal{D}$ that distinguishes $H^{(i)}$ from $H^{(i.5)}$ with non-negligible probability, then there exist $\mathsf{pk}^*, \mathsf{sk}^*$, and another $\mathcal{D}'$ that can break the property (b).

From the definition of the experiments, we know that $\mathcal{P}_{\mathsf{update}}, \mathcal{P}_{\mathsf{explain}}$ are independent of the public key and the first i secret keys, i.e. $\vec{p} = (\mathsf{pk}, \mathsf{sk}_0, \widehat{\mathsf{sk}}_1, \ldots, \widehat{\mathsf{sk}}_{i-1})$. By an average argument, there exists a fixed

$$\vec{p}^* = (\mathsf{pk}^*, \mathsf{sk}_0^*, \widehat{\mathsf{sk}}_1^*, \ldots, \widehat{\mathsf{sk}}_{i-1}^*)$$

such that $\mathcal{D}$ can distinguish $H^{(i)}$ from $H^{(i.5)}$ conditioned on $\vec{p}^*$ with non-negligible probability (the probability is over the randomness of the rest experiment). Then we are going to argue that there exist a polysized distinguisher $\mathcal{D}'$, a key pair $\mathsf{pk}', \mathsf{sk}'$ such that $\mathcal{D}'$ can distinguish $(\mathcal{P}_{\mathsf{update}}, \mathcal{P}_{\mathsf{explain}}, \mathsf{pk}', \mathsf{sk}', u)$ from $(\mathcal{P}_{\mathsf{update}}, \mathcal{P}_{\mathsf{explain}}, \mathsf{pk}', \mathsf{sk}', e)$ where u is from the uniform distribution, $\mathsf{sk}'' = \mathcal{P}_{\mathsf{update}}(\mathsf{sk}'; u)$, and $e \leftarrow \mathcal{P}_{\mathsf{explain}}(\mathsf{sk}', \mathsf{sk}'')$.

Let $\mathsf{pk}' = \mathsf{pk}^*$, $\mathsf{sk}' = \widehat{\mathsf{sk}}_{i-1}^*$, and we define $\mathcal{D}'$ (with the prefix $\vec{p}^*$ hardwired) who on the challenge input $(\mathcal{P}_{\mathsf{update}}, \mathcal{P}_{\mathsf{explain}}, \mathsf{pk}', \mathsf{sk}', z)$ does the following:

- For $j \in [i-1]$, $\mathcal{D}'$ samples $\widehat{r}_j = \mathcal{P}_{\mathsf{explain}}(\mathsf{sk}^*_{j-1}, \mathsf{sk}^*_j)$.
- Set $\mathsf{sk}_{i-1} = \mathsf{sk}'$ and $r_i = z$, $\mathsf{sk}_i = \mathcal{P}_{\mathsf{update}}(\mathsf{sk}_{i-1}, z)$.
- For $j \geq i+1$, $\mathcal{D}'$ samples r_j from the uniform distribution and sets $\mathsf{sk}_j = \mathcal{P}_{\mathsf{update}}(\mathsf{sk}_{j-1}; r_j)$.
- Finally, $\mathcal{D}'$ outputs $\mathcal{D}(\mathcal{P}_{\mathsf{update}}, \mathcal{P}_{\mathsf{explain}}, \mathsf{pk}', \mathsf{sk}^*_0, \widehat{r}_1, \mathsf{sk}^*_1, \ldots, \mathsf{sk}_{i-1}, r_i, \mathsf{sk}_i, r_{i+1}, \ldots, \mathsf{sk}_n)$.

Clearly, if the challenge z was sampled according to uniformly random (as u), then $\mathcal{D}'$ will output according to $\mathcal{D}(H^{(i)}|_{\vec{p}*})$. On the other hand, suppose it was sampled according to $\mathcal{P}_{\mathsf{explain}}$ (as e), then $\mathcal{D}'$ will output according to $\mathcal{D}(H^{i.5}|_{\vec{p}^*})$. This completes the proof of the lemma. ∎

∎

Remark. The non-uniform argument above is not necessary. We present in this way for simplicity. The uniform reduction can be obtained using a standard Markov type argument, which we omit here.

Now, we are ready to analyze the advantage of $\mathcal{B}$ (and $\mathcal{A}$). Denote $\mathsf{Adv}_{\mathcal{A},\mathsf{PKE}';D}$ as the advantage of $\mathcal{A}$ in the experiment where the leakage queries are answered according to the distribution D. By assumption, we know that $\mathsf{Adv}_{\mathcal{A},\mathsf{PKE}';D_1} = \epsilon(\kappa)$, and by definition the leakage queries are answered according to D_1. By the above lemma, we know that $|\mathsf{Adv}_{\mathcal{A},\mathsf{PKE}';D_1} - \mathsf{Adv}_{\mathcal{A},\mathsf{PKE}';D_2}| \leq \mathsf{negl}(\kappa)$, otherwise D_1 and D_2 are distinguishable. Thus, we know $\mathsf{Adv}_{\mathcal{A},\mathsf{PKE}';D_2} \geq \epsilon(\kappa) - \mathsf{negl}(\kappa)$. It is not hard to see that $\mathsf{Adv}_{\mathcal{B},\mathsf{PKE}} = \mathsf{Adv}_{\mathcal{A},\mathsf{PKE}';D_2}$, since $\mathcal{B}$ answers $\mathcal{A}$'s the leakage queries exactly according the distribution D_2. Thus, $\mathsf{Adv}_{\mathcal{B},\mathsf{PKE}} \geq \epsilon(\kappa) - \mathsf{negl}(\kappa)$, which is a contradiction. This completes the proof of the theorem. ∎

3.3 Instantiations via Obfuscation

In this section, we show how to build an explainable key update transformation from program obfuscation. Our best parameters are achieved using public-coin differing-inputs obfuscation [37] (rather than the weaker indistinguishability obfuscation (iO) [5, 29]), so we present this version here.

Let $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Update})$ be a public-key encryption scheme (or a signature scheme with algorithms $\mathsf{Verify}, \mathsf{Sign}$) with key-update, and diO be a public-coin differing-inputs obfuscator (for some class defined later). Let κ be a security parameter. Let L_{sk} be the length of secret keys in PKE and L_r be the length of randomness used by Update. For ease of notation, we suppress the dependence of these lengths on κ. We note that in the 2CLR case, it is without loss of generality to assume $L_r << L_{\mathsf{sk}}$, because we can always use pseudorandom coins (e.g. the output of a PRG) to do the update. Since only the two consecutive keys are leaked (not the randomness, e.g. the seed to the PRG), the update with the pseudorandom coins remains secure, assuming the PRG is secure.

Let $\mathcal{H}$ be a family of public-coin collision resistant hash functions, as well as a family of $(2\kappa, \epsilon)$-good unseeded extractors[8], mapping $2L_{\mathsf{sk}} + 2\kappa$ bits to κ bits. Let F_1 and F_2 be families of puncturable pseudorandom functions, where F_1 has input length $2L_{\mathsf{sk}} + 3\kappa$ bits and output length L_r bits, and it is as well an $(L_r + \kappa, \epsilon)$-good unseeded extractor; F_2 has input length κ and output length $L_{\mathsf{sk}} + 2\kappa$. Here $|u_1| = \kappa$ and $|u_2| = L_{\mathsf{sk}} + 2\kappa$, $|r'| = 2\kappa$.

Define the algorithm $\mathsf{TransformGen}(1^\kappa, \mathsf{pk})$ that on input the security parameter, a public key pk and a circuit that implements $\mathsf{PKE.Update}(\cdot)$ as follows:

- $\mathsf{TransformGen}$ samples K_1, K_2 as keys for the puncturable PRF as above, and $h \leftarrow \mathcal{H}$. Let P_1 be the program as Figure 1, and P_2 as Figure 2.
- Then it samples $\mathcal{P}_{\mathsf{update}} \leftarrow \mathsf{diO}(P_1)$, and $\mathcal{P}_{\mathsf{explain}} \leftarrow \mathsf{diO}(P_2)$. It outputs $(\mathcal{P}_{\mathsf{update}}, \mathcal{P}_{\mathsf{explain}})$.

Internal (hardcoded) state: Public key pk, keys K_1, K_2, and h.

On input secret key sk_1; randomness $u = (u_1, u_2)$.
- If $F_2(K_2, u_1) \oplus u_2 = (\mathsf{sk}_2, r')$ for (proper length) strings sk_2, r' and $u_1 = h(\mathsf{sk}_1, \mathsf{sk}_2, r')$, then output sk_2.
- Else let $x = F_1(K_1, (\mathsf{sk}_1, u))$. Output $\mathsf{sk}_2 = \mathsf{PKE.Update}(\mathsf{pk}, \mathsf{sk}_1; x)$.

Figure 1: Program Update

Internal (hardcoded) state: key K_2.

On input secret keys $\mathsf{sk}_1, \mathsf{sk}_2$; randomness $r \in \{0,1\}^\kappa$
- Set $u_1 = h(\mathsf{sk}_1, \mathsf{sk}_2, r)$. Set $u_2 = F_2(K_2, u_1) \oplus (\mathsf{sk}_2, r)$. Output $e = (u_1, u_2)$.

Figure 2: Program Explain

Then we can establish the following theorem.

Theorem 3.4 *Let* PKE *be any public key encryption scheme with key update. Assume* diO *is a secure* public-coin *differing-inputs indistinguishable obfuscator for the circuits required by the construction, F_1, F_2 are puncturable pseudorandom functions as above, and $\mathcal{H}$ is a family of public-coin collision resistant hash functions as above. Then the transformation* TransformGen *defined above is a secure explainable update transformation for* PKE *as defined in Definition 3.2.*

Proof: Recall we need to demonstrate that for any public key pk^* and secret key $\mathsf{sk}^* \in \Pi_{\mathsf{pk}}$, the following two distributions are computationally indistinguishable:

$$\{(\mathcal{P}_{\mathsf{update}}, \mathcal{P}_{\mathsf{explain}}, \mathsf{pk}^*, \mathsf{sk}^*, u^*)\} \approx \{(\mathcal{P}_{\mathsf{update}}, \mathcal{P}_{\mathsf{explain}}, \mathsf{pk}^*, \mathsf{sk}^*, e^*)\},$$

where these values are generated by

1. $(\mathcal{P}_{\mathsf{update}}, \mathcal{P}_{\mathsf{explain}}) \leftarrow \mathsf{TransformGen}(1^\kappa, \mathsf{PKE.Update}, \mathsf{pk}^*)$,
2. $u^* = (u_1^*, u_2^*) \leftarrow \{0,1\}^{L_{\mathsf{sk}}+3\kappa}$,
3. Set $x^* = F_1(K_1, \mathsf{sk}^* || u^*)$, $\mathsf{sk}' = \mathcal{P}_{\mathsf{update}}(\mathsf{sk}^*; u^*)$. Then choose uniformly random r^* of length κ, and set $e_1^* = h(\mathsf{sk}^*, \mathsf{sk}', r^*)$ and $e_2^* = F_2(K_2, e_1^*) \oplus (\mathsf{sk}', r^*)$.

We prove this through the following sequence of hybrid steps.

Hybrid 1: In this hybrid step, we change Step 3 of the above challenge. Instead of computing $\mathsf{sk}' = \mathcal{P}_{\mathsf{update}}(\mathsf{sk}^*; u^*)$, we compute $\mathsf{sk}' = \mathsf{PKE.Update}(\mathsf{pk}^*, \mathsf{sk}^*; x^*)$:

1. $(\mathcal{P}_{\mathsf{update}}, \mathcal{P}_{\mathsf{explain}}) \leftarrow \mathsf{TransformGen}(1^\kappa, \mathsf{PKE.Update}, \mathsf{pk}^*)$,
2. $u^* = (u_1^*, u_2^*) \leftarrow \{0,1\}^{L_{\mathsf{sk}}+3\kappa}$,
3. Set $x^* = F_1(K_1, \mathsf{sk}^* || u^*)$, $\mathsf{sk}' = \mathsf{PKE.Update}(\mathsf{pk}^*, \mathsf{sk}^*; x^*)$, and choose uniformly random r^* of length κ. Then, $e_1^* = h(\mathsf{sk}^*, \mathsf{sk}', r^*)$ and $e_2^* = F_2(K_2, e_1^*) \oplus (\mathsf{sk}', r^*)$.

Note that the only time in which this changes the experiment is when the values $(u_1^*, u_2^*) \leftarrow \{0,1\}^{2L_{\mathsf{sk}}+3\kappa}$ happen to satisfy $F_2(K_2, u_1^*) \oplus u_2^* = (\mathsf{sk}', r')$ such that $u_1^* = h(\mathsf{sk}^*, \mathsf{sk}', r')$. For any fixed $u_1^*, \mathsf{sk}^*, \mathsf{sk}'$, and a random u_{2^*}, we know the marginal probability of r' is still uniform given $u_1^*, \mathsf{sk}^*, \mathsf{sk}'$. Therefore, we have $\Pr_{u_{2^*}}[h(\mathsf{sk}^*, \mathsf{sk}', r') = u_1^*] = \Pr_{r'}[h(\mathsf{sk}^*, \mathsf{sk}', r') = u_1^*] < 2^{-\kappa} + \epsilon$. This is because h is a $(2\kappa, \epsilon)$-extractor, so the output of h is ϵ-close to uniform over $\{0,1\}^\kappa$, and a uniform distribution hits a particular string with probability $2^{-\kappa}$. Since we set ϵ to be some negligible, the two distributions are only different with the negligible quantity.

[8] The extractor outputs a distribution that is ϵ close to the uniform distribution if the source has min-entropy 2κ. Here we set ϵ to be some negligible. The hash function is chosen from a family of functions, and once chosen, it is a deterministic function.

Hybrid 2: In this hybrid step, we modify the program in Figure 1, puncturing key K_1 at points $\{\mathsf{sk}_1||u^*\}$ and $\{\mathsf{sk}_1||e^*\}$, and adding a line of code at the beginning of the program to ensure that the PRF is never evaluated at these two points. See Figure 3. We claim that with overwhelming probability over the choice of u^*, this modified program has identical input/output as the program that was used in Hybrid 1 (Figure 1). Note that on input (sk^*, e^*) the output of the original program was already sk' as defined in Hybrid 1, so the outputs of the two programs are identical on this input. (This follows because e^* anyway encodes sk', so when the "If"-statement is triggered in the program of Figure 1, the output is sk'.) As long as u_1^* and u_2^* do not have the property that $u_1^* = h(\mathsf{sk}^*, F_2(K_2, u_1^*) \oplus u_2^*)$, then the programs have identical output on input (sk^*, u^*) as well. (This follows because sk' is defined as $\mathsf{sk}' = \mathcal{P}_{\mathsf{update}}(\mathsf{sk}^*; F_1(K_1, \mathsf{sk}^*||u^*))$ in the challenge game, which is also the output of the program in Figure 1 when u_1^* and u_2^* fail this condition.) As we argued in Hybrid 1, with very high probability, u^* does not have this property. (We stress that u^* is fixed *before* we construct the obfuscated program described in Figure 3, so with overwhelming probability over the choice of u^*, the two programs have identical input output behavior.) Indistinguishability of Hybrids 1 and 2 follows from the security of the obfuscation.

Internal (hardcoded) state: Public key pk^*, keys $\widetilde{K}_1 = \mathsf{PRF.Punct}(K_1, \{\mathsf{sk}^*||u^*\}, \{\mathsf{sk}^*||e^*\})$, K_2, sk' (as defined in Hybrid 1) and h.

On input secret key sk_1; randomness $u = (u_1, u_2)$.

- If $(\mathsf{sk}_1, u) = (\mathsf{sk}^*, u^*)$ or $(\mathsf{sk}_1, u) = (\mathsf{sk}^*, e^*)$ output the value sk'.
- Else If $F_2(K_2; u_1) \oplus u_2 = (\mathsf{sk}_2, r')$ such that $u_1 = h(\mathsf{sk}_1, \mathsf{sk}_2, r')$, then output sk_2.
- Else let $x = F_1(\widetilde{K}_1, \mathsf{sk}_1||u)$. Output $\mathsf{sk}_2 = \mathsf{PKE.Update}(\mathsf{pk}^*, \mathsf{sk}_1; x)$.

Figure 3: Program Update, as used in Hybrid 2

Hybrid 3: In this Hybrid we change the challenge game to use truly random x^* when computing $\mathsf{sk}' = \mathsf{PKE.Update}(\mathsf{pk}^*, \mathsf{sk}^*; x^*)$, (instead of $x^* = F_1(K_1; \mathsf{sk}^*||u^*)$). Security holds by a reduction to the pseudo-randomness of F_1 at the punctured point (sk^*, u^*). More specifically, given an adversary $\mathcal{A}$ that distinguishes Hybrid 2 from Hybrid 3 on values $\mathsf{pk}^*, \mathsf{sk}^*$, we describe an reduction $\mathcal{B}$ that attacks the security of the puncturable PRF, F_1. $\mathcal{B}$ generates u^* at random and submits (sk^*, u^*) to his challenger. He receives $\widetilde{K}_1 = \mathsf{PRF.Punct}(K_1, \{\mathsf{sk}^*||u^*\})$, and a value x^* as a challenge. $\mathcal{B}$ computes $\mathsf{sk}' = \mathsf{PKE.Update}(\mathsf{pk}^*, \mathsf{sk}^*; x^*)$, chooses r^* at random, and computes e^* as in the original challenge game. He creates $\mathcal{P}_{\mathsf{update}}$ using $\widetilde{K}_1$ and sampling K_2 honestly. The same K_2 is used for creating $\mathcal{P}_{\mathsf{explain}}$. $\mathcal{B}$ obfuscates both circuits, which completes the simulation of $\mathcal{A}$'s view.

Hybrid 4: In this hybrid, we puncture K_2 at both u_1^* and e_1^*, and modify the Update program to output appropriate hardcoded values on these inputs. (See Figures 4.) To prove that Hybrids 3 and 4 are indistinguishable, we rely on security of public-coin differing-inputs obfuscation and public-coin collision resistant hash function. In particular, we will show that suppose the Hybrids are distinguishable, then we can break the security of the collision resistant hash function.

Consider the following sampler $\mathsf{Samp}(1^\kappa)$: outputs C_0, C_1 as the two update programs as in Hybrids 3 and 4 respectively; and it outputs an auxiliary input $\mathsf{aux} = (\mathsf{pk}^*, \mathsf{sk}^*, \mathsf{sk}', u^*, e^*, K_2, h, r^*)$ sampled as in the both hybrids. Note that aux includes all the random coins of the sampler. Suppose there exists a distinguisher $\mathcal{D}$ for the two hybrids, then there exists a distinguished $\mathcal{D}'$ that distinguishes $(\mathsf{diO}(C_0), \mathsf{aux})$ from $(\mathsf{diO}(C_1), \mathsf{aux})$. This is because given the challenge input, $\mathcal{D}'$ can complete the rest of the experiment either according to Hybrid 3 or Hybrid 4. Then by security of the diO, we know there exists an adversary (extractor) $\mathcal{B}$ that given (C_0, C_1, aux) finds an input such that C_0 and C_1 evaluate differently. However, this contradicting the security of the public-coin collision resistant hash function. We establish this by the following lemma.

Lemma 4 *Assume h is sampled from a family of public-coin collision resistant hash function, (and $(2\kappa, \epsilon)$-extracting) as above. Then for any PPT adversary, the probability is negligible to find a differing input given (C_0, C_1, aux) as above.*

Proof: By examining the two circuits, we observe that the differing inputs have the following two forms: $(\bar{\mathsf{sk}}, u_1^*, \bar{u}_2)$ such that $u_1^* = h(\bar{\mathsf{sk}}, F_2(K_2; u_1^*) \oplus \bar{u}_2)$, $(\bar{\mathsf{sk}}, \bar{u}_2) \neq (\mathsf{sk}^*, u_2^*)$; or $(\bar{\mathsf{sk}}, e_1^*, \bar{e}_2)$ such that $e_1^* = h(\bar{\mathsf{sk}}, F_2(K_2; e_1^*) \oplus \bar{e}_2)$, $(\bar{\mathsf{sk}}, \bar{e}_2) \neq (\mathsf{sk}^*, e_2^*)$. This is because they will run enter the first Else IF in Hybrid 3 (Figure 3), but will enter the modified line (the first Else IF) in Hybrid 4 (Figure 4). We argue that both cases happen with negligible probability; otherwise security of the hash function can be broken.

For the first case, we observe that the collision resistance and $(2\kappa, \epsilon)$ extracting guarantee that the probability of finding an pre-image of a random value u_1^* is small, even given aux; otherwise there is an adversary who can break collision resistance. For the second case, we know that $e_1^* = h(\mathsf{sk}^*, \mathsf{sk}', r^*) = h(\bar{\mathsf{sk}}, F_2(K_2; e_1^*) \oplus \bar{e}_2) = h(\bar{\mathsf{sk}}, e_2^* \oplus (\mathsf{sk}', r^*) \oplus \bar{e}_2)$. Since we know that $(\bar{\mathsf{sk}}, \bar{e}_2) \neq (\mathsf{sk}^*, e_2^*)$, we find a collision, which again remains hard even given aux.

Thus, suppose there exists a differing-input finder $\mathcal{A}$, we can define an adversary $\mathcal{B}$ to break the collision resistant hash function: on input h, $\mathcal{B}$ simulates the sampler Samp with the h. Then it runs $\mathcal{A}$ to find a differing input. Then according to the above argument, either of the two cases will lead to finding a collision. ∎

Internal (hardcoded) state: Public key pk^*, keys $\widetilde{K}_1 = \mathsf{PRF.Punct}(K_1, \{\mathsf{sk}^*||u^*\}, \{\mathsf{sk}^*||e^*\})$, $\widetilde{K}_2 = \mathsf{PRF.Punct}(K_2, \{u_1^*\}, \{e_1^*\})$, sk' (as defined in Hybrid 3) and h.

On input secret key sk_1; randomness $u = (u_1, u_2)$.

- If $(\mathsf{sk}_1, u) = (\mathsf{sk}^*, u^*)$ or $(\mathsf{sk}_1, u) = (\mathsf{sk}^*, e^*)$ output value sk'.
- Else If $u_1 = u_1^*$ or $u_1 = e_1^*$, let $x = F_1(\widetilde{K}_1, \mathsf{sk}_1||u)$. Output $\mathsf{sk}_2 = \mathsf{PKE.Update}(\mathsf{pk}^*, \mathsf{sk}_1; x)$.
- Else
  - If $F_2(K_2; u_1) \oplus u_2 = (\mathsf{sk}_2, r')$ such that $u_1 = h(\mathsf{sk}_1, \mathsf{sk}_2, r')$, then output sk_2.
  - Else let $x = F_1(\widetilde{K}_1, \mathsf{sk}_1||u)$. Output $\mathsf{sk}_2 = \mathsf{PKE.Update}(\mathsf{pk}^*, \mathsf{sk}_1; x)$.

Figure 4: Program Update, as used in Hybrid 4

Hybrid 5: In this hybrid, we puncture K_2 at both u_1^* and e_1^*, and modify the Explain program to output appropriate hardcoded values on these inputs. (See Figures 5.) Similar to the argument for the previous hybrids, we argue that Hybrids 4 and 5 are indistinguishable by security of the public-coin differing-inputs obfuscation and public-coin collision resistant hash function. Consider a sampler $\mathsf{Samp}(1^\kappa)$: outputs C_0, C_1 as the two explain programs as in Hybrids 4 and 5 respectively; and it outputs an auxiliary input $\mathsf{aux} = (\mathsf{pk}^*, \mathsf{sk}^*, \mathsf{sk}', u^*, e^*, \widetilde{K}_2, h, r^*)$ sampled as in the both hybrids (note that aux includes all the random coins of the sampler). Similar to the above argument: suppose there exists a distinguisher $\mathcal{D}$ that distinguishers Hybrids 4 and 5, then we can construct a distinguisher $\mathcal{D}'$ that distinguishes $(\mathsf{diO}(C_0), \mathsf{aux})$ from $(\mathsf{diO}(C_1), \mathsf{aux})$. This is because given the challenging input, $\mathcal{D}'$ can simulate the hybrids. Then by security of the diO, there exists an adversary (extractor) $\mathcal{B}$ that can find differing inputs. Now we want to argue that suppose the h comes from a public-coin collision resistant hash family, then no PPT adversary can find differing inputs. This leads to a contradiction.

Lemma 5 *Assume h is sampled from a family of public-coin collision resistant hash function, (and $(2\kappa, \epsilon)$-extracting) as above. Then for any* PPT *adversary, the probability is negligible to find a differing input given (C_0, C_1, aux) as above.*

Proof: The proof is almost identical to that of Lemma 4. We omit the details. ∎

Internal (hardcoded) state: key $\widetilde{K}_2 = \mathsf{PRF.Punct}(K_2, \{u_1^*\}, \{e_1^*\})$, u^*, e^*.

On input secret keys $\mathsf{sk}_1, \mathsf{sk}_2$; randomness $r \in \{0,1\}^\kappa$

- If $u_1^* = h(\mathsf{sk}_1, \mathsf{sk}_2, r)$, output u^*. Else If $e_1^* = h(\mathsf{sk}_1, \mathsf{sk}_2, r)$, output e^*.
- Else, set $u_1 = h(\mathsf{sk}_1, \mathsf{sk}_2, r)$. Set $u_2 = F_2(K_2, u_1) \oplus (\mathsf{sk}_2, r)$. Output $e = (u_1, u_2)$.

Figure 5: Program Explain, as used in Hybrid 5

Hybrid 6: In this hybrid, we change both e_1^* and e_2^* to uniformly random. Hybrids 5 and 6 are indistinguishable by the security of the puncturable PRF F_2, and by the fact that h is $(2\kappa, \epsilon)$-extracting. Clearly in this hybrid, the distributions of $\{(\mathcal{P}_{\mathsf{update}}, \mathcal{P}_{\mathsf{explain}}, \mathsf{pk}^*, \mathsf{sk}^*, u^*)\}$ and $\{(\mathcal{P}_{\mathsf{update}}, \mathcal{P}_{\mathsf{explain}}, \mathsf{pk}^*, \mathsf{sk}^*, e^*)\}$ are identical. From the indistinguishable arguments that the original game and Hybrid 6 are indistinguishable, we can argue that the distributions in the original game are indistinguishable. This concludes the proof.

■

4 2CLR from "Leakage Resilient Subspaces"

We show that the PKE scheme of Brakerski et al. [12] (BKKV), which has been proven CLR, can achieve 2CLR (with a slight adjustment in the scheme's parameters). We note that our focus on PKE here is justified by the fact that we show generically in the full version [18] that any CLR (resp. 2CLR) PKE scheme implies a CLR "one-way relation" (OWR) [21]; to the best of our knowledge, such an implication was not previously known. Therefore, by the results of Dodis et al. [21], this translates all our results about PKE to the signature setting as well. In the full version [18] of the paper, we show that the approach of Dodis et al. [21] for constructing CLR OWRs can be extended to 2CLR one-way relations, but we achieve weaker parameters this way.

Recall that in the work [12], to prove that their scheme is CLR, they show *"random subspaces are leakage resilient"*. In particular, they show that for a random subspace X, the statistical difference between $\big(X, f(v)\big)$ and $\big(X, f(u)\big)$ is negligible, where f is an arbitrary length-bounded function, v is a random point in the subspace, and u is a random point in the whole space. Then by a simple hybrid argument, they show that $\big(X, f_1(v_0), f_2(v_1), \ldots, f_t(v_{t-1})\big)$ and $\big(X, f_1(u_0), f_2(u_1), \ldots, f_t(u_{t-1})\big)$ are indistinguishable, where $f_1, \ldots, f_t$ are arbitrary and adaptively chosen length-bounded functions , $v_0, v_1, \ldots, v_{t-1}$ are independent random points in the subspace, and $u_0, u_1, \ldots, u_{t-1}$ are independent random points in the whole space. This lemma plays the core role in their proof.

In order to show that their scheme satisfies the 2CLR security, we consider random subspaces under "consecutive" leakage. That is, we want to show:

$$\big(X, f_1(v_0, v_1), f_2(v_1, v_2), \ldots, f_t(v_{t-1}, v_t)\big) \approx \big(X, f_1(u_0, u_1), f_2(u_1, u_2), \ldots, f_t(u_{t-1}, u_t)\big),$$

for arbitrary and adaptively chosen f_i's, i.e. each f_i can be chosen after seeing the previous leakage values $f_1, \ldots, f_{i-1}$. However, this does not follow by a hybrid argument of $\big(X, f(v)\big) \approx \big(X, f(u)\big)$, because in the 2CLR case each point is leaked twice. It is not clear how to embed a challenging instance of $(X, f(z))$ into the larger experiment while still being able to simulate the rest.

To handle this technical issue, we establish a new lemma showing *random subspaces are "consecutive" leakage resilient*. With the lemma and a hybrid argument, we can show that the above experiments are indistinguishable. Then we show how to use this fact to prove that the scheme of BKKV is 2CLR.

Lemma 6 *Let $t, n, \ell, d \in \mathbb{N}$, $n \geq \ell \geq 3d$, and q be a prime. Let $(A, X) \leftarrow \mathbb{Z}_q^{t\times n} \times \mathbb{Z}_q^{n\times \ell}$ such that $A \cdot X = 0$, $T, T' \leftarrow \mathsf{Rk}_d(\mathbb{Z}_q^{\ell\times d})$, $U \leftarrow \mathbb{Z}_q^{n\times d}$ such that $A \cdot U = 0$, (i.e. U is a random matrix in $\mathsf{Ker}(A)$), and f :*

$\mathbb{Z}_q^{t\times n} \times \mathbb{Z}_q^{n\times 2d} \to W$ *be any function*[9] *. Then we have:*

$$\Delta\left((A, X, f(A, XT, XT'), XT'), (A, X, f\left(A, U, XT'\right), XT')\right) \leq \epsilon,$$

as long as $|W| \leq (1 - 1/q) \cdot q^{\ell-3d+1} \cdot \epsilon^2$.

Proof: We will actually prove something stronger, namely we will prove, under the assumptions of the Lemma 6, that

$$\Delta\left(\left(A, X, f(A, X \cdot T, X \cdot T'), X \cdot T', T'\right), \left(A, X, f(A, U, X \cdot T'), X \cdot T', T'\right)\right)$$
$$\leq \frac{1}{2}\sqrt{\frac{3|W|}{(1-1/q)q^{\ell-3d+1}}} < \epsilon\ .$$

Note that this implies the Lemma by solving for ϵ, after noting that ignoring the last component in each tuple can only decrease statistical difference.

For the proof, we will apply Lemma 7 as follows. We will take hash function H to be $H\colon \mathbb{Z}_q^{n\times\ell} \times \mathbb{Z}_q^{\ell\times d} \to \mathbb{Z}_q^{n\times d}$ where $H_K(D) = KD$ (matrix multiplication), and take the set $\mathcal{Z}$ to be $\mathbb{Z}_q^{n\times\ell} \times \mathbb{Z}_q^{\ell\times d}$. Next we take random variable K to be uniform on $\mathbb{Z}_q^{n\times\ell}$ (denoted as the matrix X), D to be uniform on $\mathsf{Rk}_d(\mathbb{Z}_q^{\ell\times d})$, and finally $Z = (A, XT', T')$ where A is uniform conditioned on $AX = 0$, $T' \in \mathsf{Rk}_d(\mathbb{Z}_q^{\ell\times d})$ is independent uniform. We define $U_{|Z}$ as the uniform distribution such that $AU = 0$. This also means that U is a random matrix in the kernel of A.

It remains to prove under these settings that

$$\Pr\left[\,(D, D', Z) \in \mathsf{BAD}\,\right] \leq \frac{1}{(1-1/q)q^{\ell-3d+1}}$$

with BAD defined as in Lemma 7. For this let us consider

$$\Delta\big((H_{K|_Z}(T_1), H_{K|_Z}(T_2)), (U_{|Z}, U'_{|Z})\big)$$

where $Z = (A, XT', T')$ as defined above. The above statistical distance is zero as long as the outcomes of T_1, T_2, T' are all linearly independent. This is so because $\ell \geq 3d$. Now, by a standard formula the probability that T_1, T_2, T' have a linear dependency is bounded by $\frac{1}{(1-1/q)q^{\ell-3d+1}}$, and we are done. ∎

We note that this lemma is slightly different that the original lemma in the work [12]: the leakage function considered here also takes in a public matrix A, which is used as the public key in the system. We observe that both our work and [12] need this version of the lemma to prove security of the encryption scheme.

We actually prove Lemma 6 as a consequence of a new generalization of the Crooked Leftover Hash Lemma (LHL) [26, 6] we introduce (to handle hash functions that are only pairwise independent if some bad event does not happen), as follows.

Lemma 7 *Let* $H\colon \mathcal{K} \times \mathcal{D} \to \mathcal{R}$ *be a hash function and* (K, Z) *be joint random variables over* $(\mathcal{K}, \mathcal{Z})$ *for the set* $\mathcal{K}$ *and some set* $\mathcal{Z}$*. Define the following set*

$$\mathsf{BAD} = \left\{(d, d', z) \in \mathcal{D} \times \mathcal{D} \times \mathcal{Z} : \Delta\big((H_{K|_{Z=z}}(d), H_{K|_{Z=z}}(d')), (U_{|Z=z}, U'_{|Z=z})\big) > 0\right\}, \tag{1}$$

where $U_{|Z=z}, U'_{|Z=z}$ *denote two independent uniform distributions over* $\mathcal{R}$ *conditioned on* $Z = z$*, and* $K|_{Z=z}$ *is the conditional distribution of* K *given* $Z = z$*. We note that* $\mathcal{R}$ *might depend on* z*, so when we describe a uniform distribution over* $\mathcal{R}$*, we need to specify the condition* $Z = z$*.*

Suppose D *and* D' *are i.i.d. random variables over* $\mathcal{D}$*,* (K, Z) *are random variables over* $\mathcal{K} \times \mathcal{Z}$ *satisfying* $\Pr\left[\,(D, D', Z) \in \mathsf{BAD}\,\right] \leq \epsilon'$*. Then for any set* $\mathcal{S}$ *and function* $f\colon \mathcal{R} \times \mathcal{Z} \to \mathcal{S}$ *it holds that*

$$\Delta((K, Z, f(H_K(D), Z)), (K, Z, f(U_{|Z}, Z))) \leq \frac{1}{2}\sqrt{3\epsilon' \ |\mathcal{S}|}\ .$$

[9]Note: Rk denotes rank. Here we use n as the dimension (different from [12] who used m) to avoid overloading notation.

Proof: The proof is an extension of the proof of the Crooked LHL given in [6]. First, using Cauchy-Schwarz and Jensen's inequality we have

$$\Delta((K, Z, f(H_K(D), Z)), (K, Z, f(U_{|Z}, Z))) \leq \frac{1}{2}\sqrt{|S|\,\mathbf{E}_{k,z}\left[\sum_s (\Pr[\,f(H_k(D), z) = s\,] - \Pr[\,f(U_{|Z=z}, z) = s\,])^2\right]},$$

where $U_{|Z=z}$ is uniform on $\mathcal{R}$ conditioned on $Z = z$, and the expectation is over (k, z) drawn from (K, Z). Thus, to complete the proof it suffices to prove the following lemma.

Lemma 8

$$\mathbf{E}_{k,z}\left[\sum_s \left(\Pr[\,f(H_k(D), z) = s\,] - \Pr[\,f(U_{|Z=z}, z) = s\,]\right)^2\right] \leq 3\epsilon'\,. \tag{2}$$

Proof: By the linearity of expectation, we can express Equation 2 as:

$$\mathbf{E}_{k,z}\sum_s \Pr[\,f(H_k(D), z) = s\,]^2 - 2\mathbf{E}_{k,z}\sum_s \Pr[\,f(H_k(D), z) = s\,]\Pr[\,f(U_{|Z=z}, z) = s\,] + \mathbf{E}_z \mathsf{Col}(f(U_{|Z=z}, z)), \tag{3}$$

where $U_{|Z=z}$ is uniform on $\mathcal{R}$ conditioned on $Z = z$, and Col is the collision probability of its input random variable. Note that since $f(U_{|Z=z}, z)$ is independent of k, we can drop it in the third term. In the following, we are going to calculate bounds for the first two terms.

For any $s \in \mathcal{S}$, we can write $\Pr[\,f(H_k(D), z) = s\,] = \sum_d \Pr[\,D = d\,]\delta_{f(H_k(d),z),s}$ where $\delta_{a,b}$ is 1 if $a = b$ and 0 otherwise, and thus

$$\sum_s \Pr[\,f(H_k(D), z) = s\,]^2 = \sum_{d,d'} \Pr[\,D = d\,]\Pr[\,D = d'\,]\delta_{f(H_k(d),z),f(H_k(d'),z)}\,.$$

So we have

$$\begin{aligned}
\mathbf{E}_{k,z}\sum_s \Pr[\,f(H_k(D), z) = s\,]^2 &= \mathbf{E}_{k,z}\left[\sum_{d,d'} \Pr[\,D = d\,]\Pr[\,D = d'\,]\delta_{f(H_k(d),z),f(H_k(d'),z)}\right] \\
&= \mathbf{E}_z\left[\sum_{d,d'} \Pr[\,D = d\,]\Pr[\,D = d'\,]\mathbf{E}_k\left[\delta_{f(H_k(d),z),f(H_k(d'),z)}\right]\right] \\
&\leq \sum_{z,d,d' \notin \mathsf{BAD}} \Pr[\,Z = z\,]\Pr[\,D = d\,]\Pr[\,D = d'\,]\mathbf{E}_k\left[\delta_{f(H_k(d),z),f(H_k(d'),z)}\right] + \epsilon' \\
&= \mathbf{E}_z\left[\mathsf{Col}(f(U_{|Z=z}, z))\right] + \epsilon',
\end{aligned} \tag{4}$$

where BAD is defined as in equation (1) from Lemma 7. The inequality holds because, by our definition of BAD, if $(z, d, d') \notin \mathsf{BAD}$, $(H_k(d), H_k(d'))$ are distributed exactly as two uniformly chosen elements (conditioned on $Z = z$), and because $\Pr[(z, d, d') \in \mathsf{BAD}] \leq \epsilon'$.

By a similar calculation, we have:

$$\mathbf{E}_{k,z}\sum_s \Pr[\,f(H_k(D), z) = s\,]\Pr[\,f(U_{|Z=z}, z) = s\,] \geq \mathbf{E}_z\left[\mathsf{Col}(f(U_{|Z=z}, z))\right] - \epsilon'\,. \tag{5}$$

For the same reason, $H_k(D)$ is uniformly random except for the bad event, whose probability is bounded by ϵ'.

Putting things together, the inequality in Equation 2 follows immediately by plugging the bounds in Equations 4 and 5. This concludes the proof.

■

■

Here we describe the BKKV encryption scheme, and show it is 2CLR-secure. We begin by presenting the main scheme in BKKV, which uses the weaker linear assumption, but achieves a worse leakage rate (that can tolerate roughly $1/8 \cdot |\mathsf{sk}| - o(\kappa)$). In that work [12], it is also pointed out that under the stronger SXDH assumption, the rate can be improved to tolerate roughly $1/4 \cdot |\mathsf{sk}| - o(k)$, with essentially the same proof. The same argument also holds in the 2CLR setting. To avoid repetition, we just describe the original scheme in BKKV, and prove that it is actually 2CLR under the linear assumption.

- **Parameters.** Let G, G_T be two groups of prime order p such that there exists a bilinear map $e : G \times G \to G_T$. Let g be a generator of G (and so $e(g, g)$ is a generator of G_T). An additional parameter $\ell \geq 7$ is polynomial in the security parameter. (Setting different ℓ will enable a tradeoff between efficiency and the rate of tolerable leakage). For the scheme to be secure, we require that the linear assumption holds in the group G, which implies that the size of the group must be super-polynomial, i.e. $p = \kappa^{\omega(1)}$.
- **Key-generation.** The algorithm samples $A \leftarrow \mathbb{Z}_p^{2\times\ell}$, and $Y \leftarrow \mathsf{Ker}^2(A)$, i.e. $Y \in \mathbb{Z}_p^{\ell\times 2}$ can be viewed as two random (linearly independent) points in the kernel of A. Then it sets $\mathsf{pk} = g^A$, $\mathsf{sk} = g^Y$. Note that since A is known, Y can be sampled efficiently.
- **Key-update.** Given a secret key $g^Y \in G^{\ell\times 2}$, the algorithm samples $R \leftarrow \mathsf{Rk}_2(\mathbb{Z}_p^{2\times 2})$ and then sets $\mathsf{sk}' = g^{Y\cdot R}$.
- **Encryption.** Given a public key $\mathsf{pk} = g^A$, to encrypt 0, it samples a random $r \in \mathbb{Z}_p^2$ and outputs $c = g^{r^T\cdot A}$. To encrypt 1, it just outputs $c = g^{u^T}$ where $u \leftarrow \mathbb{Z}_p^\ell$ is a uniformly random vector.
- **Decryption.** Given a ciphertext $c = g^{v^T}$ and a secret key $\mathsf{sk} = g^Y$, the algorithm computes $e(g, g)^{v^T\cdot Y}$. If the result is $e(g, g)^0$, then it outputs 0; otherwise 1.

Then we are able to achieve the following theorem:

Theorem 4.1 *Under the linear assumption, for every $\ell \geq 7$, the encryption scheme scheme above is μ-bit leakage resilient against two-key continual and consecutive leakage, where $\mu = \frac{(\ell-6)\cdot\log p}{2} - \omega(\kappa)$. Note that the leakage rate would be $\frac{\mu}{|\mathsf{sk}|+|\mathsf{sk}|} \approx 1/8$, as ℓ is chosen sufficiently large.*

Proof: The theorem follows directly from the following lemma:

Lemma 9 *For any $t \in \mathrm{poly}(\kappa)$, $r \leftarrow \mathbb{Z}_p^2$, $A \leftarrow \mathbb{Z}_p^{2\times\ell}$, random $Y \in \mathsf{Ker}^2(A)$, and polynomial sized functions $f_1, f_2, \ldots, f_t$ where each $f_i : \mathbb{Z}_p^{\ell\times 2} \times \mathbb{Z}_p^{\ell\times 2} \to \{0,1\}^\mu$ and can be adaptively chosen (i.e. f_i can be chosen after seeing the leakage values of $f_1, \ldots, f_{i-1}$), the following two distributions, D_0 and D_1, are computationally indistinguishable:*

$$D_0 = (g, g^A, g^{r^T\cdot A}, f_1(\mathsf{sk}_0, \mathsf{sk}_1), \ldots f_t(\mathsf{sk}_{t-1}, \mathsf{sk}_t))$$

$$D_1 = (g, g^A, g^u, f_1(\mathsf{sk}_0, \mathsf{sk}_1), \ldots f_t(\mathsf{sk}_{t-1}, \mathsf{sk}_t)),$$

where $\mathsf{sk}_0 = g^Y$ and $\mathsf{sk}_{i+1} = (\mathsf{sk}_i)^{R_i}$ for R_i a random 2 by 2 matrix of rank 2.

Basically, the distribution D_0 is the view of the adversary when given an encryption of 0 as the challenge ciphertext and continual leakage of the secret keys; D_1 is the same except the challenge ciphertext is an encryption of 1. Our goal is to show that no polynomial sized adversary can distinguish between them.

We show the lemma in the following steps:

1. We first consider two modified experiment D'_0 and D'_1 where in these experiments, all the secret keys are sampled independently, i.e. $\mathsf{sk}'_{i+1} \leftarrow \mathsf{Ker}^2(A)$. In other words, instead of using a rotation of the current secret key, the update procedure resamples two random (linearly independent) points in the kernel of A. Denote $D'_b = (g, g^A, g^z, f_1(\mathsf{sk}'_0, \mathsf{sk}'_1), \ldots f_t(\mathsf{sk}'_{t-1}, \mathsf{sk}'_t))$ for g^z is sampled either from $g^{r^T \cdot A}$ or g^u depending on $b \in \{0,1\}$. Intuitively, the operations are computed in the exponent, so the adversary cannot distinguish between the modified experiments from the original ones. We formally prove this using the linear assumption.

2. Then we consider the following modified experiments: for $b \in \{0,1\}$, define

$$D''_b = (g, g^A, g^z, f_1(g^{u_0}, g^{u_1}), f_2(g^{u_1}, g^{u_2}), \cdots, f_t(g^{u_{t-1}}, g^{u_t})),$$

   where the distribution samples a random $X \in \mathbb{Z}_p^{\ell \times (\ell-3)}$ such that $A \cdot X = 0$; then it samples each $u_i = X \cdot T_i$ for $T_i \leftarrow \mathsf{Rk}_2(\mathbb{Z}_p^{(\ell-3)\times 2})$; finally it samples z either as $r^T \cdot A$ or uniformly random as in D'_b. We then show that D''_b is indistinguishable from D'_b using the new geometric lemma.

3. Finally, we show that $D''_0 \approx D''_1$ under the linear assumption.

To implement the approach just described, we establish the following lemmas.

Lemma 10 *For both $b \in \{0,1\}$, D_b is computationally indistinguishable from D'_b.*

To show this lemma, we first establish a lemma:

Lemma 11 *Under the linear assumption, $(g, g^A, g^Y, g^{Y \cdot U}) \approx (g, g^A, g^Y, g^{Y'})$, where $A \leftarrow \mathbb{Z}_p^{2\times \ell}$, Y, Y' $\mathsf{Ker}^2(A)$, and $U \leftarrow \mathsf{Rk}_2(\mathbb{Z}_p^{2\times 2})$.*

Suppose there exists a distinguisher $\mathcal{A}$ that breaks the above statement with non-negligible probability, then we can construct $\mathcal{B}$ that can break the linear assumption (the matrix form). In particular, $\mathcal{B}$ distinguishes $(g, g^C, g^{C \cdot U})$ from $(g, g^C, g^{C'})$ where C and C' are two independent and uniformly random samples from $\mathbb{Z}_p^{(\ell-2)\times 2}$, and U is uniformly random matrix from $\mathbb{Z}_p^{2\times 2}$. Note that when $p = \kappa^{\omega(1)}$ (this is required by the linear assumption), then with overwhelming probability, $(C||C')$ is a rank 4 matrix, and $(C||C \cdot U)$ is a rank 2 matrix. The linear assumption is that no polynomial time adversary can distinguish the two distributions when given in the exponent.

$\mathcal{B}$ does the following on input (g, g^C, g^Z), where Z is either $C \cdot U$ or a uniformly random matrix C':

- $\mathcal{B}$ samples a random rank 2 matrix $A \in \mathbb{Z}_p^{2\times \ell}$. Then $\mathcal{B}$ computes an arbitrary basis of $\mathsf{Ker}(A)$ (note that $\mathsf{Ker}(A) = \{v \in \mathbb{Z}_p^\ell : A \cdot v = 0\}$), denoted as X. By the rank-nullity theorem (see any linear algebra textbook), the dimension of $\mathsf{Ker}(A)$ plus $\mathsf{Rk}(A)$ is ℓ. So we know that $X \in \mathbb{Z}_p^{\ell \times (\ell-2)}$, i.e. X contains $(\ell - 2)$ vectors that are linearly independent.
- $\mathcal{B}$ computes $g^{X \cdot C}$ and $g^{X \cdot Z}$. This can be done efficiently given (g^C, g^Z) and X in the clear.
- $\mathcal{B}$ outputs $\mathcal{A}(g, g^A, g^{X \cdot C}, g^{X \cdot Z})$.

We observe that when $p = \kappa^{\omega(1)}$, the distribution of A is statistically close to a random matrix, and U is statistically close to a random rank 2 matrix. Then it is not hard to see that $g^{X \cdot C}$ is identically distributed to g^Y, and $g^{X \cdot Z}$ is distributed as $g^{(X \cdot C) \cdot U}$ if $Z = C \cdot U$, and otherwise as $g^{Y'}$. So $\mathcal{B}$ can break the linear assumption with probability essentially the same as that of $\mathcal{A}$. This completes the proof of the lemma.

Then Lemma 10 can be proven using the lemma via a standard hybrid argument. We show that $D_0 \approx D'_0$ and the other one can be shown by the same argument. For $i \in [t+1]$, define hybrids H_i as the experiment as D_0 except the first i secret keys are sampled independently, as D'_0; the rest are sampled according to rotations,

as D_0. It is not hard to see that $H_1 = D_0$, $H_{t+1} = D'_0$, and $H_i \approx H_{i+1}$ using the lemma. The argument is obvious and standard, so we omit the detail.

Then we recall the modified distribution D''_b: for $b \in \{0,1\}$,

$$D''_b = (g, g^A, g^z, f_1(g^{u_0}, g^{u_1}), f_2(g^{u_1}, g^{u_2}), \cdots, f_t(g^{u_{t-1}}, g^{u_t})),$$

where the distribution samples a random $X \in \mathbb{Z}_p^{\ell \times (\ell-2)}$ such that $A \cdot X = 0$; then it samples each $u_i = X \cdot T_i$ for $T_i \leftarrow \mathsf{Rk}_2(\mathbb{Z}_p^{(\ell-2)\times 2})$, and z is sampled either $r^T \cdot A$ or uniformly random. We then establish the following lemma.

Lemma 12 *For $b \in \{0,1\}$, D'_b is computationally indistinguishable from D''_b.*

We prove the lemma using another hybrid argument. We prove that $D'_0 \approx D''_0$, and the other follows from the same argument. We define hybrids Q_i for $i \in [t]$ where in Q_i, the first i secret keys (the exponents) are sampled randomly from $\mathsf{Ker}^2(A)$ (as D'_0), and the rest secret keys (the exponents) are sampled as $X \cdot T$ (as D''_0). Clearly, $Q_0 = D''_0$ and $Q_{t+1} = D'_0$. Then we want to show that Q_i is indistinguishable from Q_{i+1} using the extended geometric lemma (Lemma 6).

For any $i \in [t+1]$, we argue that suppose there exists an (even unbounded) adversary that distinguishes Q_i from Q_{i+1} with probability better than ϵ, then there exist a leakage function L and an adversary $\mathcal{B}$ such that $\mathcal{B}$ can distinguish $\left(A, X, L(A, X \cdot T, X \cdot T'), X \cdot T'\right)$ from $\left(A, X, L(A, U, X \cdot T'), X \cdot T'\right)$ in Lemma 6 with probability better than $\epsilon - \mathsf{negl}(\kappa)$ (dimensions will be set later). We will set the parameters of Lemma 6 such that the two distributions have negligible statistical difference; thus ϵ can be at most a negligible quantity.

Now we formally set the dimensions: let X be a random matrix in $\mathbb{Z}^{\ell \times (\ell-3)}$; T, T' be two random rank 2 matrices in $\mathbb{Z}_p^{(\ell-3)\times 2}$, i.e. $\mathsf{Rk}_2\left(\mathbb{Z}_p^{(\ell-3)\times 2}\right)$; $L : \mathbb{Z}_p^{\ell\times 2} \times \mathbb{Z}_p^{\ell\times 2} \to \{0,1\}^{2\mu}$; recall that $2\mu = (\ell-6)\cdot \log p - \omega(\kappa)$, and thus $|L| \le p^{\ell-6} \cdot \kappa^{-\omega(1)}$. By Lemma 6, for any (even computationally unbounded) L, we have

$$\Delta\left(\left(A, X, L(A, X \cdot T, X \cdot T'), X \cdot T'\right), \left(A, X, L(A, U, X \cdot T'), X \cdot T'\right)\right) < \kappa^{-\omega(1)} = \mathsf{negl}(\kappa).$$

Let g be a random generator of G, and ω is some randomness chosen uniformly. We define a particular function L^*, with g, ω hardwired, as follows: $L^*(A, w, v)$ on input A, w, v does the following:

- It first samples $Y_0, \ldots, Y_{i-1} \leftarrow \mathsf{Ker}^2(A)$, using the random coins ω. Then it sets $\mathsf{sk}_j = g^{Y_j}$ for $j \in [i-1]$.
- It simulates the leakage functions, adaptively, obtains the values $f_1(\mathsf{sk}_0, \mathsf{sk}_1), \ldots, f_{i-1}(\mathsf{sk}_{i-2}, \mathsf{sk}_{i-1})$, and obtains the next leakage function f_i.
- It computes $f_i(\mathsf{sk}_{i-1}, g^w)$, and then obtains the next leakage function f_{i+1}.
- Finally it outputs $f_i(\mathsf{sk}_{i-1}, g^w) || f_{i+1}(g^w, g^v)$.

Recall that f_i, f_{i+1} are two leakage functions with μ bits of output, so L^* has 2μ bits of output. Now we construct the adversary $\mathcal{B}$ as follows:

- Let g be the random generator, ω be the random coins as stated above, and L^* be the function defined above. Then $\mathcal{B}$ gets input $(A, X, L^*(A, Z, X \cdot T'), X \cdot T')$ where Z is either uniformly random or $X \cdot T$.
- $\mathcal{B}$ samples $Y_0, \ldots, Y_{i-1} \leftarrow \mathsf{Ker}^2(A)$, using the random coins ω. Then it sets $\mathsf{sk}_j = g^{Y_j}$ for $j \in [i-1]$. We note that the secret keys (in the first $i-1$ rounds) are consistent with the values used in the leakage function for they use the same randomness ω.
- $\mathcal{B}$ sets $\mathsf{sk}_{i+2} = g^{X \cdot T'}$.

- $\mathcal{B}$ samples $T_{i+3}, \ldots, T_{t+1} \leftarrow \mathsf{Rk}_2(\mathbb{Z}_p^{(\ell-3)\times 2})$ and sets $\mathsf{sk}_j = g^{X \cdot T_j}$ for $j \in \{i+3, \ldots, t+1\}$.
- $\mathcal{B}$ outputs $\mathcal{A}\Big(g^A, g^z, f_1(\mathsf{sk}_0, \mathsf{sk}_1), f_2(\mathsf{sk}_1, \mathsf{sk}_2), \cdots, f_{i-1}(\mathsf{sk}_{i-2}, \mathsf{sk}_{i-1}), L^*(Z, X \cdot T'), f_{i+2}(\mathsf{sk}_{i+2}, \mathsf{sk}'_{i+3}), \ldots, f_t(\mathsf{sk}'_t, \mathsf{sk}'_{t+1})\Big)$.

Then it is not hard to see that if Z comes from the distribution XT, then the simulation of $\mathcal{B}$ and L^* distributes as Q_i, and otherwise Q_{i-1}. Thus, suppose suppose $\mathcal{A}$ can distinguish Q_i from Q_{i+1} with non-negligible probability ϵ, then $\mathcal{B}$ can distinguish the two distributions with a non-negligible probability. This contradicts Lemma 6.

Finally, we show that D''_0 is computationally indistinguishable from D''_1 under the linear assumption.

Lemma 13 *Under the linear assumption, the distributions D''_0 and D''_1 are computationally indistinguishable.*

We use the same argument as the work [12]. In particular, we will prove that suppose there exists an adversary $\mathcal{A}$ that distinguishes D''_0 from D''_1, then there exists an adversary $\mathcal{B}$ that distinguishes the distributions $\{g^C : C \leftarrow \mathbb{Z}_p^{3\times 3}\}$ and $\{g^C : C \leftarrow \mathsf{Rk}_2(\mathbb{Z}_p^{3\times 3})\}$. We assume that the second distribution samples two random rows, and then sets the third row as a random linear combination of the first two rows. As argued in the work [12], this assumption is without loss of generality.

Now we describe the adversary $\mathcal{B}$. $\mathcal{B}$ on input g^C does the following.

- $\mathcal{B}$ samples a random matrix $X \leftarrow \mathbb{Z}_p^{\ell\times(\ell-3)}$, and a random matrix $B \leftarrow \mathbb{Z}_p^{3\times\ell}$ such that $B \cdot X = 0$.
- $\mathcal{B}$ computes g^{CB}, and sets its first two rows as g^A and the last row as g^z.
- $\mathcal{B}$ samples $T_1, \ldots, T_t \leftarrow \mathsf{Rk}_2(\mathbb{Z}_p^{(\ell-3)\times 2})$, and sets $\mathsf{sk}_i = g^{XT_i}$ for $i \in [t]$.
- $\mathcal{B}$ outputs $\mathcal{A}(g, g^A, g^z, f_1(\mathsf{sk}_0, \mathsf{sk}_1), \ldots, f_t(\mathsf{sk}_{t-1}, \mathsf{sk}_t))$.

As argued in the work [12], if C is uniformly random, then (A, z) is distributed uniformly as D''_1. If C is of rank 2, then (A, z) is distributed as $(A, r^T A)$ for some random $r \in \mathbb{Z}_p^2$ as D''_0. Thus, suppose $\mathcal{A}$ can distinguish D''_0 from D''_1 with non-negligible probability, then $\mathcal{B}$ breaks the linear assumption with non-negligible probability.

Lemma 9 ($D_0 \approx D_1$) follows directly from Lemmas 10, 12, and 13. This suffices to prove the theorem. We present the proofs of Lemmas 10, 12, and 13.

■

5 Bounded and continual leakage-resilient encryption schemes from Obfuscation

5.1 Making Sahai-Waters PKE Leakage-Resilient

We show that by modifying the Sahai-Waters (SW) public key encryption scheme [46] in two simple ways, the scheme already becomes non-trivially leakage resilient in the one-time, bounded setting. Recall that in this setting, the adversary, after seeing the public key and before seeing the challenge ciphertext, may request a single leakage query of length L bits. We require that semantic security hold, even given this leakage.

Our scheme can tolerate an arbitrary amount of one-time leakage. Specifically, for any $L = L(\kappa) = \mathrm{poly}(\kappa)$, we can obtain a scheme which is L-leakage resilient by setting the parameter ρ in Figure 6 depending on L. However, our leakage *rate* is not optimal, since the size of the secret key sk, grows with L. In Section 5.2, we will show how to further modify the construction to achieve optimal leakage rate.

At a high-level, we modify SW in the following ways: (1) Instead of following the general paradigm of encrypting a message m by xoring with the output of a PRF, we first apply a strong randomness extractor Ext to the output of the PRF and then xor with the message m; (2) We modify the secret key of the new scheme to be an iO of the underlying decryption circuit. Recall that in SW, decryption essentially consists of evaluating a puncturable PRF. In our scheme, sk consists of an iO of the puncturable PRF, padded with $\mathrm{poly}(L)$ bits.

We show that, even given L bits of leakage, the attacker cannot distinguish $\mathsf{Ext}(y)$ from random, where y is the output of the PRF on a fixed input t^*. This will be sufficient to prove security. We proceed by a sequence of hybrids: First, we switch sk to be an obfuscation of a circuit which has a PRF key punctured at t^* and a point function $t^* \to y$ hardcoded. On input $t \neq t^*$, the punctured PRF is used to compute the output, whereas on input t^*, the point function is used. Since the circuits compute the same function and—due to appropriate padding—they are both the same size, security of the iO implies that an adversary cannot distinguish the two scenarios. Next, just as in SW, we switch from $t^* \to y$ to $t^* \to y^*$, where y^* is uniformly random of length $L + L_{\mathsf{msg}} + 2\log(1/\epsilon)$ bits; here we rely on the security of the punctured PRF. Now, observe that since y^* is uniform and since Ext is a strong extractor for inputs of min-entropy $L_{\mathsf{msg}} + 2\log(1/\epsilon)$ and output length L_{msg}, $\mathsf{Ext}(y^*)$ looks random, even under L bits of leakage.

Encryption scheme $\mathcal{E} = (\mathcal{E}.\mathsf{Gen}, \mathcal{E}.\mathsf{Enc}, \mathcal{E}.\mathsf{Dec})$ using obfuscator iO and PRG G.

Key Generation: $(\mathsf{pk}, \mathsf{sk}_0) \leftarrow \mathcal{E}.\mathsf{Gen}(1^\kappa)$

Compute $k \leftarrow \mathsf{PRF.Gen}(1^\kappa)$, where $\mathsf{PRF} : \{0,1\}^\kappa \times \{0,1\}^\rho \to \{0,1\}^\rho$. Let C_k be the circuit described in Figure 7, and let $C_{\mathsf{Enc}} \leftarrow \mathsf{iO}(C_k)$.
Let $C_{k,\kappa+\rho}$ be the circuit described in Figure 8, and let $C_{\mathsf{Dec}} \leftarrow \mathsf{iO}(C_{k,\kappa+\rho})$.
Output $\mathsf{pk} = (C_{\mathsf{Enc}})$ and $\mathsf{sk} = (C_{\mathsf{Dec}})$.

Encryption: $c \leftarrow \mathcal{E}.\mathsf{Enc}(\mathsf{pk}, m)$

On input message $m \in \{0,1\}^{L_{\mathsf{msg}}}$, sample $r \leftarrow \{0,1\}^\kappa, w \leftarrow \{0,1\}^d$, and output $c = (G(r), w, \mathsf{Ext}(C_{\mathsf{Enc}}(r), w) \oplus m)$, where PRG $G : \{0,1\}^\kappa \to \{0,1\}^\rho$, and $\mathsf{Ext} : \{0,1\}^\rho \times \{0,1\}^d \to \{0,1\}^{L_{\mathsf{msg}}}$.

Decryption: $\hat{m} \leftarrow \mathcal{E}.\mathsf{Dec}(\mathsf{sk}, c)$

On input ciphertext $c = (t, w, v)$, compute $y := C_{\mathsf{Dec}}(t)$.

If $y \neq \perp$, output $\hat{m} = \mathsf{Ext}(y, w) \oplus v$. Otherwise, output $\hat{m} = \perp$.

Figure 6: The one-time, bounded leakage encryption scheme, $\mathcal{E}$.

Internal (hardcoded) state: k.

On input: r
- Output $z = \mathsf{PRF.Eval}(k, G(r))$, where G is the same PRG used in $\mathcal{E}.\mathsf{Enc}$.

Figure 7: This program C_k is obfuscated using iO and placed in the public key to be used for encryption.

Internal (hardcoded) state: k.

On input: t
- Output $z = \mathsf{PRF.Eval}(k, t)$.

Figure 8: The circuit above is padded with $\mathrm{poly}(\kappa + \rho)$ dummy gates to obtain the circuit $C_{k,\kappa+\rho}$. $C_{k,\kappa+\rho}$ is then obfuscated using iO and placed in the secret key.

Theorem 5.1 *Assume*

- $\mathsf{PRF} : \{0,1\}^\kappa \times \{0,1\}^\rho \to \{0,1\}^\rho$ *is a puncturable pseudorandom function.*

- iO *is indistinguishability obfuscator for circuits in this scheme.*
- $\mathsf{Ext} : \{0,1\}^{\rho} \times \{0,1\}^{d} \to \{0,1\}^{L_{\mathsf{msg}}}$ *is a* $(L_{\mathsf{msg}} + 2\log(1/\epsilon), \epsilon)$*-strong extractor, where* $\epsilon = \mathsf{negl}(\kappa)$.

Then $\mathcal{E}$ *is* L*-leakage resilient against one-time key leakage where*

$$L = \rho - 2\log(1/\epsilon) - L_{\mathsf{msg}}$$

First, note that extractors that satisfy the requirements of Theorem 5.1 can be constructed via the Leftover Hash Lemma (c.f. [36]).

In order to prove Theorem 5.1, we prove (in Lemma 14) that even under leakage, it is hard for any PPT adversary $\mathcal{A}$ to distinguish the output of the extractor, Ext from uniform random. Given this, Theorem 5.1 follows immediately.

Lemma 14 *For every* PPT *leaking adversary* $\mathcal{A}$*, who is given oracle access to a leakage oracle* $\mathcal{O}$ *and may leak at most* $\rho - 2\log(1/\epsilon) - L_{\mathsf{msg}}$ *bits of the secret key, there exist random variables* $\mathsf{pk}', \widetilde{\mathsf{sk}}$ *such that:*

$$\Big(\mathsf{pk}, t, w, \mathsf{Ext}(y, w), f(\mathsf{sk}) \leftarrow \mathcal{A}^{\mathcal{O}(\cdot)}(\mathsf{pk})\Big) \overset{c}{\approx} \Big(\mathsf{pk}', U_{\rho}, w, U_{L_{\mathsf{msg}}}, f(\widetilde{\mathsf{sk}}) \leftarrow \mathcal{A}^{\mathcal{O}(\cdot)}(\mathsf{pk}')\Big)$$

where $y = C_{\mathsf{Dec}}(t = G(r))$ *and the distributions are taken over coins of* $\mathcal{A}$ *and choice of* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathcal{E}.\mathsf{Gen}(1^{\kappa})$*,* w, r *and choice of* $\mathsf{pk}', \widetilde{\mathsf{sk}}, w$*, respectively.*

We prove the lemma via the following sequence of hybrids: Note that Hybrids 1, 2 are essentially identical to the Sahai-Waters hybrids. We differ from Sahai-Waters when we modify the secret key in Hybrids 3 and 4.

Hybrid 0: This hybrid is identical to the real game.

Let $\mathcal{D}^{\mathcal{A}}_{H_0}$ denote the distribution $(\mathsf{pk}, t, w, \mathsf{Ext}(y, w), f(\mathsf{sk}) \leftarrow \mathcal{A}^{\mathcal{O}(\cdot)}(\mathsf{pk}))$ as in the left side of Lemma 14.

Hybrid 1: This hybrid is the same as Hybrid 0 except we replace pseudorandom $t = G(r)$ in the challenge ciphertext with uniform random $t^* \leftarrow \{0,1\}^{\rho}$.

Let $\mathcal{D}^{\mathcal{A}}_{H_1}$ denote the distribution $(\mathsf{pk}, t^*, w, \mathsf{Ext}(y, w), f(\mathsf{sk}) \leftarrow \mathcal{A}^{\mathcal{O}(\cdot)}(\mathsf{pk}))$ where $y = C_{\mathsf{Dec}}(t^*)$ and the distribution is taken over coins of $\mathcal{A}$, choice of $(\mathsf{pk}, \mathsf{sk})$, w, t^* as described above.

Claim 5.2 *For every* PPT *adversary* $\mathcal{A}$*,*

$$\mathcal{D}^{\mathcal{A}}_{H_0} \overset{c}{\approx} \mathcal{D}^{\mathcal{A}}_{H_1}.$$

Proof: The proof is by reduction to the security of the pseudorandom generator G. Assume towards contradiction that there exists a PPT adversary $\mathcal{A}$, a corresponding PPT distinguisher D and a polynomial $p(\cdot)$ such that for infinitely many κ, D distinguishes $\mathcal{D}^{\mathcal{A}}_{H_0}$ and $\mathcal{D}^{\mathcal{A}}_{H_1}$ with probability at least $1/p(\kappa)$. We construct a PPT adversary $\mathcal{S}$ that distinguishes the output of the PRG from uniform random with probability at least $1/p(\kappa)$, for infinitely many κ. $\mathcal{S}$ does the following: $\mathcal{S}$ runs $\mathcal{E}.\mathsf{Gen}(1^{\kappa})$ honestly to generate $(\mathsf{pk}, \mathsf{sk})$. $\mathcal{S}$ hands pk to $\mathcal{A}$ and responds to leakage query f by applying the leakage function directly to sk to compute $f(\mathsf{sk})$. Upon receiving its challenge t' as the external PRG challenge, $\mathcal{S}$ sets $y = C_{\mathsf{Dec}}(t')$, hands $(\mathsf{pk}, t', w, \mathsf{Ext}(y, w), f(\mathsf{sk}))$ to the distinguisher D, and outputs whatever D does. The reader can verify that $\mathcal{S}$'s distinguishing advantage is the same as D's. ■

Hybrid 2: This hybrid is the same as Hybrid 1 except we replace the key k used in C_{Enc} with a punctured key, $\widetilde{k} = \mathsf{PRF.Punct}(k, t^*)$, and denote it as C'_{Enc}. We denote the resulting public key by pk'.

Let $\mathcal{D}^{\mathcal{A}}_{H_2}$ denote the distribution $\big(\mathsf{pk}', t^*, w, \mathsf{Ext}(y, w), f(\mathsf{sk}) \leftarrow \mathcal{A}^{\mathcal{O}(\cdot)}(\mathsf{pk}')\big)$ where $y = C_{\mathsf{Dec}}(t^*)$ and the distribution is taken over coins of $\mathcal{A}$, and choice of $(\mathsf{pk}', \mathsf{sk})$, w, t^* as described above.

Claim 5.3 *For every* PPT *adversary* $\mathcal{A}$*,*

$$\mathcal{D}^{\mathcal{A}}_{H_1} \overset{c}{\approx} \mathcal{D}^{\mathcal{A}}_{H_2}.$$

Internal (hardcoded) state: $\widetilde{k} = \mathsf{PRF.Punct}(k, t^*)$.

On input: r

– Output $z = \mathsf{PRF.Eval}(k, G(r))$, where G is the same PRG used in $\mathcal{E}.\mathsf{Enc}$.

Figure 9: Program $C_{\widetilde{k}}$. This program replaces C_k. It is obfuscated and placed in the public key to be used for encryption.

Proof: The proof is by a reduction to the security of the indistinguishability obfuscation. The main observation is that with all but negligible probability, t^* is not in the range of the PRG , in which case C_{Enc} and the modified circuit C'_{Enc} used in Hybrid 2 have identical behavior. Thus, with high probability for all inputs neither program can call on $\mathsf{PRF.Punct}(k, t^*)$. Therefore, puncturing t^* out from the key k will not effect the input/output behavior. Therefore, if there is a difference in advantage, we can create an algorithm $\mathcal{B}$ that breaks the security of indistinguishability obfuscation.

$\mathcal{B}$ runs as the challenger, but where t^* is chosen at random. When it is to create the obfuscated program it submits both programs $C_0 = C_k$ and $C_1 = C_{\widetilde{k}}$ to an iO challenger. If the iO challenger chooses the first then we are in Hybrid 1. If it chooses the second then we are in Hybrid 2. Any adversary with non-negligible advantages in the two hybrids leads to $\mathcal{B}$ as an attacker on iO security. ■

Hybrid 3: This hybrid is the same as Hybrid 2 except we replace $C_{\mathsf{Dec}} = \mathsf{iO}(C_{k,\kappa+\rho})$ with $C'_{\mathsf{Dec}} = \mathsf{iO}(C'_{\widetilde{k}})$, where $C'_{\widetilde{k}}$ is specified below in Figure 10. Note that we puncture k at the challenge point t^*. We denote by sk' the resulting secret key.

Internal (hardcoded) state: $(t^*, \beta = \mathsf{PRF.Eval}(k, t^*)), \widetilde{k} = \mathsf{PRF.Punct}(k, t^*)$.

On input: t

– If $t = t^*$, output β.

– Otherwise, output $\mathsf{PRF.Eval}(\widetilde{k}, t)$.

Figure 10: Program $C'_{\widetilde{k}}$. This program replaces $C_{k,\kappa+\rho}$. It is obfuscated and placed in the secret key.

Let $\mathcal{D}^{\mathcal{A}}_{H_3}$ denote the distribution $\left(\mathsf{pk}', t^*, w, \mathsf{Ext}(y, w), f(\mathsf{sk}') \leftarrow \mathcal{A}^{\mathcal{O}(\cdot)}(\widehat{\mathsf{pk}})\right)$ where $y = C'_{\mathsf{Dec}}(\, t^*)$ and the distribution is taken over coins of $\mathcal{A}$, and choice of $(\mathsf{pk}, \mathsf{sk}')$, w, t^* as described above.

Claim 5.4 *For every* PPT *adversary* $\mathcal{A}$*,*

$$\mathcal{D}^{\mathcal{A}}_{H_2} \stackrel{c}{\approx} \mathcal{D}^{\mathcal{A}}_{H_3}.$$

Proof: The proof is by a reduction to the security of the indistinguishability obfuscation. The main observation is that the size of the circuit does not change since the description of $C_{k,\kappa+\rho}$ is padded with $\mathrm{poly}(\kappa + \rho)$ gates (for appropriate poly). Thus, $C_{k,\kappa+\rho}$ and $C'_{\widetilde{k}}$ are the same size. Moreover, puncturing t^* out from the key k will not effect the input/output behavior since on input t^* we output the hardcoded value $\beta = \mathsf{PRF.Eval}(k, t^*)$. Therefore, if there is a difference in advantage, we can create an algorithm $\mathcal{B}$ that breaks the security of indistinguishability obfuscation.

$\mathcal{B}$ runs as the challenger, but where t^* is chosen at random. When it is to create the obfuscated program it submits both programs $C_0 = C_{k,\kappa+\rho}$ and $C_1 = C'_{\widetilde{k}}$ to an iO challenger. If the iO challenger chooses the first then we are in Hybrid 1. If it chooses the second then we are in Hybrid 2. Any adversary with non-negligible advantages in the two hybrids leads to $\mathcal{B}$ as an attacker on iO security. ■

Hybrid 4: This hybrid is the same as Hybrid 3 except we replace the hardcoded β with y^*, where y^* is uniformly random. We denote by $\widetilde{\mathsf{sk}}$ the resulting secret key. Note that the public key pk' remains the same.

Let $\mathcal{D}^{\mathcal{A}}_{H_4}$ denote the distribution $\left(\mathsf{pk}', t^*, w, \mathsf{Ext}(y^*, w), f(\widetilde{\mathsf{sk}}) \leftarrow \mathcal{A}^{\mathcal{O}(\cdot)}(\mathsf{pk}')\right)$ where $y^* = C'_{\mathsf{Dec}}(t^*)$ and the distribution is taken over coins of $\mathcal{A}$, and choice of $(\mathsf{pk}', \widetilde{\mathsf{sk}})$, w, t^* as described above.

Claim 5.5 *For every* PPT *adversary* $\mathcal{A}$*,*

$$\mathcal{D}^{\mathcal{A}}_{H_3} \overset{c}{\approx} \mathcal{D}^{\mathcal{A}}_{H_4}.$$

Proof: The proof is through a reduction to the security of the puncturable PRF. Recall, the security notion of puncturable PRFs states that, given $\mathsf{PRF.Punct}(k, t^*)$, an adversary cannot distinguish $\mathsf{PRF.Eval}(k, t^*)$ from random. The reduction is straightforward: to break the security of the PRF, $\mathcal{S}$ generates t^* at random and submits it to his challenger. He receives $\mathsf{PRF.Punct}(k, t^*)$, along with either $y^* = \mathsf{PRF.Eval}(k, t^*)$ or $y^* \leftarrow \{0,1\}^\rho$ as a challenge. He uses y^*, and samples all the remaining necessary keys for simulating $\widetilde{\mathsf{pk}}$ and $\widetilde{\mathsf{sk}}$. He chooses w at random and computes $\mathsf{Ext}(y^*, w)$. He answers leakage queries on $\widetilde{\mathsf{sk}}$ honestly. The reader can verify that $\mathcal{S}$'s advantage is the same as $\mathcal{A}$'s advantage in distinguishing the two hybrids. ■

Claim 5.6

$$\mathcal{D}^{\mathcal{A}}_{H_4} \overset{s}{\approx} \left(\mathsf{pk}', U_\rho, w, U_{L_{\mathsf{msg}}}, f(\widetilde{\mathsf{sk}}) \leftarrow \mathcal{A}^{\mathcal{O}(\cdot)}(\widetilde{\mathsf{pk}})\right)$$

Note that the right side above is the same as the right side of Lemma 14

Proof: We claim that the min-entropy of y^* conditioned on $\mathsf{pk}', f(\widetilde{\mathsf{sk}})$ is at least $L_{\mathsf{msg}} + 2\log(1/\epsilon)$. Note that y^* initially has min entropy ρ since it is chosen uniformly at random. Thus, leaking $\rho - 2\log(1/\epsilon) - L_{\mathsf{msg}}$ number of bits of the secret key reduces y^*'s min entropy by at most $\rho - 2\log(1/\epsilon) - L_{\mathsf{msg}}$. Therefore, y^* maintains min-entropy at least $L_{\mathsf{msg}} + 2\log(1/\epsilon)$. and so the claim follows by the properties of the strong extractor, Ext. ■

This concludes the proof of Lemma 14.

5.2 Improving the Leakage Rate

In this section, we show how to modify the previous construction to achieve optimal leakage rate. The key observation is that the leakage rate tolerated by the previous construction is low because the entire obfuscated circuit $\mathsf{iO}(C_{k,\kappa+\rho})$ must be stored in the secret key. Ideally, since the circuit is obfuscated, we would like to put it in the public key. However, this cannot possibly work since anyone can then decrypt the challenge ciphertext. Therefore, we store a collision-resistant hash $h(\mathsf{ct}_{\mathsf{dummy}})$ in the obfuscated circuit, and include a ciphertext encrypted using a symmetric key encryption scheme, $\mathsf{ct}_{\mathsf{dummy}}$, in the secret key: the circuit will only decrypt if the user provides a proper pre-image to the hardcoded value $h(\mathsf{ct}_{\mathsf{dummy}})$. This scheme seems to preserve semantic security, but we must prove security in the LR setting. Specifically, we must show that even when leaking $1 - o(1)$-fraction of $\mathsf{ct}_{\mathsf{dummy}}$, the adversary cannot find a valid input to the obfuscated circuit. To prove this, the idea is that in the hybrids, we switch $\mathsf{ct}_{\mathsf{dummy}}$ from a "dummy input" to an encryption of the point function $t^* \to y^*$, where y^* is random. The obfuscated circuit will also be changed (as in the proof of the previous construction) so that on input t^*, it outputs the output of the point function. Note that even under leakage, y^* has high min-entropy and thus $\mathsf{Ext}(y^*)$ will still look random. Finally, we note that in order for the argument to work, we must now rely on public-coin differing-inputs obfuscation, since in the hybrid arguments the obfuscated circuits in the public key will produce different outputs on inputs $\mathsf{ct}_{\mathsf{dummy}} \neq \mathsf{ct}'_{\mathsf{dummy}}$, such that $h(\mathsf{ct}'_{\mathsf{dummy}}) = h(\mathsf{ct}_{\mathsf{dummy}})$, which are hard for an efficient adversary to find.

Theorem 5.7 *Assume*

- E *is a semantically-secure symmetric key encryption scheme with ciphertexts of length* $L_{\mathsf{ct}}(\kappa, L_{\mathsf{msg}})$ *for* L_{msg}*-bit messages and security parameter* κ*.*
- h *is a collision-resistant hash function. with output length* $L_{\mathsf{h}}(\kappa)$ *for security parameter* κ*.*

Encryption Scheme $\mathcal{E} = (\mathcal{E}.\mathsf{Gen}, \mathcal{E}.\mathsf{Enc}, \mathcal{E}.\mathsf{Dec})$

Key Generation: $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathcal{E}.\mathsf{Gen}(1^\kappa)$

Compute the following:

- $(\mathsf{sk}_\mathsf{E}) \leftarrow \mathsf{E.Gen}(1^\kappa)$,
- $h \leftarrow \mathcal{H}$,
- $k \leftarrow \mathsf{PRF.Gen}(1^\kappa)$, where $\mathsf{PRF} : \{0,1\}^\kappa \times \{0,1\}^\rho \to \{0,1\}^\rho$.
- $\mathsf{ct}_\mathsf{dummy} \leftarrow \mathsf{E.Enc}(\mathsf{sk}_\mathsf{E}, 0^\kappa || 0^\rho; r_0)$, and $h^* = h(\mathsf{ct}_\mathsf{dummy})$.

Let C_k be the circuit described in Figure 12, and let $C_\mathsf{Enc} \leftarrow \mathsf{diO}(C_k)$.
Let $\mathsf{keys} = \{\mathsf{sk}_\mathsf{E}, h^*\}$, let C_keys be the circuit described in Figure 13, and let $C_\mathsf{Dec} \leftarrow \mathsf{diO}(C_\mathsf{keys})$.
Output $\mathsf{pk} = (C_\mathsf{Enc}, C_\mathsf{Dec})$ and $\mathsf{sk} = (\mathsf{ct}_\mathsf{dummy})$.

Encryption: $c \leftarrow \mathcal{E}.\mathsf{Enc}(\mathsf{pk}, m)$

On input message $m \in \{0,1\}^{L_\mathsf{msg}}$, sample $r \leftarrow \{0,1\}^\kappa, w \leftarrow \{0,1\}^d$, and output $c = (G(r), w, \mathsf{Ext}(C_\mathsf{Enc}(r), w) \oplus m)$, where PRG $G : \{0,1\}^\kappa \to \{0,1\}^\rho$, and $\mathsf{Ext} : \{0,1\}^\rho \times \{0,1\}^d \to \{0,1\}^{L_\mathsf{msg}}$.

Decryption: $\hat{m} \leftarrow \mathcal{E}.\mathsf{Dec}(\mathsf{sk}, c)$

On input ciphertext $c = (t, w, v)$, compute $y := C_\mathsf{Dec}(\mathsf{ct}_\mathsf{dummy}, t)$.

If $y \neq \perp$, output $\hat{m} = \mathsf{Ext}(y, w) \oplus v$. Otherwise, output $\hat{m} = \perp$.

Figure 11: The one-time, bounded leakage encryption scheme, $\mathcal{E}$.

Internal (hardcoded) state: k.

On input: r

– Output $z = \mathsf{PRF.Eval}(k, G(r))$, where G is the same PRG used in $\mathcal{E}.\mathsf{Enc}$.

Figure 12: This program C_k is obfuscated and placed in the public key to be used for encryption.

Internal (hardcoded) state: $\mathsf{keys} = \{k, h^*\}$.

On input: $\mathsf{ct}_\mathsf{dummy}, t$

– If $h(\mathsf{ct}_\mathsf{dummy}) \neq h^*$ output $\perp$.
– Otherwise, output $z = \mathsf{PRF.Eval}(k, t)$.

Figure 13: This program C_keys is obfuscated and placed in the public key. It is used during decryption.

- $\mathsf{PRF} : \{0,1\}^\kappa \times \{0,1\}^\rho \to \{0,1\}^\rho$ *is a puncturable pseudorandom function.*
- diO *is a public-coin, differing-input obfuscator for circuits in this scheme.*
- $\mathsf{Ext} : \{0,1\}^\rho \times \{0,1\}^d \to \{0,1\}^{L_\mathsf{msg}}$ *is a* $(L_\mathsf{msg} + 2\log(1/\epsilon), \epsilon)$*-strong extractor, where* $\epsilon = \mathsf{negl}(\kappa)$.

Then $\mathcal{E}$ *is* L*-leakage resilient against one-time key leakage where*

$$L = |\mathsf{sk}| \cdot \frac{\rho - 2\log(1/\epsilon) - L_\mathsf{msg} - L_\mathsf{h}(\kappa)}{(L_\mathsf{ct}(\kappa, \kappa + \rho))}$$

Proof: First, note that extractors that satisfy the requirements of Theorem 5.7 can be constructed via the Leftover Hash Lemma (c.f. [36]). We can choose a semantically-secure symmetric key encryption scheme with $L_{\mathsf{ct}}(\kappa, \kappa+\rho) = O(\kappa) + \kappa + \rho$, for messages of length $\kappa + \rho$, as this is achieved by appropriate modes of operation. Finally, choosing a collision-resistant hash function h with output length $L_{\mathsf{h}}(\kappa) = O(\kappa)$, and setting $\rho = \omega(\kappa)$, $\epsilon = 2^{o(\kappa)}$, $L_{\mathsf{msg}} = \Theta(\kappa)$, yields an encryption scheme for messages of length $\Theta(\kappa)$ with leakage rate $1 - o(1)$.

In order to prove Theorem 5.7, we prove (in Lemma 15) that even under leakage, it is hard for any PPT adversary $\mathcal{A}$ to distinguish the output of the extractor, Ext from uniform random. Given this, Theorem 5.7 follows immediately.

Lemma 15 *For every* PPT *leaking adversary $\mathcal{A}$, who is given oracle access to a leakage oracle $\mathcal{O}$ and may leak at most $\rho - 2\log(1/\epsilon) - L_{\mathsf{msg}}$ bits of the secret key, there exist random variables $\widetilde{\mathsf{pk}}, \widetilde{\mathsf{sk}}$ such that:*

$$\left(\mathsf{pk}, t, w, \mathsf{Ext}(y, w), f(\mathsf{sk}) \leftarrow \mathcal{A}^{\mathcal{O}(\cdot)}(\mathsf{pk})\right) \overset{c}{\approx} \left(\widetilde{\mathsf{pk}}, U_\rho, w, U_{L_{\mathsf{msg}}}, f(\widetilde{\mathsf{sk}}) \leftarrow \mathcal{A}^{\mathcal{O}(\cdot)}(\widetilde{\mathsf{pk}})\right)$$

where $y = C_{\mathsf{Dec}}(\mathsf{ct}_{\mathsf{dummy}}, t = G(r))$ and the distributions are taken over coins of $\mathcal{A}$ and choice of $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathcal{E}.\mathsf{Gen}(1^\kappa)$, w, r and choice of $\widetilde{\mathsf{pk}}, \widetilde{\mathsf{sk}}, w$, respectively.

We prove the lemma via the following sequence of hybrids:

Hybrid 0: This hybrid is identical to the real game.

Let $\mathcal{D}^{\mathcal{A}}_{H_0}$ denote the distribution $(\mathsf{pk}, w, \mathsf{Ext}(y, w), f(\mathsf{sk}) \leftarrow \mathcal{A}^{\mathcal{O}(\cdot)}(\mathsf{pk}))$ as in the left side of Lemma 15.

Hybrid 1: This hybrid is the same as Hybrid 0 except we replace pseudorandom $t = G(r)$ in the challenge ciphertext with uniform random $t^* \leftarrow \{0,1\}^\rho$.

Let $\mathcal{D}^{\mathcal{A}}_{H_1}$ denote the distribution $(\mathsf{pk}, t^*, w, \mathsf{Ext}(y, w), f(\mathsf{sk}) \leftarrow \mathcal{A}^{\mathcal{O}(\cdot)}(\mathsf{pk}))$ where $y = C_{\mathsf{Dec}}(\mathsf{ct}_{\mathsf{dummy}}, t^*)$ and the distribution is taken over coins of $\mathcal{A}$, choice of $(\mathsf{pk}, \mathsf{sk})$, w, t^* as described above.

Claim 5.8 *For every* PPT *adversary $\mathcal{A}$,*

$$\mathcal{D}^{\mathcal{A}}_{H_0} \overset{c}{\approx} \mathcal{D}^{\mathcal{A}}_{H_1}.$$

Proof: The proof is by reduction to the security of the pseudorandom generator G. Assume towards contradiction that there exists a PPT adversary $\mathcal{A}$, a corresponding PPT distinguisher D and a polynomial $p(\cdot)$ such that for infinitely many κ, D distinguishes $\mathcal{D}^{\mathcal{A}}_{H_0}$ and $\mathcal{D}^{\mathcal{A}}_{H_1}$ with probability at least $1/p(\kappa)$. We construct a PPT adversary $\mathcal{S}$ that distinguishes the output of the PRG from uniform random with probability at least $1/p(\kappa)$, for infinitely many κ. $\mathcal{S}$ does the following: $\mathcal{S}$ runs $\mathcal{E}.\mathsf{Gen}(1^\kappa)$ honestly to generate $(\mathsf{pk}, \mathsf{sk})$. $\mathcal{S}$ hands pk to $\mathcal{A}$ and responds to leakage query f by apply the leakage function directly to sk to compute $f(\mathsf{sk})$. Upon receiving its challenge t' as the external PRG challenge, $\mathcal{S}$ sets $y = C_{\mathsf{Dec}}(\mathsf{ct}_{\mathsf{dummy}}, t')$, hands $(\mathsf{pk}, t', w, \mathsf{Ext}(y, w), f(\mathsf{sk}))$ to the distinguisher D, and outputs whatever D does. The reader can verify that $\mathcal{S}$'s distinguishing advantage is the same as D's. ■

Hybrid 2: This hybrid is the same as Hybrid 1 except we replace the key k used in C_{Enc} with a punctured key, $\widetilde{k} = \mathsf{PRF.Punct}(k, t^*)$, and denote it as C'_{Enc}. We denote the resulting public key by pk'.

Let $\mathcal{D}^{\mathcal{A}}_{H_2}$ denote the distribution $\left(\mathsf{pk}', t^*, w, \mathsf{Ext}(y, w), f(\mathsf{sk}) \leftarrow \mathcal{A}^{\mathcal{O}(\cdot)}(\mathsf{pk}')\right)$ where $y = C_{\mathsf{Dec}}(\mathsf{ct}_{\mathsf{dummy}}, t^*)$ and the distribution is taken over coins of $\mathcal{A}$, and choice of $(\mathsf{pk}', \mathsf{sk})$, w, t^* as described above.

Claim 5.9 *For every* PPT *adversary $\mathcal{A}$,*

$$\mathcal{D}^{\mathcal{A}}_{H_1} \overset{c}{\approx} \mathcal{D}^{\mathcal{A}}_{H_2}.$$

Proof: The proof is by a reduction to the security of the indistinguishability obfuscation (iO). The main observation is that with all but negligible probability, t^* is not in the range of the PRG , in which case C_{Enc} and the modified circuit C'_{Enc} used in Hybrid 2 have identical behavior. Thus, with high probability for all inputs

Internal (hardcoded) state: $\widetilde{k} = \mathsf{PRF.Punct}(k, t^*)$.

On input: r

– Output $z = \mathsf{PRF.Eval}(k, G(r))$, where G is the same PRG used in $\mathcal{E}.\mathsf{Enc}$.

Figure 14: Program $C_{\widetilde{k}}$. This program replaces C_k. It is obfuscated and placed in the public key to be used for encryption.

neither program can call on $\mathsf{PRF.Punct}(k, t^*)$. Therefore, puncturing t^* out from the key k will not effect the input/output behavior. Therefore, if there is a difference in advantage, we can create an algorithm $\mathcal{B}$ that breaks the security of indistinguishability obfuscation.

$\mathcal{B}$ runs as the challenger, but where t^* is chosen at random. When it is to create the obfuscated program it submits both programs $C_0 = C_k$ and $C_1 = C_{\widetilde{k}}$ to an iO challenger. If the iO challenger chooses the first then we are in Hybrid 1. If it chooses the second then we are in Hybrid 2. Any adversary with non-negligible advantages in the two hybrids leads to $\mathcal{B}$ as an attacker on iO security. ■

Hybrid 3: This hybrid is the same as Hybrid 2 except:

- we replace $\mathsf{ct}_{\mathsf{dummy}}$ with $\mathsf{ct}''_{\mathsf{dummy}}$, where $\mathsf{ct}''_{\mathsf{dummy}}$ is an encryption of $t^*||y$ and $y = \mathsf{PRF.Eval}(k, t^*)$.
- we replace h^* with $h''^* = h(\mathsf{ct}''_{\mathsf{dummy}})$.

We denote the resulting public key by pk'' and the resulting secret key by sk''.

Let $\mathcal{D}^{\mathcal{A}}_{H_3}$ denote the distribution $\left(\mathsf{pk}'', t^*, w, \mathsf{Ext}(y, w), f(\mathsf{sk}'') \leftarrow \mathcal{A}^{\mathcal{O}(\cdot)}(\mathsf{pk}'')\right)$ where $y = C_{\mathsf{Dec}}(\mathsf{ct}''_{\mathsf{dummy}}, t^*)$ and the distribution is taken over coins of $\mathcal{A}$, and choice of $(\mathsf{pk}'', \mathsf{sk}'')$, w, t^* as described above.

Claim 5.10 *For every* PPT *adversary* $\mathcal{A}$*,*

$$\mathcal{D}^{\mathcal{A}}_{H_2} \overset{c}{\approx} \mathcal{D}^{\mathcal{A}}_{H_3}.$$

The proof is by a reduction to the semantic security of E.

Hybrid 4: This hybrid is the same as Hybrid 3 except we replace $C_{\mathsf{Dec}} = \mathsf{diO}(C_{\mathsf{keys}})$ with $C'_{\mathsf{Dec}} = \mathsf{diO}(C_{\mathsf{keys}'})$, where $C_{\mathsf{keys}'}$ is specified below in Figure 15. We denote by $\widehat{\mathsf{pk}}$ the resulting public key.

Internal (hardcoded) state: $\mathsf{keys}' = \{\mathsf{sk}_{\mathsf{E}}, \widetilde{k} = \mathsf{PRF.Punct}(k, t^*), h''^*\}$.

On input: $\mathsf{ct}_{\mathsf{dummy}}, t$

– If $h(\mathsf{ct}_{\mathsf{dummy}}) \neq h''^*$ output $\perp$.

– Compute $\alpha||\beta = \mathsf{E.Dec}(\mathsf{sk}_{\mathsf{E}}, \mathsf{ct}_{\mathsf{dummy}})$.

– If $t = \alpha$, output β.

– Otherwise, output $\mathsf{PRF.Eval}(\widetilde{k}, t)$.

Figure 15: Program $C_{\mathsf{keys}'}$. This program replaces C_{keys}. It is obfuscated and placed in the public key. It is used during decryption.

Let $\mathcal{D}^{\mathcal{A}}_{H_4}$ denote the distribution $\left(\widehat{\mathsf{pk}}, t^*, w, \mathsf{Ext}(y, w), f(\mathsf{sk}'') \leftarrow \mathcal{A}^{\mathcal{O}(\cdot)}(\widehat{\mathsf{pk}})\right)$ where $y = C'_{\mathsf{Dec}}(\mathsf{ct}''_{\mathsf{dummy}}, t^*)$ and the distribution is taken over coins of $\mathcal{A}$, and choice of $(\widehat{\mathsf{pk}}, \mathsf{sk}'')$, w, t^* as described above.

Claim 5.11 *For every* PPT *adversary* $\mathcal{A}$*,*

$$\mathcal{D}^{\mathcal{A}}_{H_3} \overset{c}{\approx} \mathcal{D}^{\mathcal{A}}_{H_4}.$$

Proof: We define the following sampler Samp and show that the circuit family $\mathcal{C}$ associated with Samp is a differing-inputs circuit family.
$\mathsf{Samp}(1^\kappa)$ does the following:

- Set $\mathsf{keys} = (k, h''^*)$ and set $\mathsf{keys}' = (\mathsf{sk}_\mathsf{E}, \widetilde{k}, h''^*)$.
- Let $C_0 = C_{\mathsf{keys}}$ and let $C_1 = C_{\mathsf{keys}'}$.
- Set $\mathsf{aux} = (\mathsf{sk}_\mathsf{E}, h, h''^*, \mathsf{ct}''_{\mathsf{dummy}}, r\ t^*, y)$, where r is the randomness used for $\mathsf{ct}''_{\mathsf{dummy}}$.
- Return (C_0, C_1, aux).

Note that aux contains all of the random coins used by Samp.
We now show that for every PPT adversary $\mathcal{A}$ there exists a negligible function negl such that

$$\Pr[C_0(x) \neq C_1(x) : (C_0, C_1, \mathsf{aux}) \leftarrow \mathsf{Samp}(1^\kappa), x \leftarrow \mathcal{A}(1^\kappa, C_0, C_1, \mathsf{aux})] \leq \mathsf{negl}(\kappa).$$

Assume towards contradiction that there exists a PPT adversary $\mathcal{A}$ and a polynomial $p(\cdot)$ such that for infinitely many κ, $\mathcal{A}$ outputs a distinguishing input with probability at least $1/p(\kappa)$. We construct a PPT adversary $\mathcal{S}$ that finds a collision on h.
On input $h \leftarrow \mathcal{H}$, $\mathcal{S}$ does the following:

- $\mathcal{S}$ simulates Samp by doing the following:
  - Run $(\mathsf{sk}_\mathsf{E}) \leftarrow \mathsf{E.Gen}(1^\kappa)$, $k \leftarrow \mathsf{PRF.Gen}(1^\kappa)$.
    Choose t^* at random and set $\widetilde{k} = \mathsf{PRF.Punct}(k, t^*)$.
  - $\mathcal{S}$ computes $y = \mathsf{PRF.Eval}(k, t^*)$ to generate $\mathsf{ct}''_{\mathsf{dummy}}$. It computes $h^* = h(\mathsf{ct}''_{\mathsf{dummy}})$.
  - Set $\mathsf{keys} = (k, h''^*)$ and $\mathsf{keys}' = (\mathsf{sk}_\mathsf{E}, \widetilde{k}, h''^*)$.
  - Let $C_0 = C_{\mathsf{keys}}$ and let $C_1 = C_{\mathsf{keys}'}$.
  - Set $\mathsf{aux} = (\mathsf{sk}_\mathsf{E}, h, h''^*\ \mathsf{ct}''_{\mathsf{dummy}}, r\ t^*, y)$.
- $\mathcal{S}$ runs $\mathcal{A}(1^\kappa, C_0, C_1, \mathsf{aux})$ and receives x in return.
- $\mathcal{S}$ parses x as (m, t) and outputs (m).

Note that $C_0(\mathsf{ct}''_{\mathsf{dummy}}, \cdot)$ and $C_1(\mathsf{ct}''_{\mathsf{dummy}}, \cdot)$ are functionally equivalent. Furthermore, on any input (m, t) where $h(m) \neq h''^*$, both circuits output $\perp$. Therefore, if $\mathcal{A}$ finds a distinguishing input $x = (m, t)$, then it must be the case that both of the following hold:

- $(m \neq \mathsf{ct}''_{\mathsf{dummy}})$
- $h(m) = h''^*$.

Thus, whenever $\mathcal{A}$ outputs a differing input, $\mathcal{S}$ successfully finds a collision on h. Therefore, we have that for infinitely many κ, $\mathcal{S}$ outputs a collision with probability at least $1/p(\kappa)$.

Claim 5.11 follows from the fact that diO is a public-coin differing-inputs obfuscator and from the fact that the circuit family $\mathcal{C}$ associated with Samp is a differing inputs family. This is the case since $\mathcal{D}^{\mathcal{A}}_{H_3}$ can be simulated given $(\mathsf{diO}(C_0), \mathsf{aux})$ and $\mathcal{D}^{\mathcal{A}}_{H_4}$ can be simulated given $(\mathsf{diO}(C_1), \mathsf{aux})$. ∎

Hybrid 5: This hybrid is the same as Hybrid 4 except we replace $\mathsf{ct}''_{\mathsf{dummy}}$ with $\widetilde{\mathsf{ct}}_{\mathsf{dummy}}$, where $\widetilde{\mathsf{ct}}_{\mathsf{dummy}}$ is an encryption of $t^*||y^*$, where y^* is uniformly random. We denote by $\widetilde{\mathsf{sk}}$ the resulting secret key. We replace h^* with $\widetilde{h}^* = h(\widetilde{\mathsf{ct}}_{\mathsf{dummy}})$ and denote by $\widetilde{\mathsf{pk}}$ the updated public key.

Let $\mathcal{D}^{\mathcal{A}}_{H_5}$ denote the distribution $\left(\widetilde{\mathsf{pk}}, t^*, w, \mathsf{Ext}(y^*, w), f(\widetilde{\mathsf{sk}}) \leftarrow \mathcal{A}^{\mathcal{O}(\cdot)}(\widetilde{\mathsf{pk}})\right)$ where $y^* = C'_{\mathsf{Dec}}(\widetilde{\mathsf{ct}}_{\mathsf{dummy}}, t^*)$ and the distribution is taken over coins of $\mathcal{A}$, and choice of $(\widetilde{\mathsf{pk}}, \widetilde{\mathsf{sk}})$, w, t^* as described above.

Claim 5.12 *For every* PPT *adversary* $\mathcal{A}$*,*

$$\mathcal{D}^{\mathcal{A}}_{H_4} \stackrel{c}{\approx} \mathcal{D}^{\mathcal{A}}_{H_5}.$$

Proof: The proof is through a reduction to the security of the puncturable PRF. Recall, the security notion of puncturable PRFs states that, given $\mathsf{PRF.Punct}(k, t^*)$, an adversary cannot distinguish $\mathsf{PRF.Eval}(k, t^*)$ from random. The reduction is straightforward: to break the security of the PRF, $\mathcal{S}$ generates t^* at random and submits it to his challenger. He receives $\mathsf{PRF.Punct}(k, t^*)$, along with either $y^* = \mathsf{PRF.Eval}(\kappa, t^*)$ or $y^* \leftarrow \{0,1\}^\rho$ as a challenge. He uses y^*, and samples all the remaining necessary keys for simulating $\widehat{\mathsf{pk}}$ and $\widehat{\mathsf{sk}}$. He chooses w at random and computes $\mathsf{Ext}(y^*, w)$. He answers leakage queries on $\widehat{\mathsf{sk}}$ honestly. The reader can verify that $\mathcal{S}$'s advantage is the same as $\mathcal{A}$'s advantage in distinguishing the two hybrids. ■

Claim 5.13

$$\mathcal{D}^{\mathcal{A}}_{H_5} \stackrel{s}{\approx} \left(\widetilde{\mathsf{pk}}, U_\rho, w, U_{L_{\mathsf{msg}}}, f(\widetilde{\mathsf{sk}}) \leftarrow \mathcal{A}^{\mathcal{O}(\cdot)}(\widetilde{\mathsf{pk}})\right)$$

Note that the right side above is the same as the right side of Lemma 15

Proof: We claim that the min-entropy of y^* conditioned on $\widetilde{\mathsf{pk}}, f(\widetilde{\mathsf{sk}})$ is at least $L_{\mathsf{msg}} + 2\log(1/\epsilon)$. Note that y^* initially has min entropy ρ since it is chosen uniformly at random. Recall that, $\mathsf{ct}_{\mathsf{dummy}}$ has length $L_{\mathsf{ct}}(\kappa, \kappa+\rho)$, and h has output length $L_{\mathsf{h}}(\kappa)$. Thus, conditioning on $\widetilde{\mathsf{pk}}$ reduces y^*'s min entropy by at most $L_{\mathsf{h}}(\kappa)$ (since only $\widetilde{h}^*$ contains information about y^*). Moreover, leaking another $\rho - 2\log(1/\epsilon) - L_{\mathsf{msg}} - L_{\mathsf{h}}(\kappa)$ number of bits of $\mathsf{ct}_{\mathsf{dummy}}$ reduces y^*'s min entropy further by at most $\rho - 2\log(1/\epsilon) - L_{\mathsf{msg}}$. Therefore, y^* maintains min-entropy at least $L_{\mathsf{msg}} + 2\log(1/\epsilon)$, and the claim follows by the properties of the strong extractor, Ext. ■

This concludes the proof of Lemma 15. ■

5.3 Extending to Continual Leakage

In this section, we show how to extend our "leakage-resilient" version of the Sahai-Waters PKE scheme to continual leakage. Recall that in the previous construction, the secret key consists of a signed ciphertext. An initial idea for how to achieve (consecutive) CLR is to refresh the secret key by *re-randomizing* the ciphertext. Specifically, in the real construction, the ciphertext in the secret key just encrypts zeros and is rerandomized, but in the proof it contains the PRF output and the underlying plaintext is refreshed from round to round. However, since the underyling plaintext changes from round to round, we run into some technical difficulties. Specifically, an adversary who knows the underlying secret key for the ciphertext embedded in the construction's secret key will be able to distinguish consecutive hybrids. On the other hand, an adversary who does not know the secret key for the ciphertext will not be able to produce a correctly distributed obfuscated circuit to be placed in the public key. To resolve this, the idea is to use an encryption scheme with special properties: The challenge ciphertext remains semantically secure while at the same time, the adversary can efficiently *simulate* a decryption oracle which either successfully decrypts the submitted ciphertext or indicates that the submitted ciphertext is a re-randomization of the challenge ciphertext. Note that this notion is a strengthening of the notion of re-randomizable RCCA (relaxed CCA) security. Specifically, we define a new special "DIO-compatible" notion of "relaxed" [15] or "controlled" malleability CCA security for rerandomizable encryption [17]. We then show how to realize our new notion from the decision linear (DLIN) assumption in bilinear groups, following [17]. Our resulting continual leakage-resilient scheme presented in Section 5.3.4 combines this assumption with diO.

5.3.1 Rerandomizable Encryption

A *rerandomizable encryption scheme* is a tuple of algorithms $\mathsf{RPKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{ReRand})$ defined as follows. First, the triple $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is a standard encryption scheme. Second,

$$(\mathrm{PK}, \mathrm{SK}, c) \approx_c (\mathrm{PK}, \mathrm{SK}, c')$$

where

$$(\mathrm{PK}, \mathrm{SK}) \leftarrow \mathsf{Gen}(1^\kappa)\ ;\ c \leftarrow \mathsf{Enc}(\mathrm{PK}, m)\ ;\ c' \leftarrow \mathsf{ReRand}(\mathrm{PK}, c)\ .$$

5.3.2 DIO-Compatible RCCA Encryption

Intuitively, an RCCA encryption scheme [15] is like a CCA-secure encryption scheme that allows *replay attacks*. We define a special type of RCCA encryption scheme that we call *DIO compatible*, inspired by the definition of *controlled malleability* of Chase et al. [17]. The intuition for DIO-compatible RCCA encryption, is that the security proof allows for placing the secret key of the encryption scheme in an obfuscation, while arguing about semantic security of a challenge ciphertext. Specifically, the way this is achieved is by requiring that the DIO-compatible RCCA encryption have a specific hybrid structure for proving security. In each hybrid there is an efficient algorithm which simulates the decryption oracle. Moreover, indistinguishability of consecutive hybrids reduces either to the security of an underlying primitive, or reduces to the fact that, even given the code of the simulated decryption oracle, the attacker cannot find a *distinguishing input*. A detailed definition follows.

Syntax. A *DIO-compatible RCCA encryption scheme* is a tuple of algorithms $\mathsf{RCCA} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{SimEnc}, \mathsf{Dec})$ where

- Algorithm $\mathsf{Gen}(1^\kappa)$ outputs keys $(\mathrm{PK}, (\mathrm{SK}_1, \mathrm{SK}_2), \tau_{\mathsf{sim}})$.
- Algorithm $\mathsf{Enc}(\mathrm{PK}, m)$ outputs a ciphertext c.
- Algorithm $\mathsf{SimEnc}(\tau_{\mathsf{sim}}, m)$ outputs a ciphertext c_{sim}.
- Algorithm $\mathsf{Dec}(\mathrm{SK}_1, c)$ outputs a message value in $\{m, \perp\}$.
- Algorithm $\mathsf{Dec}(\mathrm{SK}_2, c)$ outputs a message value in $\{m, \perp, \mathsf{SimFlag}\}$.

Correctness. For correctness we require that $\Pr[\mathsf{Dec}(\mathrm{SK}_1, \mathsf{Enc}(\mathrm{PK}, m)) = m] = 1$ where $(\mathrm{PK}, (\mathrm{SK}_1, \mathrm{SK}_2), \tau_{\mathsf{sim}})$ is output by $\mathsf{Gen}(1^\kappa)$. (This can be relaxed to allow negligble probability of failure.)

Security. For security we require[10]

- **Indistinguishability of simulated ciphertexts from real ciphertexts:** For any efficient adversary A and message m

$$A(\mathrm{PK}, \mathsf{Enc}(\mathrm{PK}, m), \mathrm{SK}_1) \stackrel{c}{\approx} A(\mathrm{PK}, \mathsf{SimEnc}(\tau_{\mathsf{sim}}, m), \mathrm{SK}_1)\ .$$

- **Indistinguishability of simulated ciphertexts under chosen plaintext attack:** For any efficient adversary A, and any message pair (m_0, m_1),

$$A(\mathrm{PK}, \mathrm{SK}_2, \mathsf{SimEnc}(\tau_{\mathsf{sim}}, m_0)) \stackrel{c}{\approx} A(\mathrm{PK}, \mathrm{SK}_2, \mathsf{SimEnc}(\tau_{\mathsf{sim}}, m_1))\ .$$

- **Simulation soundness:** For any efficient A and message m the probability that the following experiment outputs 1 is negligible:

  $(\mathrm{PK}, (\mathrm{SK}_1, \mathrm{SK}_2), \tau_{\mathsf{sim}}) \leftarrow \mathsf{Gen}(1^\kappa)\ ;\ c \leftarrow \mathsf{SimEnc}(\tau_{\mathsf{sim}}, m)\ ;\ c^* \leftarrow A(\mathrm{SK}_1, \mathrm{SK}_2, c)$
  Return 1 if

[10] We note that the presented conditions only refer to *single-message security*, which is all that is needed in our applications. More generally they could provide the adversary many challenge encryptions (say, via appropriate encryption oracles).

1. $\mathsf{Dec}(\mathrm{SK}_2, c^*) = \mathsf{SimFlag} \bigwedge \mathsf{Dec}(\mathrm{SK}_1, c^*) \neq m$
2. $\mathsf{Dec}(\mathrm{SK}_2, c^*) \neq \mathsf{SimFlag} \bigwedge \mathsf{Dec}(\mathrm{SK}_2, c^*) \neq \mathsf{Dec}(\mathrm{SK}_1, c^*)$

where the probability is taken over the randomness used in key generation, SimEnc, and by A when computing c^*.

Rerandomizability. We say that a DIO-compatible RCCA-secure encryption scheme is *rerandomizable* if it has an additional algorithm ReRand defined analogously to rerandomizable encryption.

5.3.3 A Construction of DIO-Compatible RCCA-secure Rerandomizable PKE

In our extension to continual leakage, we need a DIO-Compatible RCCA-secure *rerandomizable* scheme. We now sketch how a CM-CCA encryption scheme of Chase et al. [17] instantiates this notion based on the Decision-Linear Assumption [7]. (Compared to their security analysis, we prove something stronger since we require an RCCA scheme of a particular "DIO-compatible" form, but we use the same RCCA construction of [17] and what we require follows by basically the same analysis.)

The instantiation uses a rerandomizable encryption scheme and a matching rerandomizable CM-NIZK proof system as defined in [17]. By "matching" we mean that the NIZK proof system is controlled-malleable with respect to whatever operation the underlying encryption scheme uses for rerandomization.

The Instantiation. For simplicity, we assume the NIZK in the scheme is "same-string" (the CRS does not change distribution); otherwise, the definition in Section 5.3.2 needs to be extended a bit (in a straightforward but messy way). For our construction we have $\mathsf{RCCA} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{SimEnc}, \mathsf{Dec})$ as follows:

- Algorithm $\mathsf{Gen}(1^\kappa)$ outputs the public-key of the encryption scheme and the NIZK CRS as the PK. It outputs the decryption key of the encryption scheme as SK_1 and the NIZK extraction trapdoor as SK_2. It outputs the NIZK simulation trapdoor as τ_{sim}.
- Algorithm $\mathsf{Enc}(\mathrm{PK}, m)$ honestly encrypts message using the encryption scheme as c^1 and gives an honest proof of plaintext knowledge as c^2; it outputs ciphertext $c = c^1 \| c^2$.
- Algorithm $\mathsf{SimEnc}(\tau_{\mathsf{sim}}, m)$ honestly encrypts a "dummy message" as c^1_{sim} and gives a *simulated* proof of plaintext knowledge as c^2_{sim}; it outputs ciphertext $c = c^1_{\mathsf{sim}} \| c^2_{\mathsf{sim}}$.
- Algorithm $\mathsf{Dec}_1(\mathrm{SK}_1, c = c^1 \| c^2)$ decrypts c^1 using SK_1 and outputs the result if c^2 verifies (which one can check publicly).
- Algorithm $\mathsf{Dec}(\mathrm{SK}_2, c = c^1 \| c^2)$ runs the NIZK extraction on c^2 using SK_2. If it fails then it outputs $\mathsf{SimFlag}$, otherwise it outputs the result of the extraction.

Security. The fact that our instantiation satisfies our security requirements follows readily from the analysis of [17, Appendix F]. In particular, simulation soundness uses the fact that simulation soundness of the underlying NIZK from [17] holds even when the adversary is given the extraction trapdoor.

Concrete parameters. As the underlying rerandomizable encryption scheme we use the DLIN-based encryption scheme of BBS [7] (which is rerandomizable via exponentiation), and as the NIZK proof system as in [17] we combine a a Groth-Sahai proof of plaintext knowledge (which is itself rerandomizable and supports exponentiation malleability of the underlying statement) along with the signature scheme from Abe et al. [1]. The overall ciphertext in our resulting DIO-Compatible RCCA-secure Rerandomizable PKE scheme contains a constant number of group elements, although this constant is large (on the order of several hundred).

5.3.4 Our Construction

To guarantee security in the presence of continual leakage, we modify our construction in two ways. First, we strengthen the security of the encryption scheme used to encrypt $\mathsf{ct}_{\mathsf{dummy}}$, requiring that it provide relaxed CCA security (RCCA) [15]. Recall that such encryption schemes allow users to re-randomize ciphertexts, while guaranteeing the ciphertexts are otherwise secure against chosen ciphertext attacks.

The other change that we make to the scheme of Section 5.2 comes up in the proof of security. Because we are leaking over multiple rounds, storing t^* in a list no longer suffices. After enough rounds, the value will be fully recovered by the adversary, and he will distinguish neighboring hybrids. To fix this, we instead store random values whose inner product yields the challenge point, along with a hash of the challenge. In our proof of security, the most interesting hybrids are hybrid 4, where use RCCA security, hybrid 5, where we reduce to diO, and hybrid 8, where we use the fact that inner product is a good 2-source (and, therefore, strong) extractor. The other hybrids are fairly straightforward.

Encryption Scheme $\mathcal{E} = (\mathcal{E}.\mathsf{Gen}, \mathcal{E}.\mathsf{Enc}, \mathcal{E}.\mathsf{Dec}, \mathcal{E}.\mathsf{Update})$

Key Generation: $(\mathsf{pk}, \mathsf{sk}_0) \leftarrow \mathcal{E}.\mathsf{Gen}(1^\kappa)$

Compute the following:

- $(\mathrm{PK}_{\mathsf{RCCA}}, \mathrm{SK}_{\mathsf{RCCA}}) \leftarrow \mathsf{RCCA.Gen}(1^\kappa)$,
- $(\mathsf{vk}, \mathsf{td}) \leftarrow \mathsf{SIG.Gen}(1^\kappa)$,
- $k \leftarrow \mathsf{PRF.Gen}(1^\kappa)$.
- $\sigma \leftarrow \mathsf{SIG.Sign}(\mathsf{td}, 0^{2\kappa+\rho+L_{\mathsf{msg}}})$.
- $\mathsf{ct}_{\mathsf{dummy}} \leftarrow \mathsf{RCCA.Enc}(\mathrm{PK}_{\mathsf{RCCA}}, \sigma||0^{2\kappa+\rho+L_{\mathsf{msg}}}; r_0)$, where $r_0 \leftarrow \{0,1\}^\kappa$.

Let C_k be the circuit described in Figure 17, and compute $C_{\mathsf{Enc}} \leftarrow \mathsf{diO}(C_k)$.
Let $\mathsf{keys} = \{\mathrm{SK}_{\mathsf{RCCA}}, k, \mathsf{vk}\}$, let C_{keys} be the circuit in Figure 18, and compute $C_{\mathsf{Dec}} \leftarrow \mathsf{diO}(C_{\mathsf{keys}})$.
Output $\mathsf{pk} = (\mathrm{PK}_{\mathsf{RCCA}}, C_{\mathsf{Enc}}, C_{\mathsf{Dec}})$ and $\mathsf{sk}_0 = (\mathsf{ct}_{\mathsf{dummy}})$.

Encryption: $c \leftarrow \mathcal{E}.\mathsf{Enc}(\mathsf{pk}, m)$

On input message $m \in \{0,1\}^{L_{\mathsf{msg}}}$, sample $r \leftarrow \{0,1\}^\kappa$, and output $c = (G(r), C_{\mathsf{Enc}}(r) \oplus m)$, where G is some fixed pseudorandom generator.

Decryption: $\hat{m} \leftarrow \mathcal{E}.\mathsf{Dec}(\mathsf{sk}_i, c)$

In round i, on input ciphertext $c = (t, v)$, compute $y := C_{\mathsf{Dec}}(\mathsf{sk}_i, t)$.
If $y \neq \perp$, output $\hat{m} = y \oplus v$. Otherwise, output $\hat{m} = \perp$.

Key Update: $\mathsf{sk}_i \leftarrow \mathcal{E}.\mathsf{Update}(\mathsf{sk}_{i-1})$

In round i, on input secret key sk_{i-1}, randomly choose $r_i \leftarrow \{0,1\}^\kappa$, compute and output $\mathsf{sk}_i \leftarrow \mathsf{RCCA.ReRand}(\mathrm{PK}_{\mathsf{RCCA}}, \mathsf{sk}_{i-1}; r_i)$.

Figure 16: The continual-leakage resistant encryption scheme, $\mathcal{E}$.

Theorem 5.14 *Assume*

- RCCA *is a DIO-Compatible RCCA-secure Rerandomizable PKE with ciphertexts of length* $L_{\mathsf{ct}}(\kappa, L_{\mathsf{msg}})$ *for* L_{msg}*-bit messages and security parameter* κ*.*
- SIG *is a strong existentially unforgeable digital signature scheme with signatures of length* $L_{\mathsf{sig}}(\kappa, L_{\mathsf{msg}})$ *for* L_{msg}*-bit messages and security parameter* κ*.*

Internal (hardcoded) state: k.

On input: r

– Output $z = \mathsf{PRF.Eval}(k, G(r))$, where G is some fixed pseudorandom generator.

Figure 17: Program C_k. This program is obfuscated and placed in the public key to be used for encryption.

Internal (hardcoded) state: $\mathsf{keys} = \{k, \mathsf{vk}, \mathrm{SK}_{\mathsf{RCCA}}\}$.

On input: $\mathsf{ct}_{\mathsf{dummy}}, t$

– Compute $(\sigma', m) = \mathsf{RCCA.Dec}(\mathrm{SK}_{\mathsf{RCCA}}, \mathsf{ct}_{\mathsf{dummy}})$.

– If $\mathsf{SIG.Verify}(\sigma', m; \mathsf{vk}) = 0$ output $\perp$.

– Output $z = \mathsf{PRF.Eval}(k, t)$.

Figure 18: Program C_{keys}. This program is obfuscated and placed in the public key. It is used during decryption.

- PRF *is a puncturable pseudorandom function* $\{0,1\}^\kappa \times \{0,1\}^\rho \to \{0,1\}^{L_{\mathsf{msg}}}$ *for some* $\rho = \rho(\kappa) = \omega(\kappa^2)$.
- G *is a pseudorandom generator* $\{0,1\}^\kappa \to \{0,1\}^\rho$.
- diO *is a differing-inputs obfuscator for circuits in this scheme.*
- $\mathcal{H}$ *is a family of collision resistant hash functions with output size* κ *bits.*

Then $\mathcal{E}$ *is* L*-2CLR where*

$$\frac{L}{|\mathsf{sk}|} = \frac{(1/6 - o(1))\rho}{L_{\mathsf{ct}}\Big(\kappa, 2\kappa + \rho + L_{\mathsf{msg}} + L_{\mathsf{sig}}(\kappa, 2\kappa + \rho + L_{\mathsf{msg}})\Big)}$$

We choose a DIO-Compatible RCCA-secure Rerandomizable PKE with $L_{\mathsf{ct}}(\kappa, L_{\mathsf{msg}}) = c_1 \cdot L_{\mathsf{msg}}$, for some constant c_1, which is achieved by the Chase et al. [17] scheme, and a signature scheme with $L_{\mathsf{sig}}(\kappa, L_{\mathsf{msg}}) = o(L_{\mathsf{msg}})$, which can be constructed from collision-resistant hash functions, and setting $\rho = \rho(\kappa) = \omega(\kappa^2)$ yields an encryption scheme for messages of length $\Theta(\kappa)$ with constant leakage rate $c_1/4 - o(1)$.

In order to prove Theorem 5.14, we prove (in Lemma 16) that even under leakage, it is hard for any PPT adversary $\mathcal{A}$ to distinguish the output of the PRF y from uniform random. Given this, Theorem 5.14 follows immediately.

In fact, we will prove a slightly stronger lemma, by allowing $\mathcal{O}$ to leak on two consecutive keys in any given round. By combining this property with the results from Section 3, we prove that this construction achieves security when the adversary is allowed to leak on updates. More specifically, in the lemma below, in round i, we allow the adversary to specify a leakage function $f_i(\cdot, \cdot)$, and $\mathcal{O}$ returns $f_i(\mathsf{sk}_{i-1}, \mathsf{sk}_i)$.

Lemma 16 *For every* PPT *leaking adversary* $\mathcal{A}$*, who is given oracle access to a leakage oracle* $\mathcal{O}$ *and may leak at most* L *bits of the secret key, there exist random variables* $\widetilde{\mathsf{pk}}, \widetilde{\mathsf{sk}}_0, \ldots, \widetilde{\mathsf{sk}}_n$ *such that:*

$$\Big(\mathsf{pk}, t^*, y, \{f_i(\mathsf{sk}_{i-1}, \mathsf{sk}_i)\}_{i=1}^n \leftarrow \mathcal{A}^{\mathcal{O}(\cdot)}(\mathsf{pk})\Big) \overset{c}{\approx} \Big(\widetilde{\mathsf{pk}}, U_\rho, U_{L_{\mathsf{msg}}}, \{f_i(\widetilde{\mathsf{sk}}_{i-1}, \widetilde{\mathsf{sk}}_i)\}_{i=1}^n \leftarrow \mathcal{A}^{\mathcal{O}(\cdot)}(\widetilde{\mathsf{pk}})\Big)$$

where $y = C_{\mathsf{Dec}}(\mathsf{ct}_{\mathsf{dummy}}, t = G(r))$*,* n *is the number of key-update rounds requested by* $\mathcal{A}$*, and the distributions are taken over coins of* $\mathcal{A}$ *and choice of* $(\mathsf{pk}, \mathsf{sk}_0) \leftarrow \mathcal{E}.\mathsf{Gen}(1^\kappa)$*,* $\mathsf{sk}_i \leftarrow \mathcal{E}.\mathsf{Update}(\mathsf{sk}_{i-1})$*,* r *and choice of* $\widetilde{\mathsf{pk}}, \widetilde{\mathsf{sk}}_0, \ldots, \widetilde{\mathsf{sk}}_n, w$*, respectively.*

Proof: We prove the lemma via the following sequence of hybrids:

Hybrid 0: This hybrid is identical to the real game.

Let $\mathcal{D}^{\mathcal{A}}_{H_0}$ denote the distribution $(\mathsf{pk}, t, y, \{f_i(\mathsf{sk}_{i-1}, \mathsf{sk}_i)\}_{i=1}^n \leftarrow \mathcal{A}^{\mathcal{O}(\cdot)}(\mathsf{pk}))$ as in the left side of Lemma 16.

Hybrid 1: This hybrid is the same as Hybrid 0 except we replace pseudorandom $t = G(r)$ in the challenge ciphertext with uniform random $t^* \leftarrow \{0,1\}^\rho$.

Let $\mathcal{D}^{\mathcal{A}}_{H_1}$ denote the distribution $(\mathsf{pk}, t^*, y, \{f_i(\mathsf{sk}_{i-1}, \mathsf{sk}_i)\}_{i=1}^n \leftarrow \mathcal{A}^{\mathcal{O}(\cdot)}(\mathsf{pk}))$ where $y = C_{\mathsf{Dec}}(\mathsf{sk}_n, t^*)$ and the distribution is taken over coins of $\mathcal{A}$, choice of $(\mathsf{pk}, \mathsf{sk}_0) \leftarrow \mathcal{E}.\mathsf{Gen}(1^\kappa)$, $\mathsf{sk}_i \leftarrow \mathcal{E}.\mathsf{Update}(\mathsf{sk}_{i-1})$, t^* as described above.

Claim 5.15 *For every* PPT *adversary* $\mathcal{A}$*,*

$$\mathcal{D}^{\mathcal{A}}_{H_0} \overset{c}{\approx} \mathcal{D}^{\mathcal{A}}_{H_1}.$$

Proof: The proof follows from the security of the PRG used in C_{Enc}. We refer the reader to the proof of Claim 5.8. We note that the reduction holds even if the adversary were given all sk_i in full. ∎

Hybrid 2: This hybrid is the same as Hybrid 1 except we replace the key k used in C_{Enc} with a punctured key, $\widetilde{k} = \mathsf{PRF.Punct}(k, t^*)$. We denote the resulting public key by pk'. Let $\mathcal{D}^{\mathcal{A}}_{H_2}$ denote the distribution $\left(\mathsf{pk}', t^*, y, \{f_i(\mathsf{sk}_{i-1}, \mathsf{sk}_i)\}_{i=1}^n \leftarrow \mathcal{A}^{\mathcal{O}(\cdot)}(\mathsf{pk}')\right)$ where $y = C_{\mathsf{Dec}}(\mathsf{sk}_n, t^*)$ and the distribution is taken over coins of $\mathcal{A}$, and choice of $(\mathsf{pk}', \mathsf{sk}_0, \ldots, \mathsf{sk}_n)$, t^* as described above.

Claim 5.16 *For every* PPT *adversary* $\mathcal{A}$*,*

$$\mathcal{D}^{\mathcal{A}}_{H_1} \overset{c}{\approx} \mathcal{D}^{\mathcal{A}}_{H_2}.$$

Proof: The proof follows from the security of the obfuscation used in C_{Enc}. We refer the reader to the proof of Claim 5.9. We note again that the reduction holds even if $\mathcal{A}$ is given all sk_i in full. ∎

Hybrid $3^{(j)}$: We define a sequence of n hybrids, where n is the number of key-update rounds requested by the adversary. In hybrid $3^{(j)}$, in the first j update rounds, instead of refreshing $\mathsf{ct}_{\mathsf{dummy}}$ by computing $\mathsf{ct}_{\mathsf{dummy}} = \mathsf{RCCA.ReRand}(\mathrm{PK}_{\mathsf{RCCA}}, \mathsf{ct}_{\mathsf{dummy}})$, we replace the secret key with a fresh ciphertext: $\mathsf{ct}_{\mathsf{dummy}} = \mathsf{RCCA.Enc}(\mathrm{PK}_{\mathsf{RCCA}}, \sigma || 0^{2\kappa+\rho+L_{\mathsf{msg}}})$. For rounds $i > j$, the secret key is still refreshed through a re-randomization. We denote the resulting set of secret keys by $\{\mathsf{sk}_0^{(3|j)}, \ldots, \mathsf{sk}_n^{(3|j)}\}$. Let $\mathcal{D}^{\mathcal{A}}_{H_{3|j}}$ denote the distribution $\left(\mathsf{pk}', t^*, y, \{f_i(\mathsf{sk}_{i-1}^{(3|j)}, \mathsf{sk}_i^{(3|j)})\}_{i=1}^n \leftarrow \mathcal{A}^{\mathcal{O}(\cdot)}(\mathsf{pk}')\right)$ where $y = C_{\mathsf{Dec}}(\ \mathsf{sk}_n^{(3|j)},\ t^*)$ and the distribution is taken over coins of $\mathcal{A}$, and choice of $(\mathsf{pk}', \mathsf{sk}_0^{(3|j)}, \ldots, \mathsf{sk}_n^{(3|j)})$, t^* as described above.

Claim 5.17 *For every* PPT *adversary* $\mathcal{A}$*, and every* $j \in [n]$

$$\mathcal{D}^{\mathcal{A}}_{H_{3|j-1}} \overset{c}{\approx} \mathcal{D}^{\mathcal{A}}_{H_{3|j}}.$$

Proof: The proof is through a reduction to the property that the distribution of fresh ciphertexts and the distribution of re-randomized ciphertexts are indistinguishable, even given the secret key sk_j. The reduction adversary $\mathcal{S}$ receives $(\mathsf{pk}, \mathsf{sk}_j)$ from his challenger and uses them to generate the public keys for $\mathcal{E}$, along with the secret keys $\{\mathsf{sk}_0 = \mathsf{RCCA.Enc}(\mathrm{PK}_{\mathsf{RCCA}}, \sigma || 0^{2\kappa+\rho+L_{\mathsf{msg}}}), \ldots, \mathsf{sk}_{j-1} = \mathsf{RCCA.Enc}(\mathrm{PK}_{\mathsf{RCCA}}, \sigma || 0^{2\kappa+\rho+L_{\mathsf{msg}}})\}$. He submits ciphertext sk_{j-1} along with the underlying plaintext value as his challenge and receives c^* which is either a re-randomization sk_{j-1}, or a fresh encryption. He computes sk_{j+1} by re-randomizing his challenge ciphertext, and for $i \in \{j+1, \ldots, n\}$, he computes sk_i by re-randomizing sk_{i-1}. These ciphertexts are distributed either identically to those in hybrid $3^{(j-1)}$ or to those in hybrid $3^{(j)}$ and can be used by $\mathcal{S}$ to perfectly simulate the responses to $\mathcal{A}$'s leakage queries. It follows that the advantage of $\mathcal{S}$ is the same as the advantage of distinguishing $\mathcal{D}^{\mathcal{A}}_{H_{3|j-1}}$ from $\mathcal{D}^{\mathcal{A}}_{H_{3|j}}$. ∎

Hybrid 4: In this hybrid step, we again change the update phase. Instead of replacing $\mathsf{ct}_{\mathsf{dummy}}$ with a fresh encryption of $\sigma||0^{2\kappa+\rho+L_{\mathsf{msg}}}$, we will replace $\mathsf{ct}_{\mathsf{dummy}}$ with a fresh encryption of $\sigma'||((s_i, \alpha_i, H(t^*))||\ y)$. Here $s_i, \alpha_i, H, y, \sigma'$ are defined as follows: $s_i \leftarrow \mathbb{F}_q^{\rho/\kappa}$ where $q = 2^\kappa$; $H \leftarrow \mathcal{H}$ where $\mathcal{H}$ is a family of collision-resistant hash functions, and $H(t^*)$ is of size κ; $\alpha_i = \langle s_i, t^* \rangle$ is of size κ, where t^* is interpreted as an element in $\mathbb{F}_q^{\rho/\kappa}$, and α_i is interpreted as an element in $\mathbb{F}_q$; $y = \mathsf{PRF.Eval}(k, t^*)$ is of size L_{msg} and $\sigma' = \mathsf{SIG.Sign}(\mathsf{td}, ((s_i, \alpha_i, H(t^*))||\ y))$.

Let sk_i'' denote the secret key after update round i when computed as described above. Let $\mathcal{D}^{\mathcal{A}}_{H_4}$ denote the distribution $\left(\mathsf{pk}', t^*, y, \{f_i(\mathsf{sk}''_{i-1}, \mathsf{sk}''_i)\}_{i=1}^n \leftarrow \mathcal{A}^{\mathcal{O}(\cdot)}(\mathsf{pk}')\right)$ where $y = C_{\mathsf{Dec}}(\ \mathsf{sk}''_n, t^*)$ and the distribution is taken over coins of $\mathcal{A}$, and choice of $(\mathsf{pk}', \mathsf{sk}''_1, \ldots, \mathsf{sk}''_n)$, t^* as described above.

Claim 5.18 *For every* PPT *adversary* $\mathcal{A}$*,*

$$\mathcal{D}^{\mathcal{A}}_{H_{3|n}} \overset{c}{\approx} \mathcal{D}^{\mathcal{A}}_{H_4}.$$

Proof: The proof proceeds through an iteration of sub-hybrid steps, where in the jth iteration we change only the jth ciphertext. Let Hybrid $4^{(j)}$ denote the hybrid game where we have changed only the content of the first j ciphertexts. Let $\mathsf{sk}_i^{(4|j)}$ denote the secret key that is generated in the ith update round of hybrid $4^{(j)}$. Let $\mathcal{D}^{\mathcal{A}}_{H_{4|j}}$ denote the distribution $\left(\mathsf{pk}', t^*, y, \{f_i(\mathsf{sk}_{i-1}^{(4|j)}, \mathsf{sk}_i^{(4|j)})\}_{i=1}^n \leftarrow \mathcal{A}^{\mathcal{O}(\cdot)}(\mathsf{pk}')\right)$ where $y = C_{\mathsf{Dec}}(\ \mathsf{sk}_n^{(4|j)}, t^*)$ and the distribution is taken over coins of $\mathcal{A}$, and choice of $(\mathsf{pk}', \mathsf{sk}_0^{(4|j)}, \ldots, \mathsf{sk}_n^{(4|j)})$, t^* as described above. The proof follows then from the following claim. (Note that hybrid $4^{(0)}$ is equivalent to hybrid $3^{(n)}$.)

Claim 5.19 *For* $j \in \{0, \ldots, n-1\}$*,*

$$\mathcal{D}^{\mathcal{A}}_{H_{4|j}} \overset{c}{\approx} \mathcal{D}^{\mathcal{A}}_{H_{4|j+1}}$$

Proof: We now define the set of hybrids that allows us to prove Claim 5.19. We note that, by the definitions of hybrids $4^{(j)}$ and $4^{(j+1)}$ given above, for any $i \neq j$, $\mathsf{sk}_i^{(4|j)} = \mathsf{sk}_i^{(4|j+1)}$. Therefore, in each of the following sub-hybrids, we only need to make change to $\mathsf{sk}_j^{(4|j)}$.

Hybrid $4a^{(j)}$**:** In this hybrid we modify the update procedure in round j. Instead of replacing $\mathsf{ct}_{\mathsf{dummy}}$ with a fresh encryption $\mathsf{RCCA.Enc}(\mathrm{PK}_{\mathsf{RCCA}}, \sigma||0^{2\kappa+\rho+L_{\mathsf{msg}}})$, we set $\mathsf{sk}_j^{(4a|j)} = \mathsf{RCCA.SimEnc}(\tau_{\mathsf{sim}}, \sigma||0^{2\kappa+\rho+L_{\mathsf{msg}}})$. The other keys remain as they are in hybrid $4^{(j)}$.

Let $\mathsf{sk}_i^{(4a|j)}$ denote the secret key that is generated in the ith update round of hybrid $4a^{(j)}$. Let $\mathcal{D}^{\mathcal{A}}_{H_{4a|j}}$ denote the distribution $\left(\mathsf{pk}', t^*, y, \{f_i(\mathsf{sk}_{i-1}^{(4a|j)}, \mathsf{sk}_i^{(4a|j)})\}_{i=1}^n \leftarrow \mathcal{A}^{\mathcal{O}(\cdot)}(\mathsf{pk}')\right)$ where $y = C_{\mathsf{Dec}}(\ \mathsf{sk}_n^{(4a|j)}, t^*)$ and the distribution is taken over coins of $\mathcal{A}$, and choice of $(\mathsf{pk}', \mathsf{sk}_0^{(4a|j)}, \ldots, \mathsf{sk}_n^{(4a|j)})$, t^* as described above. We claim the following

Claim 5.20 *For any* PPT *adversary* $\mathcal{A}$*,*

$$\mathcal{D}^{\mathcal{A}}_{H_{4|j}} \overset{c}{\approx} \mathcal{D}^{\mathcal{A}}_{H_{4a|j}}$$

Proof: The proof follows by a reduction to the RCCA property ensuring the indistinguishability of simulated ciphertexts from real ciphertexts. Recall, For any efficient adversary $\mathcal{S}$

$$\mathcal{S}^{\mathsf{Enc}(\mathrm{PK}_{\mathsf{RCCA}}, \cdot)}(\mathrm{PK}_{\mathsf{RCCA}}, \mathrm{SK}_1) \overset{c}{\approx} \mathcal{S}^{\mathsf{SimEnc}(\tau_{\mathsf{sim}}, \cdot)}(\mathrm{PK}_{\mathsf{RCCA}}, \mathrm{SK}_1)\ .$$

To build the reduction, $\mathcal{S}$ receives keys $(\mathrm{PK}_{\mathsf{RCCA}}, \mathrm{SK}_1)$ for the RCCA scheme, which suffices to build the pk of $\mathcal{E}$. For update rounds $i < j$, he constructs sk_i by creating a fresh encryption of $\sigma'||((s_i, \alpha_i, H(t^*))||\ y)$ (as defined above), and for $i > j$, he constructs sk_i by creating a fresh encryption of $\sigma||0^{2\kappa+\rho+L_{\mathsf{msg}}}$. To create sk_j, he submits $\sigma||0^{2\kappa+\rho+L_{\mathsf{msg}}}$ to his challenger and uses the challenge ciphertext in the jth update round. The

distribution of secret keys generated by $\mathcal{S}$ is either identical to that in hybrid $4^{(j)}$, or to that in hybrid $4a^{(j)}$. It follows that $\mathcal{S}$'s advantage is the same as the distinguishing advantage between $\mathcal{D}^{\mathcal{A}}_{H_{4|j}}$ and $\mathcal{D}^{\mathcal{A}}_{H_{4a|j}}$.

■

Hybrid $4b^{(j)}$: In this hybrid we modify the circuit C_{keys} into $C_{\mathsf{keys}'}$ described in Figure 19. In words, the change involves using SK_2 instead of SK_1. If decryption under SK_2 outputs a message, we still verify the signature just as in C_{keys}, and if decryption outputs $\mathsf{SimFlag}$, we proceed as though the signature has been verified. Intuitively, these circuits have differing-inputs security because of the "simulation soundness" property of the RCCA encryption scheme. Denote the resulting public key by $\mathsf{pk}^{(4b|j)}$. Let $\mathcal{D}^{\mathcal{A}}_{H_{4b|j}}$ denote the distribution $\left(\mathsf{pk}^{(4b|j)}, t^*, y, \{f_i(\mathsf{sk}_{i-1}^{(4a|j)}, \mathsf{sk}_i^{(4a|j)})\}_{i=1}^n \leftarrow \mathcal{A}^{\mathcal{O}(\cdot)}(\mathsf{pk}^{(4b|j)})\right)$ where $y = C_{\mathsf{Dec}}(\mathsf{sk}_n^{(4a|j)}, t^*)$ and the distribution is taken over coins of $\mathcal{A}$, and choice of $(\mathsf{pk}^{(4b|j)}, \mathsf{sk}_0^{(4a|j)}, \ldots, \mathsf{sk}_n^{(4a|j)})$, t^* as described above. We claim the following

Internal (hardcoded) state: $\mathsf{keys}' = \{k, \mathsf{vk}, \mathrm{SK}_2\}$.

On input: $\mathsf{ct}_{\mathsf{dummy}}, t$

- If $\mathsf{RCCA.Dec}(\mathrm{SK}_2, \mathsf{ct}_{\mathsf{dummy}}) = \mathsf{SimFlag}$, output $z = \mathsf{PRF.Eval}(k, t)$.
- If $\mathsf{RCCA.Dec}(\mathrm{SK}_2, \mathsf{ct}_{\mathsf{dummy}}) = \perp$, output $\perp$.
- Else, parse $\mathsf{RCCA.Dec}(\mathrm{SK}_2, \mathsf{ct}_{\mathsf{dummy}})$ as (σ', m).
- If $\mathsf{SIG.Verify}(\sigma', m; \mathsf{vk}) = 0$ output $\perp$.
- Otherwise, output $z = \mathsf{PRF.Eval}(k, t)$.

Figure 19: Program $C_{\mathsf{keys}'}$. This program is obfuscated and placed in the public key, replacing C_{keys}. It is used during decryption.

Claim 5.21 *For any efficient adversary $\mathcal{A}$,*

$$\mathcal{D}^{\mathcal{A}}_{H_{4a|j}} \overset{c}{\approx} \mathcal{D}^{\mathcal{A}}_{H_{4b|j}}$$

Proof: We define the following sampler Samp and show that the circuit family $\mathcal{C}$ associated with Samp is a differing-inputs circuit family.
$\mathsf{Samp}(1^\kappa)$ does the following:

- Set $\mathsf{keys} = (\mathrm{SK}_1, k, \mathsf{vk})$ and set $\mathsf{keys}' = (\mathrm{SK}_2, k, \mathsf{vk})$.
- Let $C_0 = C_{\mathsf{keys}}$ and let $C_1 = C_{\mathsf{keys}'}$.
- Set $\mathsf{aux} = (\mathsf{vk}, \{\mathsf{sk}_i^{(4a|j)}\}_{i=0}^n, t^*, y)$
- Return (C_0, C_1, aux).

We now show that for every PPT adversary $\mathcal{A}$ there exists a negligible function negl such that

$$\Pr[C_0(x) \neq C_1(x) : (C_0, C_1, \mathsf{aux}) \leftarrow \mathsf{Samp}(1^\kappa), x \leftarrow \mathcal{A}(1^\kappa, C_0, C_1, \mathsf{aux})] \leq \mathsf{negl}(\kappa).$$

Assume towards contradiction that there exists a PPT adversary $\mathcal{A}$ and a polynomial $p(\cdot)$ such that for infinitely many κ, $\mathcal{A}$ outputs a distinguishing input with probability at least $1/p(\kappa)$. We construct a PPT adversary $\mathcal{S}$ that breaks the simulation soundness property of the RCCA scheme.
Upon receiving $(\mathrm{PK}_{\mathsf{RCCA}}, \mathrm{SK}_1, \mathrm{SK}_2) \leftarrow \mathsf{Gen}(1^\kappa)$ from the challenger, $\mathcal{S}$ does the following:

- Run $k \leftarrow \mathsf{PRF.Gen}(1^\kappa)$ and $(\mathsf{vk}, \mathsf{td}) \leftarrow \mathsf{SIG.Gen}(1^\kappa)$. Choose t^* at random, compute $y = \mathsf{PRF.Eval}(k, t^*)$, and set $\mathsf{keys} = (\mathrm{SK}_1, k, \mathsf{vk})$ and $\mathsf{keys}' = (\mathrm{SK}_2, k, \mathsf{vk})$.

- Sample $\sigma \leftarrow \mathsf{SIG.Sign}(\mathsf{td}, 0^{2\kappa+\rho+L_{\mathsf{msg}}})$ and submit $m^* = (\sigma||0^{2\kappa+\rho+L_{\mathsf{msg}}})$ as a challenge message.
- $\mathcal{S}$ receives challenge $c^* \leftarrow \mathsf{SimEnc}(\tau_{\mathsf{sim}}, m^*)$ and simulates Samp by doing the following:
  - Let $C_0 = C_{\mathsf{keys}}$ and let $C_1 = C_{\mathsf{keys}'}$.
  - $\mathcal{S}$ sets $\mathsf{sk}_j = c^*$. For $i \neq j$, $\mathcal{S}$ uses $\mathrm{PK}_{\mathsf{RCCA}}$ to generate sk_i honestly, as done in hybrid $4a^{(j)}$ above.
  - Set $\mathsf{aux} = (\mathsf{vk}, \{\mathsf{sk}_i\}_{i=0}^n, t^*, y)$
- $\mathcal{S}$ runs $\mathcal{A}(1^\kappa, C_0, C_1, \mathsf{aux})$ and receives x in return. He outputs x.

Note that if $\mathsf{Dec}(\mathrm{SK}_2, x) = \mathsf{SimFlag}$ and $\mathsf{Dec}(\mathrm{SK}_1, x) = m^*$, then both C_0 and C_1 output $y = \mathsf{PRF.Eval}(k, t^*)$. On the other hand, if $\mathsf{Dec}(\mathrm{SK}_2, x) = \mathsf{SimFlag}$ and $\mathsf{Dec}(\mathrm{SK}_1, x) \neq m^*$, then x violates the simulation soundness property of the RCCA scheme and $\mathcal{S}$ wins his game. Similarly, if $\mathsf{Dec}(\mathrm{SK}_2, x) \neq \mathsf{SimFlag}$ and $\mathsf{Dec}(\mathrm{SK}_2, x) = \mathsf{Dec}(\mathrm{SK}_1, x)$, then C_0 and C_1 have the same output (either y or $\perp$). If $\mathsf{Dec}(\mathrm{SK}_2, x) \neq \mathsf{SimFlag}$ and $\mathsf{Dec}(\mathrm{SK}_2, x) \neq \mathsf{Dec}(\mathrm{SK}_1, x)$ then, again, $\mathcal{S}$ wins his game.

Claim 5.21 follows from the fact that diO is a differing-inputs obfuscator and from the fact that the circuit family $\mathcal{C}$ associated with Samp is a differing inputs family. This is the case since $\mathcal{D}^{\mathcal{A}}_{H_{4a|j}}$ can be simulated given $(\mathsf{diO}(C_0), \mathsf{aux})$ and $\mathcal{D}^{\mathcal{A}}_{H_{4b|j}}$ can be simulated given $(\mathsf{diO}(C_1), \mathsf{aux})$. ■

Hybrid $4c^{(j)}$: In this hybrid we modify the update procedure in round j. Instead of replacing $\mathsf{ct}_{\mathsf{dummy}}$ with the simulated ciphertext $\mathsf{RCCA.SimEnc}(\tau_{\mathsf{sim}}, \sigma||0^{2\kappa+\rho+L_{\mathsf{msg}}})$, we compute

$$\mathsf{sk}_j^{(4c|j)} = \mathsf{RCCA.SimEnc}(\tau_{\mathsf{sim}}, \sigma'||((s_i, \alpha_i, H(t^*))|| \ y))$$

as described in Hybrid 4. The other keys remain as they are in hybrid $4b^{(j)}$. Let $\mathsf{sk}_i^{(4c|j)}$ denote the secret key that is generated in the ith update round of hybrid $4c^{(j)}$. Let $\mathcal{D}^{\mathcal{A}}_{H_{4c|j}}$ denote the distribution

$$\left(\mathsf{pk}^{(4b|j)}, t^*, y, \{f_i(\mathsf{sk}_{i-1}^{(4c|j)}, \mathsf{sk}_i^{(4c|j)})\}_{i=1}^n \leftarrow \mathcal{A}^{\mathcal{O}(\cdot)}(\mathsf{pk}^{(4b|j)})\right)$$

where $y = C_{\mathsf{Dec}}(\ \mathsf{sk}_n^{(4c|j)}, t^*)$ and the distribution is taken over coins of $\mathcal{A}$, and choice of $(\mathsf{pk}^{(4b|j)}, \mathsf{sk}_0^{(4c|j)}, \ldots, \mathsf{sk}_n^{(4c|j)})$, t^* as described above. We claim the following

Claim 5.22 *For any efficient adversary $\mathcal{A}$,*

$$\mathcal{D}^{\mathcal{A}}_{H_{4b|j}} \overset{c}{\approx} \mathcal{D}^{\mathcal{A}}_{H_{4c|j}}$$

Proof: The proof follows by a reduction to the RCCA property that ensures the indistinguishability of simulated ciphertexts, even when given SK_2. Recall, for any efficient adversary $\mathcal{S}$, and any message pair (m_0, m_1),

$$\mathcal{A}^{\mathsf{SimEnc}(\tau_{\mathsf{sim}}, \cdot)}(\mathrm{PK}_{\mathsf{RCCA}}, \mathrm{SK}_2, \mathsf{SimEnc}(\tau_{\mathsf{sim}}, m_0)) \overset{c}{\approx} \mathcal{A}^{\mathsf{SimEnc}(\tau_{\mathsf{sim}}, \cdot)}(\mathrm{PK}_{\mathsf{RCCA}}, \mathrm{SK}_2, \mathsf{SimEnc}(\tau_{\mathsf{sim}}, m_1)) \ .$$

To build the reduction, $\mathcal{S}$ receives keys $(\mathrm{PK}_{\mathsf{RCCA}}, \mathrm{SK}_2)$ for the RCCA scheme, which suffices to build the pk of $\mathcal{E}$. For update rounds $i < j$, he constructs sk_i by creating a fresh encryption $\mathsf{RCCA.Enc}(\mathrm{PK}_{\mathsf{RCCA}}, \sigma'||((s_i, \alpha_i, H(t^*))|| \ y))$ (as defined above), and for $i > j$, he constructs sk_i by creating a fresh encryption of $\sigma||0^{2\kappa+\rho+L_{\mathsf{msg}}}$. To create sk_j, he submits challenge plaintext pair: $(\sigma||0^{2\kappa+\rho+L_{\mathsf{msg}}}), (\sigma'||((s_i, \alpha_i, H(t^*))|| \ y))$ to his challenger and uses the challenge ciphertext in the jth update round. The distribution of secret keys generated by $\mathcal{S}$ is either identical to that in hybrid $4b^{(j)}$, or to that in hybrid $4c^{(j)}$. It follows that $\mathcal{S}$'s advantage is the same as the distinguishing advantage between $\mathcal{D}^{\mathcal{A}}_{H_{4b|j}}$ and $\mathcal{D}^{\mathcal{A}}_{H_{4c|j}}$.

■

Hybrid $4d^{(j)}$ In this hybrid, we modify $C_{\mathsf{keys}'}$ back to the circuit C_{keys} used in the real world. I.e. we return to using SK_1. Denote the resulting public key by $\mathsf{pk}^{(4d|j)}$. Let $\mathcal{D}^{\mathcal{A}}_{H_{4d|j}}$ denote the distribution

$$\left(\mathsf{pk}^{(4d|j)}, t^*, y, \{f_i(\mathsf{sk}_{i-1}^{(4c|j)}, \mathsf{sk}_i^{(4c|j)})\}_{i=1}^n \leftarrow \mathcal{A}^{\mathcal{O}(\cdot)}(\mathsf{pk}^{(4d|j)})\right)$$

where $y = C_{\mathsf{Dec}}(\ \mathsf{sk}_n^{(4c|j)}, t^*)$ and the distribution is taken over coins of $\mathcal{A}$, and choice of $(\mathsf{pk}^{(4d|j)}, \mathsf{sk}_0^{(4c|j)}, \ldots, \mathsf{sk}_n^{(4c|j)})$, t^* as described above. We claim the following

Claim 5.23 *For any efficient adversary $\mathcal{A}$,*

$$\mathcal{D}^{\mathcal{A}}_{H_{4c|j}} \overset{c}{\approx} \mathcal{D}^{\mathcal{A}}_{H_{4d|j}}$$

Proof: The proof follows from the security of diO. The proof is nearly identical to the proof of Claim 5.21, so we omit it. ∎

Hybrid $4e^{(j)}$**:** In this hybrid we modify the update procedure in round j. Instead of replacing $\mathsf{ct}_{\mathsf{dummy}}$ with a simulated ciphertext, we return to using a real ciphertext by computing

$$\mathsf{sk}_j^{(4e|j)} = \mathsf{RCCA.Enc}(\sigma' || ((s_i, \alpha_i, H(t^*)) || \ y))$$

The other keys remain as they are in hybrid $4d^{(j)}$. Let $\mathsf{sk}_i^{(4e|j)}$ denote the secret key that is generated in the ith update round of hybrid $4e^{(j)}$. Let $\mathcal{D}^{\mathcal{A}}_{H_{4e|j}}$ denote the distribution

$$\left(\mathsf{pk}^{(4d|j)}, t^*, y, \{f_i(\mathsf{sk}_{i-1}^{(4e|j)}, \mathsf{sk}_i^{(4e|j)})\}_{i=1}^n \leftarrow \mathcal{A}^{\mathcal{O}(\cdot)}(\mathsf{pk}^{(4d|j)})\right)$$

where $y = C_{\mathsf{Dec}}(\ \mathsf{sk}_n^{(4e|j)}, t^*)$ and the distribution is taken over coins of $\mathcal{A}$, and choice of $(\mathsf{pk}^{(4d|j)}, \mathsf{sk}_0^{(4e|j)}, \ldots, \mathsf{sk}_n^{(4e|j)})$, t^* as described above. We claim the following

Claim 5.24 *For any efficient adversary $\mathcal{A}$,*

$$\mathcal{D}^{\mathcal{A}}_{H_{4d|j}} \overset{c}{\approx} \mathcal{D}^{\mathcal{A}}_{H_{4e|j}}$$

Proof: The proof is again by the indistinguishability of a simulated ciphertext from a real ciphertext given sk_1. The proof proceeds identically to the proof of Claim 5.20, so we omit it. ∎

We note that hybrid $4e^{(j)}$ is identical to hybrid $4^{(j+1)}$, so this concludes the proofs of Claims 5.18 and 5.19. ∎

∎

Hybrid 5: In this hybrid game we replace: $C_{\mathsf{Dec}} = \mathsf{diO}(C_{\mathsf{keys}})$ with $C''_{\mathsf{Dec}} = \mathsf{diO}(C_{\mathsf{keys}''})$, where $C_{\mathsf{keys}''}$ is the circuit described in Figure 20. The difference between keys and keys$''$ is that we puncture k at the challenge point t^* in keys$''$. The difference between $C_{\mathsf{keys}''}$ and C_{keys} is that $C_{\mathsf{keys}''}$ will attempt to use the point obfuscation before turning to the PRF key. See Figure 20 for details. Denote the resulting public key by $\mathsf{pk}^{(5)}$, and let $\mathsf{sk}_i^{(5)}$ denote the secret key after the ith round of update, computed as described in the previous hybrid. Let $\mathcal{D}^{\mathcal{A}}_{H_5}$ denote the distribution $\left(\mathsf{pk}^{(5)}, t^*, y, \{f_i(\mathsf{sk}_{i-1}^{(5)}, \mathsf{sk}_i^{(5)})\}_{i=1}^n \leftarrow \mathcal{A}^{\mathcal{O}(\cdot)}(\mathsf{pk}^{(5)})\right)$ where $y = C''_{\mathsf{Dec}}(\ \mathsf{sk}_n^{(5)}, t^*)$ and the distribution is taken over coins of $\mathcal{A}$, and choice of $(\mathsf{pk}^{(5)}, \mathsf{sk}_0^{(5)}, \ldots, \mathsf{sk}_n^{(5)})$, t^* as described above. We claim the following

Claim 5.25 *For any efficient adversary $\mathcal{A}$,*

$$\mathcal{D}^{\mathcal{A}}_{H_4} \overset{c}{\approx} \mathcal{D}^{\mathcal{A}}_{H_5}$$

Internal (hardcoded) state: $\mathsf{keys}'' = \{k^* = \mathsf{PRF.Punct}(k, t^*), \mathsf{vk}, \mathrm{SK}_1, H\}$.

On input: $\mathsf{ct}_{\mathsf{dummy}}, t$

- Compute $(\sigma', (s, \alpha, H(t')|| \ y)) = \mathsf{RCCA.Dec}(\mathrm{SK}_1, \mathsf{ct}_{\mathsf{dummy}})$.
- If $\mathsf{SIG.Verify}(\sigma', (s, \alpha, H(t')|| \ y); \mathsf{vk}) = 0$ output $\perp$.
- If $\langle s, t\rangle = \alpha \bigwedge H(t) = H(t')$, output y.
- Else, output $\mathsf{PRF.Eval}(k^*, t)$.

Figure 20: Program $C_{\mathsf{keys}''}$. This program replaces C_{keys}. Recall it is obfuscated and placed in the public key. It is used during decryption.

Proof: The proof follows from the security of diO, and from the collision resistance of H and the security of the signature scheme. We define the following sampler Samp and show that the circuit family $\mathcal{C}$ associated with Samp is a differing-inputs circuit family.
$\mathsf{Samp}(1^\kappa)$ does the following:

- Set $\mathsf{keys} = (\mathrm{SK}_1, k, \mathsf{vk})$ and $\mathsf{keys}'' = (\mathrm{SK}_1, k^*, \mathsf{vk}, H)$.
- Let $C_0 = C_{\mathsf{keys}}$ and let $C_1 = C_{\mathsf{keys}''}$.
- Set $\mathsf{aux} = (\mathsf{vk}, \{\mathsf{sk}_i^{(5)}\}_{i=0}^n, t^*, y)$
- Return (C_0, C_1, aux).

We now show that for every PPT adversary $\mathcal{A}$ there exists a negligible function negl such that

$$\Pr[C_0(x) \neq C_1(x) : (C_0, C_1, \mathsf{aux}) \leftarrow \mathsf{Samp}(1^\kappa), x \leftarrow \mathcal{A}(1^\kappa, C_0, C_1, \mathsf{aux})] \leq \mathsf{negl}(\kappa).$$

Assume towards contradiction that there exists a PPT adversary $\mathcal{A}$ and a polynomial $p(\cdot)$ such that for infinitely many κ, $\mathcal{A}$ outputs a distinguishing input with probability at least $1/p(\kappa)$. Denote the event that $\mathcal{A}$ does so by Win. We show that the value output by $\mathcal{A}$ will either enable us to find a collision under H, or to break the existential unforgeability of the signature scheme.

Let $(\mathsf{ct}'_{\mathsf{dummy}}, t')$ denote the output of $\mathcal{A}$ when given circuits and auxiliary input sampled as described above. Let $(\sigma'||m') = \mathsf{RCCA.Dec}(\mathsf{sk}_1, \mathsf{ct}'_{\mathsf{dummy}})$. Letting Coll denote the probabilistic event that $(\sigma'||m') = \mathsf{Dec}(\mathrm{SK}_1, \mathsf{sk}_1^{(5)}) = \cdots = \mathsf{Dec}(\mathrm{SK}_1, \mathsf{sk}_n^{(5)})$ (where the randomness is over the coins of the Samp and the coins of $\mathcal{A}$), we divide our analysis into the following two cases.

Claim 5.26 *There exists an attacker $\mathcal{S}$ that finds collisions on $\mathcal{H}$ with probability* $\Pr[\ \mathsf{Win} \mid \mathsf{Coll}\ ]$.

Proof: Upon receiving $H \leftarrow \mathcal{H}$ from the challenger, $\mathcal{S}$ does the following:

- Run $(\mathrm{PK}_{\mathsf{RCCA}}, \mathrm{SK}_1) \leftarrow \mathsf{RCCA.Gen}(1^\kappa)$, $k \leftarrow \mathsf{PRF.Gen}(1^\kappa)$ and $(\mathsf{vk}, \mathsf{td}) \leftarrow \mathsf{SIG.Gen}(1^\kappa)$. Choose t^* at random, and compute $k^* = \mathsf{PRF.Punct}(k, t^*)$. Set $\mathsf{keys} = (\mathrm{SK}_1, k, \mathsf{vk})$ and $\mathsf{keys}'' = (\mathrm{SK}_1, k^*, \mathsf{vk}, H)$.
- $\mathcal{S}$ simulates Samp by doing the following:
  - $\mathcal{S}$ samples $s \leftarrow \mathbb{F}_q^{\rho/\kappa}$ and computes $\alpha = \langle s, t^*\rangle$. He computes $y = \mathsf{PRF.Eval}(k, t^*)$. He uses these values, along with challenge H and signing key td, to generate $\mathsf{sk}_i^{(5)}$ honestly.
  - Set $\mathsf{aux} = (\mathsf{vk}, \{\mathsf{sk}_i^{(5)}\}_{i=0}^n, t^*, y)$
- $\mathcal{S}$ runs $\mathcal{A}(1^\kappa, C_0, C_1, \mathsf{aux})$ and receives $x = (\mathsf{ct}''_{\mathsf{dummy}}, t'')$ in return. He outputs t'' as a collision with t^* under H.

Because we condition on event Coll, we have that
$\mathsf{RCCA.Dec}(\mathrm{SK}_1, \mathsf{ct}''_{\mathsf{dummy}}) = \sigma''||(s, \alpha, H(t^*)||\ y)$, where s and t^* are the values sampled by $\mathcal{S}$ when simulating $\mathsf{sk}_i^{(5)}$, $\alpha = \langle s, t^* \rangle$, and $\mathsf{SIG.Verify}(\sigma'', (s, \alpha, H(t^*)||\ y)) = 1$. It follows that the only way for $(\mathsf{ct}''_{\mathsf{dummy}}, t'')$ to constitute a differing input is if the following condition holds:

$$t'' \neq t^* \bigwedge \mathsf{PRF.Eval}(k, t'') \neq y \bigwedge \langle s, t'' \rangle = \alpha \bigwedge H(t'') = H(t^*)$$

To see why this condition is necessary, note that if $t'' = t^*$, or if $\mathsf{PRF.Eval}(k, t'') = y$, both circuits output y. If $\langle s, t'' \rangle \neq \alpha$, or $H(t'') \neq H(t^*)$, both circuits output $\mathsf{PRF.Eval}(k, t'')$. Now, since $t'' \neq t^*$, but $H(t'') = H(t^*)$, $\mathcal{S}$ has succeeded in finding a collision for function H. ■

Claim 5.27 *There exists an attacker $\mathcal{S}$ that finds forgeries with respect to* Sign *with probability* $\Pr[\ \mathsf{Win}\ |\ \overline{\mathsf{Coll}}\ ]$.

Proof: Upon receiving vk from the challenger, $\mathcal{S}$, who has the access to the signing oracle $\mathsf{SIG.Sign}(\mathsf{td}, \cdot)$, does the following:

- Run $(\mathrm{PK}_{\mathsf{RCCA}}, \mathrm{SK}_1) \leftarrow \mathsf{RCCA.Gen}(1^\kappa)$, $k \leftarrow \mathsf{PRF.Gen}(1^\kappa)$. Choose t^* at random, and compute $k^* = \mathsf{PRF.Punct}(k, t^*)$. Choose $H \leftarrow \mathcal{H}$. Set $\mathsf{keys} = (\mathrm{SK}_1, k, \mathsf{vk})$ and $\mathsf{keys}'' = (\mathrm{SK}_1, k^*, \mathsf{vk}, H)$.
- $\mathcal{S}$ simulates Samp by doing the following:
  - $\mathcal{S}$ samples $s \leftarrow \mathbb{F}_q^{\rho/\kappa}$ and computes $\alpha = \langle s, t^* \rangle$. He computes $y = \mathsf{PRF.Eval}(k, t^*)$. He uses these values to define message $m = (s, \alpha, H(t^*)||y)$ and uses his external signing oracle $\mathsf{SIG.Sign}(\mathsf{td}, \cdot)$ to get signature σ. He generates $\mathsf{sk}_i^{(5)}$ by repeatedly encrypting (σ, m).
  - Set $\mathsf{aux} = (\mathsf{vk}, \{\mathsf{sk}_i^{(5)}\}_{i=0}^n, t^*, y)$
- $\mathcal{S}$ runs $\mathcal{A}(1^\kappa, C_0, C_1, \mathsf{aux})$ and receives $x = (\mathsf{ct}''_{\mathsf{dummy}}, t'')$ in return. Then he computes $(\sigma'', m'') = \mathsf{RCCA.Dec}(\mathrm{SK}_1, \mathsf{ct}''_{\mathsf{dummy}})$, and outputs (m'', σ'') as a forged signature.

Because we are conditioning on the event $\overline{\mathsf{Coll}}$, it follows that $\mathcal{S}$ never obtains (σ'', m'') through his signing oracle. Furthermore, note that if $\mathsf{SIG.Verify}(\mathsf{vk}, m'', \sigma'') = 0$, then both C_0 and C_1 output $\perp$. It follows that (m'', σ'') is a successful forgery, and $\mathcal{S}$ wins his game. Note that we require a *strongly* unforgeable signature scheme, because, while $\overline{\mathsf{Coll}}$ states that $(m'', \sigma'') \neq (m, \sigma)$, it may be that $m'' = m$.

■

From Claims 5.26 and 5.27, and the security of $\mathcal{H}$ and Sign, it follows that $\Pr[\mathsf{Win}] < \mathsf{negl}$. Claim 5.23 follows from the fact that diO is a differing-inputs obfuscator and from the fact that the circuit family $\mathcal{C}$ associated with Samp is a differing inputs family. This is the case since $\mathcal{D}_{H_4}^{\mathcal{A}}$ can be simulated given $(\mathsf{diO}(C_0), \mathsf{aux})$ and $\mathcal{D}_{H_5}^{\mathcal{A}}$ can be simulated given $(\mathsf{diO}(C_1), \mathsf{aux})$. ■

Hybrid 6: In this game we modify the key update phase (in every round) to use $(s, \alpha, H(t^*)||\ y^*)$, where y^* is chosen uniformly at random, rather than as $\mathsf{PRF.Eval}(k, t^*)$. Let $\mathcal{D}_{H_6}^{\mathcal{A}}$ denote the distribution $\left(\mathsf{pk}^{(6)}, t^*, y, \{f_i(\mathsf{sk}_{i-1}^{(6)}, \mathsf{sk}_i^{(6)})\}_{i=1}^n \leftarrow \mathcal{A}^{\mathcal{O}(\cdot)}(\mathsf{pk}^{(6)})\right)$ where $y = C''_{\mathsf{Dec}}(\ \mathsf{sk}_n^{(6)},\ t^*)$ and the distribution is taken over coins of $\mathcal{A}$, and choice of $(\mathsf{pk}^{(6)}, \mathsf{sk}_0^{(6)}, \ldots, \mathsf{sk}_n^{(6)})$, t^* as described above. We claim the following

Claim 5.28 *For any* PPT *adversary $\mathcal{A}$,*

$$\mathcal{D}_{H_5}^{\mathcal{A}} \overset{c}{\approx} \mathcal{D}_{H_6}^{\mathcal{A}}$$

Proof: The proof is by reduction to the security of the punctured PRF. Specifically, $\mathcal{S}$ attacks the PRF by submitting t^* to his challenger and receiving $(\mathsf{PRF.Punct}(k, t^*), y^*)$ as a challenge. He then generates all the other necessary keys to simulate the view of $\mathcal{A}$. And uses $\mathcal{A}$'s guess to form his own. ■

Hybrid 7: In this game we replace C''_{Dec} with C'''_{Dec} by changing k^* in C''_{Dec} in the previous hybrid with the original k.

Claim 5.29 *For any* PPT *adversary* $\mathcal{A}$*,*

$$\mathcal{D}^{\mathcal{A}}_{H_6} \overset{c}{\approx} \mathcal{D}^{\mathcal{A}}_{H_7}$$

Proof: The proof is by a reduction to the security of the indistinguishability obfuscation. The main observation is that if $H(t') = H(t)$ and $\langle s, t\rangle = \alpha$, then both circuits output y^*. If this doesn't hold, then C''_{Dec} returns $y'' = \mathsf{PRF.Eval}(k^*, t)$, and C'''_{Dec} returns $y''' = \mathsf{PRF.Eval}(k, t)$; note that $y'' = y'''$ on points $t \neq t*$. Therefore, changing k^* into k will not effect the input/output behavior. If there is a difference in advantage, we can create an algorithm $\mathcal{B}$ that breaks the security of indistinguishability obfuscation. ∎

Hybrid $8^{(j)}$**:** In this sequence of hybrids we modify the key update procedure as follows. Instead of replacing $\mathsf{ct}_{\mathsf{dummy}}$ with an encryption of $(s, \alpha, H(t^*)||\ y^*)$, where $\langle s, t^*\rangle = \alpha$, in the first j key-update rounds we instead replace it with a fresh encryption of $(s_i, \alpha_i, H(t^*)||\ y^*)$ where $s_i \leftarrow \mathbb{F}_q^{\rho/\kappa}$ and $\alpha_i \leftarrow \mathbb{F}_q$. Note that the difference in this hybrid is that α_i is no longer necessarily equal to $\alpha = \langle s_i, t^*\rangle$.

Let $\mathcal{D}^{\mathcal{A}}_{H_{8|j}}$ denote the distribution $\left(\mathsf{pk}^{(8|j)}, t^*, y^*, \{f_i(\mathsf{sk}^{(8|j)}_{i-1}, \mathsf{sk}^{(8|j)}_i)\}^n_{i=1} \leftarrow \mathcal{A}^{\mathcal{O}(\cdot)}(\mathsf{pk}^{(8|j)})\right)$ where $y^* \leftarrow \{0,1\}^{L_{\mathsf{msg}}}$ is chosen uniformly at random, and the distribution is taken over coins of $\mathcal{A}$, and choice of $(\mathsf{pk}^{(8|j)}, \mathsf{sk}^{(8|j)}_0, \ldots, \mathsf{sk}^{(8|j)}_n)$, w, and y^* as described above. Noting that hybrid $8^{(0)}$ is the same as hybrid 7, We claim the following

Claim 5.30 *For any* PPT *adversary* $\mathcal{A}$*, and any* $j \in \{0, \ldots, n-1\}$

$$\mathcal{D}^{\mathcal{A}}_{H_{8|j}} \overset{c}{\approx} \mathcal{D}^{\mathcal{A}}_{H_{8|j+1}}$$

The heart of this proof relies on Theorem 5.31 which is stated and proven next. Intuitively, the claim in this theorem is that, for any leakage function f with bounded output length, it is hard to tell from $(t^*, f(s_j, H(t^*), \alpha_j, \alpha_{j+1}))$ whether $\alpha_j = \langle s_j, t^*\rangle$. The argument uses the fact that the inner product is a strong two-source extractor.

Lemma 17 (Strong Inner-Product Two-Source Extractor) *Let* $\vec{X}, \vec{Y}, Z$ *be correlated variables, where* $\vec{X}, \vec{Y}$ *have their support in* $\mathbb{F}_q^m$*, and are independent conditioned on* Z*. Let* U *be uniform and independent on* $\mathbb{F}_q$*. Then*

$$\Delta((Z, Y, \langle \vec{X}, \vec{Y}\rangle), (Z, Y, U)) \leq 2^{-s}$$

for some $s \geq 1 + \frac{1}{2}(k_X + k_Y - (m+1)\log q)$*, where* $k_X := \widetilde{\mathbf{H}}_\infty(\vec{X}|Z)$*,* $k_Y := \widetilde{\mathbf{H}}_\infty(\vec{Y}|Z)$

The worst-case version of this lemma is Theorem 1 of Lee et al. [41]. The average-case version that we use above follows as in Wichs' thesis [50, Lemma 4.1.4]. (Wichs does not state his lemma for the *strong* extraction property but this follows readily given the result of Lee et al. [41].)

Theorem 5.31 *Let* S_1, T *be random on* $\mathbb{F}_q^m$ *and* U *be random on* $\mathbb{F}_q$*. Fix any* $s_2 \in \mathbb{F}_q^m$ *and let* $A_2 = \langle s_2, T\rangle$*. Suppose* H *outputs* κ *bits and* f *outputs* L' *bits. Then*

$$\Delta\left((T, f(S_1, H(T), \langle S_1, T\rangle, A_2)), (T, f(S_1, H(T), U, A_2))\right) \leq 2^{-s'}, \tag{6}$$

where $s' = (m\log q - 3L' - 1 - \kappa - 2\log q)/3$*.*

Thus for the statistical distance in Equation 6 to be negligible we need

$$(m\log q - 3L' - 1 - \kappa - 2\log q)/3 \geq \log(1/\epsilon')\ ,$$

for some negligible ϵ'. Taking $\kappa = \log 1/\epsilon' = \log q$ and setting $m = \omega(\kappa)$ such that $m \cdot \kappa = \rho(\kappa)$, $L' = \rho/3 - O(\kappa)$, we can tolerate 2CLR leakage functions of length $L = L'/2$, which we will argue later.

Proof: Let BAD be the set of ℓ such that conditioning on $f(S_1, H(T), U, A_2) = \ell$ gives S_1 too little min-entropy. That is,

$$\mathsf{BAD} := \{\ell \text{ such that } \mathbf{H}_\infty(S_1 \mid f(S_1, H(T), U, A_2) = \ell) < m \log q - L' - s' - 1\} .$$

Additionally, for fixed t, $\alpha_2 = \langle s_2, t\rangle$, let $\mathcal{S}^t$ be the set of ℓ defined as,

$$\mathcal{S}^t := \{\ell \text{ such that } \Pr[f(S_1, H(t), U, \alpha_2) = \ell] > \Pr[f(S_1, H(t), \langle S_1, t\rangle, \alpha_2) = \ell]\}.$$

We claim that

$$\begin{aligned}
& \Delta\left((T, f(S_1, H(T), U, A_2)), (T, f(S_1, H(T), \langle S_1, T\rangle, A_2))\right) \\
= \;& \mathbf{E}_t\, \Delta\left(f(S_1, H(t), U, \alpha_2), f(S_1, H(t), \langle S_1, t\rangle, \alpha_2)\right) \\
\leq \;& \mathbf{E}_t \left(\sum_{\ell \in \overline{\mathsf{BAD}} \cap \mathcal{S}^t} \Pr\left[f(S_1, H(t), U, \alpha_2) = \ell\right] - \Pr\left[f(S_1, H(t), \langle S_1, t\rangle, A_2) = \ell\right] \right) \\
& \quad + \mathbf{E}_t \left(\sum_{\ell \in \mathsf{BAD} \cap \mathcal{S}^t} \Pr\left[f(S_1, H(t), U, \alpha_2) = \ell\right] \right) \\
\leq \;& \sum_{\ell \notin \mathsf{BAD}} \left(\mathbf{E}_t \left|\Pr\left[f(S_1, H(t), U, \alpha_2) = \ell\right] - \Pr\left[f(S_1, H(t), \langle S_1, t\rangle, \alpha_2) = \ell\right]\right|\right) \\
& \quad + \sum_{\ell \in \mathsf{BAD}} \left(\mathbf{E}_t \Pr\left[f(S_1, H(t), U, \alpha_2) = \ell\right]\right) \\
= \;& \sum_{\ell \notin \mathsf{BAD}} \left(\mathbf{E}_t \left|\Pr\left[f(S_1, H(t), U, \alpha_2) = \ell\right] - \Pr\left[f(S_1, H(t), \langle S_1, t\rangle, \alpha_2) = \ell\right]\right|\right) \\
& \quad + \Pr[f(S_1, H(T), U, A_2) \in \mathsf{BAD}] \\
\leq \;& \sum_{\ell \notin \mathsf{BAD}} \left(\mathbf{E}_t \left|\Pr\left[f(S_1, H(t), U, \alpha_2) = \ell\right] - \Pr\left[f(S_1, H(t), \langle S_1, t\rangle, \alpha_2) = \ell\right]\right|\right) + 2^{-s'-1} \qquad (7) \\
\leq \;& 2^{L'} \cdot 2^{-s'-1-L'} + 2^{-s'-1}, \qquad (8) \\
= \;& 2^{-s'}.
\end{aligned}$$

where (7) is due to Markov's inequality and the definition of the set BAD. (8) is due to the following claim:

Claim 5.32 *For any $\ell \notin \mathsf{BAD}$ in the range of f,*

$$\mathbf{E}_t \left|\Pr\left[f(S_1, H(t), U, \alpha_2) = \ell\right] - \Pr\left[f(S_1, H(t), \langle S_1, t\rangle, \alpha_2) = \ell\right]\right| \leq 2^{-s'-1-L'}.$$

Proof: First, for fixed t, β and $\widehat{h} = H(t)$, we have:

$$\begin{aligned}
\Pr[f(S_1, \widehat{h}, \langle S_1, t\rangle, \alpha_2) = \ell \wedge \langle S_1, t\rangle = \beta] &= \Pr[f(S_1, \widehat{h}, \beta, \alpha_2) = \ell \wedge \langle S_1, t\rangle = \beta] \\
&= \Pr[f(S_1, \widehat{h}, \beta, \alpha_2) = \ell] \cdot \Pr[\langle S_1, t\rangle = \beta \mid f(S_1, \widehat{h}, \beta, \alpha_2) = \ell]. \qquad (9)
\end{aligned}$$

and

$$\begin{aligned}
\Pr[f(S_1, \widehat{h}, U, \alpha_2) = \ell \wedge U = \beta] &= \Pr[f(S_1, \widehat{h}, \beta, \alpha_2) = \ell \wedge U = \beta] \\
&= \Pr[f(S_1, \widehat{h}, \beta, \alpha_2) = \ell] \cdot \Pr[U = \beta]. \qquad (10)
\end{aligned}$$

Now, we have that:

$$
\begin{aligned}
&\mathbf{E}_{t\leftarrow T}\left|\Pr\left[f(S_1,H(t),U,\alpha_2)=\ell\right]-\Pr\left[f(S_1,H(t),\langle S_1,t\rangle,\alpha_2)=\ell\right]\right| \\
&=\sum_{\widehat{h},\alpha_2}\Pr[H(T)=\widehat{h}\wedge\langle s_2,T\rangle=\alpha_2]\cdot\mathop{\mathbf{E}}_{t\leftarrow T'}\left|\Pr[f(S_1,\widehat{h},U,\alpha_2)=\ell]-\Pr[f(S_1,\widehat{h},\langle S_1,t\rangle,\alpha_2)=\ell]\right| \\
&\le\sum_{\widehat{h},\alpha_2}\Pr[H(T)=\widehat{h}\wedge\langle s_2,T\rangle=\alpha_2] \\
&\qquad\cdot\mathop{\mathbf{E}}_{t\leftarrow T'}\sum_{\beta}\left|\Pr[f(S_1,\widehat{h},U,\alpha_2)=\ell\wedge U=\beta]-\Pr[f(S_1,\widehat{h},\langle S_1,t\rangle,\alpha_2)=\ell\wedge\langle S_1,t\rangle=\beta]\right| \qquad (11)\\
&=\sum_{\widehat{h},\alpha_2}\Pr[H(T)=\widehat{h}\wedge\langle s_2,T\rangle=\alpha_2] \\
&\qquad\cdot\sum_{\beta}\left(\mathop{\mathbf{E}}_{t\leftarrow T'}\left|\Pr[f(S_1,\widehat{h},U,\alpha_2)=\ell\wedge U=\beta]-\Pr[f(S_1,\widehat{h},\langle S_1,t\rangle,\alpha_2)=\ell\wedge\langle S_1,t\rangle=\beta]\right|\right) \\
&=\sum_{\widehat{h},\alpha_2}\Pr[H(T)=\widehat{h}\wedge\langle s_2,T\rangle=\alpha_2] \\
&\qquad\cdot\sum_{\beta}\left(\mathop{\mathbf{E}}_{t\leftarrow T'}\left[\Pr[f(S_1,\widehat{h},\beta,\alpha_2)=\ell]\cdot\left|\Pr[U=\beta]-\Pr[\langle S_1,t\rangle=\beta\mid f(S_1,\widehat{h},\beta,\alpha_2)=\ell]\right|\right]\right) \qquad (12)\\
&\le\sum_{\widehat{h},\alpha_2}\Pr[H(T)=\widehat{h}\wedge\langle s_2,T\rangle=\alpha_2]\cdot\sum_{\beta}\left(\mathop{\mathbf{E}}_{t\leftarrow T'}\left|\Pr[U=\beta]-\Pr[\langle S_1,t\rangle=\beta\mid f(S_1,\widehat{h},\beta,\alpha_2)=\ell]\right|\right) \\
&\qquad\qquad (13)\\
&=\sum_{\widehat{h},\alpha_2}\Pr[H(T)=\widehat{h}\wedge\langle s_2,T\rangle=\alpha_2]\;\Delta\left(\left(T',\left(\langle S_1,T'\rangle\mid f(S_1,\widehat{h},\beta,\alpha_2)=\ell\right)\right),\left(T',U\right)\right) \\
&\le 2^{-s'-1-L'}, \qquad (14)
\end{aligned}
$$

where T' is uniform on the set $\{t\mid H(t)=\widehat{h}\wedge\langle s_2,t\rangle=\alpha_2\}$, (11) follows by triangle inequality, (12) follows from (9) and (10), (13) follows since the quantity $\Pr[f(S_1,\widehat{h},\beta,\alpha_2)=\ell]$ is always less than or equal to 1.

To see why (14) holds, note that $X:=\left(S_1\mid f(S_1,\widehat{h},\beta,\alpha_2)=\ell\right)$ and $Y:=\left(T\mid H(T)=\widehat{h}\wedge\langle s_2,T\rangle=\alpha_2\right)$ are independent sources. Furthermore,

$$k_X:=\widetilde{\mathbf{H}}_\infty\left(S_1\mid f(S_1,\widehat{h},\beta,\alpha_2)=\ell\right)\ge m\log q-L'-s'-1$$

by the assumption $\ell\notin\mathsf{BAD}$. And

$$k_Y:=\widetilde{\mathbf{H}}_\infty\left(T\mid H(T)=\widehat{h}\wedge\langle s_2,T\rangle=\alpha_2\right)\ge m\log q-\kappa-\log q$$

by the "chain rule" for average min-entropy [25, Lemma 2.2]. Thus the last inequality follows by Lemma 17. ■

■

We now use theorem 5.31 to prove Claim 5.30.

Proof: We show that if $\mathcal{A}$ can distinguish hybrid $8^{(j)}$ from $8^{(j+1)}$ with some advantage ϵ', then there is a distinguisher $\mathcal{S}$ and functions f^* and H with output lengths at most $L'=2L$ bits and κ bits respectively, such that $\mathcal{S}$ distinguishes the two distributions $(T,f^*(S_1,H(T),\langle S_1,T\rangle,A_2))$ and $(T,f^*(S_1,H(T),U,A_2))$ from one another with the same advantage. Since this violates the assertion in Theorem 5.31, the claim follows directly.

$\mathcal{S}$ simulates the view of $\mathcal{A}$ as follows. He samples $(\mathsf{pk}_{\mathsf{RCCA}}, \mathsf{sk}_1, \mathsf{sk}_2) \leftarrow \mathsf{RCCA.Gen}(1^\kappa)$, $(\mathsf{vk}, \mathsf{td}) \leftarrow \mathsf{SIG.Gen}(1^\kappa)$, $k \leftarrow \mathsf{PRF.Gen}(1^\kappa)$, and constructs C_{Dec} and C_{Enc} as described in the previous hybrid. Then, to simulate the replies to $\mathcal{A}$'s leakage queries, $\mathcal{S}$ acts as follows. For $i \in \{0, \ldots, j-1\}$, he samples $s_i \leftarrow \mathbb{F}_q^{\rho/\kappa}$, and $\alpha_i \leftarrow \mathbb{F}_q$. For $i \in \{0, \ldots, j+1\}$ he samples randomness $r_i^{(1)}$ to be used in signing the necessary plaintext values, and $r_i^{(2)}$ to be used in encrypting the necessary values. Finally, he samples r_{tape} to be used as $\mathcal{A}$'s random tape, and $y^* \leftarrow \{0,1\}^{L_{\mathsf{msg}}}$. We denote the union of these sets of random values by rand. He then submits function $f^*_{\mathsf{rand},\mathsf{td},\mathsf{pk}_{\mathsf{RCCA}}}$ to his challenger, where $f^*_{\mathsf{rand},\mathsf{td},\mathsf{pk}_{\mathsf{RCCA}}}$ is defined as in Figure 21.

Internal (hardcoded) state: $\mathsf{rand} = \left\{\{s_i, \alpha_i, r_i^{(1)}, r_i^{(2)}\}_{i=0}^{j-1}, r_j^{(1)}, r_j^{(2)}, r_{j+1}^{(1)}, r_{j+1}^{(2)}, y^*, r_{\mathsf{tape}}\right\}$, td, $\mathsf{pk}_{\mathsf{RCCA}}$.

On input: $s_j, H(t^*), \alpha_j, \alpha_{j+1}$

- For $i \in \{0, \ldots, j+1\}$,
  - let $m_i = (s_i, H(t^*), \alpha_i || y^*)$,
  - let $\sigma_i = \mathsf{SIG.Sign}(\mathsf{td}, m_i\ ; r_i^{(1)})$,
  - let $\mathsf{sk}_i = \mathsf{RCCA.Enc}(\mathsf{pk}_{\mathsf{RCCA}}, \sigma_i \mid\mid m_i\ ; r_i^{(2)})$.
- Run the code of $\mathcal{A}$ using random tape r_{rand} until he has made $j+1$ leakage queries. For $i \in \{1, \ldots, j+1\}$, reply to leakage query f_i by computing $f_i(\mathsf{sk}_{i-1}, \mathsf{sk}_i)$.
- Output $f_j(\mathsf{sk}_{j-1}, \mathsf{sk}_j), f_{j+1}(\mathsf{sk}_j, \mathsf{sk}_{j+1})$

Figure 21: The function $f^*_{\mathsf{rand},\mathsf{td},\mathsf{pk}_{\mathsf{RCCA}}}$ is the leakage function sent by reduction adversary $\mathcal{S}$ to his challenger. In response, he receives either a sample from $(T, f^*_{\mathsf{rand},\mathsf{td},\mathsf{pk}_{\mathsf{RCCA}}}(S_1, H(T), \langle S_1, T\rangle, \alpha_2))$, or from $(T, f^*_{\mathsf{rand},\mathsf{td},\mathsf{pk}_{\mathsf{RCCA}}}(S_1, H(T), \beta, \alpha_2))$.

$\mathcal{S}$ receives challenge value $(t^*, f_{j-1}(\mathsf{sk}_{j-2}, \mathsf{sk}_{j-1}), f_j(\mathsf{sk}_{j-1}, \mathsf{sk}_j), \alpha_{j+1})$. Once he knows t^*, we note that $\mathcal{S}$ has all the information needed to simulate $\mathcal{A}$'s view for the first $j-2$ leakage queries. He does so precisely as was done by his challenger when running $f^*_{\mathsf{rand},\mathsf{td},\mathsf{pk}_{\mathsf{RCCA}}}$, using identical random values, and eliciting identical leakage queries. To simulate the replies to leakage queries f_{j-1} and f_j he uses the two outputs of $f^*_{\mathsf{rand},\mathsf{td},\mathsf{pk}_{\mathsf{RCCA}}}$.

By our setting of parameters, output size of f is L, and output size of f^* is $L' = 2L$. To simulate the replies to query f_{j+1}, $\mathcal{S}$ computes $\alpha_{j+1} = \langle s_{j+1}, t^*\rangle$, where, recall, s_{j+1} was fixed prior to his challenge query. He simulates $\mathsf{sk}_{j+1}^{(8|j)}$ by signing and encrypting $(s_{j+1}, H(t^*), \alpha_{j+1} || y^*)$ with random coins $r_{j+1}^{(1)}, r_{j+1}^{(2)}$. For $j+1 < i < n$, he constructs $\mathsf{sk}_i^{(8|j)}$ by sampling $s_i \leftarrow \mathbb{F}_q^{\rho/\kappa}$, computing $\alpha_i = \langle s_i, t^*\rangle$, and then signing and encrypting $(s_i, H(t^*), \alpha_i || y^*)$ using uniformly chosen coins. He simulates leakage queries $f_{j+1}, \ldots, f_n$ using the $\mathsf{sk}_{j+1}^{(8|j)}, \ldots, \mathsf{sk}_n^{(8|j)}$. The above simulation is distributed exactly as Hybrid $8^{(j)}$ when $\mathcal{S}$'s challenge comes from
$(T, f^*_{\mathsf{rand},\mathsf{td},\mathsf{pk}_{\mathsf{RCCA}}}(S_j, H(T), \langle S_j, T\rangle, \alpha_{j+1}))$, and it is distributed exactly as Hybrid $8^{(j+1)}$ when $\mathcal{S}$'s challenge comes from $(T, f^*_{\mathsf{rand},\mathsf{td},\mathsf{pk}_{\mathsf{RCCA}}}(S_j, H(T), \beta, \alpha_{j+1}))$, which concludes the proof.

■

Hybrid 9: In this hybrid, we change the circuit C_{Dec} such that after decrypting $\mathsf{ct}_{\mathsf{dummy}}$, it verifies the signature, but otherwise ignores the content. Note that by the end of the Hybrid 8 sequence, the inner product relationship has been "broken", so with all but negligible probability over the choices of s_i, α_i, we are already ignoring the plaintext values anyway. Let $C_{\mathsf{keys}}^{(9)}$ denote the resulting circuit, and let $\mathsf{pk}^{(9)}$ denote the modified public key that results from obfuscating the updated circuit. Let $\mathcal{D}_{H_9}^{\mathcal{A}}$ denote the distribution $\left(\mathsf{pk}^{(9)}, t^*, y^*, \{f_i(\mathsf{sk}_{i-1}^{(9)}, \mathsf{sk}_i^{(9)})\}_{i=1}^n \leftarrow \mathcal{A}^{\mathcal{O}(\cdot)}(\mathsf{pk}^{(9)})\right)$ where y^* is chosen uniformly at random, and the distribution is taken over coins of $\mathcal{A}$, and choice of $(\mathsf{pk}^{(8)}, \mathsf{sk}_0^{(8)}, \ldots, \mathsf{sk}_n^{(8)})$, y^* as described above. We claim the following

Claim 5.33 *For any* PPT *adversary* $\mathcal{A}$*,*

$$\mathcal{D}^{\mathcal{A}}_{H_{8|n}} \overset{c}{\approx} \mathcal{D}^{\mathcal{A}}_{H_9}$$

Proof: The proof follows from a reduction to the security of the diO scheme. The argument that these two circuits have differing-input security follows almost identically as in the proof of Claim 5.25, with a reduction to the either the security of the signature scheme, or the collision resistance of H. We omit repeating the proof. ■

Hybrid 10: In this hybrid, we replace the content of $\mathsf{ct}_{\mathsf{dummy}}$ with a fresh encryption of $\sigma||0^{2\kappa+\rho+L_{\mathsf{msg}}}$, where σ is a signature on $0^{2\kappa+\rho+L_{\mathsf{msg}}}$. Let $\mathsf{sk}_i^{(10)}$ denote the resulting secret key in update round i. Let $\mathsf{pk}^{(10)}$ denote the public key (which is generated in the same fashion as in hybrid 9.). Let $\mathcal{D}^{\mathcal{A}}_{H_{10}}$ denote the distribution $\left(\mathsf{pk}^{(10)}, t^*, y^*, \{f_i(\mathsf{sk}_{i-1}^{(10)}, \mathsf{sk}_i^{(10)})\}_{i=1}^n \leftarrow \mathcal{A}^{\mathcal{O}(\cdot)}(\mathsf{pk}^{(10)})\right)$ where y^* is chosen uniformly at random, and the distribution is taken over coins of $\mathcal{A}$, and choice of $(\mathsf{pk}^{(10)}, \mathsf{sk}_0^{(10)}, \ldots, \mathsf{sk}_n^{(10)})$, y^* as described above. We claim the following

Claim 5.34 *For any* PPT *adversary* $\mathcal{A}$*,*

$$\mathcal{D}^{\mathcal{A}}_{H_9} \overset{c}{\approx} \mathcal{D}^{\mathcal{A}}_{H_{10}}$$

Proof: The proof follows from the security of the RCCA scheme. We transition through a sequence of hybrids (and sub-hybrids), changing one plaintext value at a time, just as we did in Claim 5.19. Note that, just as in that Claim, our circuit only verifies the signature on the plaintext, and makes no use of the value otherwise. Since we only need to verify the signature, we can replace decryption with SK_1 by a check for $\mathsf{SimFlag}$ after decrypting with SK_2, and use diO security, as we did previously. We omit the details of the proof. ■

Finally, we have the following claim, where the right hand side is the same as in Lemma 16.

Claim 5.35

$$\mathcal{D}^{\mathcal{A}}_{H_{10}} \overset{s}{\approx} \left(\widetilde{\mathsf{pk}}, U_\rho, U_{L_{\mathsf{msg}}}, \{f_i(\widetilde{\mathsf{sk}}_{i-1}, \widetilde{\mathsf{sk}}_i)\}_{i=1}^n \leftarrow \mathcal{A}^{\mathcal{O}(\cdot)}(\widetilde{\mathsf{pk}})\right)$$

Proof: Note that $\mathsf{pk}^{(10)}$ and $\mathsf{sk}_0^{(10)}, \ldots, \mathsf{sk}_n^{(10)}$ contain no information about y^*. ■

This concludes the proof of Lemma 16. ■

6 Continual Leakage Resilience for One Way Relation

Dodis et al. [21] defined one-way relation (OWR) in the regular continual leakage resilience setting, and present a construction based on a simpler primitive – leakage-indistinguishable re-randomizable relation (LIRR). In this section, we first extend their definition to *2CLR* and *CLR with leakage on key updates*. Then we prove their LIRR-based construction actually achieves the 2CLR security. By using our generic transformation, we can have a construction for achieving CLR with leakage on key updates. Additionally, we give a new construction of 2CLR OWR based on 2CLR PKE, which can be obtained from the previous sections.

6.1 Continual Leakage Model

A one-way relation scheme OWR consists of two algorithms: $(\mathsf{OWR.Gen}, \mathsf{OWR.Verify})$. In the continual leakage setting, we require an additional algorithm $\mathsf{OWR.Update}$ which updates the secret keys. Note that the public key remains unchanged.

- $\mathsf{OWR.Gen}(1^\kappa) \to (\mathsf{pk}, \mathsf{sk}_0)$. The key generation algorithm takes in the security parameter κ, and outputs a secret key sk_0 and a public key pk.

- $\mathsf{OWR.Verify}(\mathsf{pk}, \mathsf{sk}) \to \{0, 1\}$. The verification algorithm takes in the public key pk, a secret key sk, and outputs either 0 or 1.
- $\mathsf{OWR.Update}(\mathsf{sk}_{i-1}) \to \mathsf{sk}_i$. The update algorithm takes in a secret key sk_{i-1} and produces a new secret key sk_i for the *same* public key.

Correctness. The OWR scheme satisfies correctness if for any polynomial $q = q(\kappa)$, it holds that for all $i \in \{0, 1, \ldots, q\}$, $\mathsf{OWR.Verify}(\mathsf{pk}, \mathsf{sk}_i) = 1$, where $(\mathsf{pk}, \mathsf{sk}_0) \leftarrow \mathsf{OWR.Gen}(1^\kappa)$, and $\mathsf{sk}_{i+1} \leftarrow \mathsf{OWR.Update}(\mathsf{sk}_i)$.

Security. We define continual leakage security for one-way relations in terms of the following game between a challenger and an attacker. We let κ denote the security parameter, and the parameter μ controls the amount of leakage allowed.

Setup Phase. The game begins with a setup phase. The challenger calls $\mathsf{OWR.Gen}(1^\kappa)$ to create the initial secret key sk_0 and public key pk. It gives pk to the attacker. No leakage is allowed in this phase.

Query Phase. In this phase, the attacker launches a polynomial number of leakage queries. Each time, say in the i-th query, the attacker provides an efficiently computable leakage function f_i whose output is at most μ bits, and the challenger chooses randomness r_i, updates the secret key from sk_{i-1} to sk_i, and gives the attacker the leakage response ℓ_i. In the CLR model, the leakage attack is applied on a single secret key, and the leakage response $\ell_i = f_i(\mathsf{sk}_{i-1})$. In the 2CLR model, the leakage attack is applied on consecutive two secret keys, i.e., $\ell_i = f_i(\mathsf{sk}_{i-1}, \mathsf{sk}_i)$. In the model of CLR with leakage on key updates, the leakage attack is applied on the current secret key and the randomness used for updating the secret key, i.e., $\ell_i = f_i(\mathsf{sk}_{i-1}, r_i)$.

Recovery Phase. The attacker outputs some sk^*. The attacker wins the game if $\mathsf{OWR.Verify}(\mathsf{pk}, \mathsf{sk}^*) = 1$. We define the success probability of the attacker in this game as $\Pr[\mathsf{OWR.Verify}(\mathsf{pk}, \mathsf{sk}^*) = 1]$.

Definition 6.1 (Continual Leakage Resilience) *We say a One Way Relation scheme is μ-CLR secure (respectively, μ-2CLR secure, or μ-CLR secure with leakage on key updates) if any* PPT *attacker only has a negligible advantage (negligible in κ) in the above game.*

6.2 Construction based on Leakage-Indistinguishable Re-randomizable Relation

6.2.1 Leakage-Indistinguishable Re-randomizable Relation

In [21], Dodis et al introduce a new primitive, *leakage-indistinguishable re-randomizable relation (LIRR)*, and show that this primitive can be used to construct OWR in the CLR model where the adversary is allowed to leak on the secret key in each round of leakage attack. LIRR allows one to sample two types of secret-keys: "good" keys and "bad" keys. Both types of keys look valid and are acceptable by the verification procedure, but they are produced in very different ways. In fact, given the ability to produce good keys, it is hard to produce any bad key and vice-versa. On the other hand, even though the two types of keys are very different, they are hard to distinguish from each other. More precisely, given the ability to produce both types of keys, and μ bits of leakage on a "challenge" key of an unknown type (good or bad), it is hard to come up with a new key of the same type. More formally, a LIRR consists of PPT algorithms $(\mathsf{Setup}, \mathsf{SampG}, \mathsf{SampB}, \mathsf{Update}, \mathsf{Verify}, \mathsf{isGood})$ with the following syntax:

- $(\mathsf{pk}, s_G, s_B, \mathsf{dk}) \leftarrow \mathsf{Setup}(1^\kappa)$: This algorithm returns a public-key pk, a "good" sampling-key s_G, a "bad" sampling key s_B, and a distinguishing-trapdoor dk.
- $\mathsf{sk}_G \leftarrow \mathsf{SampG}_{\mathsf{pk}}(s_G)$ and $\mathsf{sk}_B \leftarrow \mathsf{SampB}_{\mathsf{pk}}(s_B)$: These algorithms sample good/bad secret-keys using good/bad sampling keys respectively. We omit the subscript pk when clear from context.

- $b \leftarrow \mathsf{isGood}(\mathsf{pk}, \mathsf{sk}, \mathsf{dk})$: This algorithm uses dk to distinguish good secret-keys sk from bad ones.
- $\mathsf{sk}_i \leftarrow \mathsf{Update}(\mathsf{sk}_{i-1})$ and $b \leftarrow \mathsf{Verify}(\mathsf{pk}, \mathsf{sk})$: These two algorithms have the same syntax as in the definition of OWR in the CLR model.

Definition 6.2 *We say* $(\mathsf{Setup}, \mathsf{SampG}, \mathsf{SampB}, \mathsf{Update}, \mathsf{Verify}, \mathsf{isGood})$ *is a* μ *leakage-indistinguishable re-randomizable relation (LIRR) if it satisfies the following properties:*

Correctness: *If* $(\mathsf{pk}, s_G, s_B, \mathsf{dk}) \leftarrow \mathsf{Setup}(1^\kappa)$, $\mathsf{sk}_G \leftarrow \mathsf{SampG}(s_G)$, $\mathsf{sk}_B \leftarrow \mathsf{SampB}(s_B)$ *then*

$$\Pr\left[\begin{array}{c}\mathsf{Verify}(\mathsf{pk}, \mathsf{sk}_G) = 1 \bigwedge \mathsf{isGood}(\mathsf{pk}, \mathsf{sk}_G, \mathsf{dk}) = 1 \\ \bigwedge \mathsf{Verify}(\mathsf{pk}, \mathsf{sk}_B) = 1 \bigwedge \mathsf{isGood}(\mathsf{pk}, \mathsf{sk}_B, \mathsf{dk}) = 0\end{array}\right] = 1 - \mathsf{negl}(\kappa)$$

Re-Randomization: *We require that* $(\mathsf{pk}, s_G, \mathsf{sk}_0, \mathsf{sk}_1) \overset{c}{\approx} (\mathsf{pk}, s_G, \mathsf{sk}_0, \mathsf{sk}'_1)$ *where*

$(\mathsf{pk}, s_G, s_B, \mathsf{dk}) \leftarrow \mathsf{Setup}()$, $\mathsf{sk}_0 \leftarrow \mathsf{SampG}(s_G)$ *and* $\mathsf{sk}_1 \leftarrow \mathsf{Update}(\mathsf{sk}_0)$, $\mathsf{sk}'_1 \leftarrow \mathsf{SampG}(s_G)$

Hardness of Bad Keys: *Given* s_G*, it's hard to produce a valid "bad key". Formally, for any* PPT $\mathcal{A}$*,*

$$\Pr\left[\begin{array}{c}(\mathsf{pk}, s_G, s_B, \mathsf{dk}) \leftarrow \mathsf{Setup}(1^\kappa), \mathsf{sk}^* \leftarrow \mathcal{A}(\mathsf{pk}, s_G) : \\ \mathsf{Verify}(\mathsf{pk}, \mathsf{sk}^*) = 1 \bigwedge \mathsf{isGood}(\mathsf{pk}, \mathsf{sk}^*, \mathsf{dk}) = 0\end{array}\right] \leq \mathsf{negl}(\kappa)$$

Hardness of Good Keys: *Given* s_B*, it's hard to produce a valid "good key". Formally, for any* PPT $\mathcal{A}$*,*

$$\Pr\left[\begin{array}{c}(\mathsf{pk}, s_G, s_B, \mathsf{dk}) \leftarrow \mathsf{Setup}(1^\kappa), \mathsf{sk}^* \leftarrow \mathcal{A}(\mathsf{pk}, s_B) : \\ \mathsf{isGood}(\mathsf{pk}, \mathsf{sk}^*, \mathsf{dk}) = 1\end{array}\right] \leq \mathsf{negl}(\kappa)$$

μ **Leakage-Indistinguishability:** *Given both sampling keys* s_G, s_B*, and* μ *bits of leakage on a secret-key* sk *(which is either good or bad), it is hard to produce a secret-key* sk^* *which is in the same category as* sk*. Formally, for any* PPT *adversary* $\mathcal{A}$*, we have* $|\Pr[\mathcal{A} \text{ wins}] - 1/2| \leq \mathsf{negl}(\kappa)$ *in the following game:*

- *The challenger chooses* $(\mathsf{pk}, s_G, s_B, \mathsf{dk}) \leftarrow \mathsf{Setup}(1^\kappa)$ *and gives* pk, s_G, s_B *to* $\mathcal{A}$*. The challenger chooses a random bit* $b \in \{0, 1\}$*. If* $b = 1$ *then it samples* $\mathsf{sk} \leftarrow \mathsf{SampG}(s_G)$*, and otherwise it samples* $\mathsf{sk} \leftarrow \mathsf{SampB}(s_B)$*.*
- *The adversary* $\mathcal{A}$ *can make up to* q *queries in total to the leakage-oracle*
- *The adversary outputs* sk^* *and wins if* $\mathsf{isGood}(\mathsf{pk}, \mathsf{sk}^*, \mathsf{dk}) = b$*.*

6.2.2 Construction

A μ LIRR can be used to construct a μ-2CLR secure OWR, as follows:

- $\mathsf{Gen}(1^\kappa)$: Sample $(\mathsf{pk}, s_G, \cdot, \cdot) \leftarrow \mathsf{Setup}(1^\kappa)$, $\mathsf{sk} \leftarrow \mathsf{SampG}(s_G)$ and output $(\mathsf{pk}, \mathsf{sk})$
- $\mathsf{Update}(\cdot), \mathsf{Verify}(\cdot, \cdot)$: Same as for LIRR

Note that the CLR-OWR completely ignores the the bad sampling algorithm SampB, the "bad" sampling key s_B, the distinguishing algorithm isGood, and the distinguishing key dk of the LIRR. These are only used in the argument of security. Moreover, the "good" sampling key s_G is only used as an intermediate step during key-generation to sample the secret-key sk, but is never explicitly stored afterwards.

Theorem 6.3 *Given any* 2μ*-LIRR scheme, the construction above is a* μ*-2CLR secure OWR.*

Proof: The proof is very similar to that in [21]. To prove the theorem statement, we develop a sequence of games.

Game $\mathcal{H}_0$: This is the original μ-2CLR Game as Definition 6.1. The adversary is allowed to apply leakage function on consecutive two secret keys in each round of leakage attack.

Games $\mathcal{H}_{0.i}$ - $\mathcal{H}_1$: Let q be the total number of leakage rounds for which $\mathcal{A}$ runs. We define the Games $\mathcal{H}_{0.i}$ for $i = 0, 1, \ldots, q$ as follows. The challenger initially samples $(\mathsf{pk}, s_G, s_B, \mathsf{dk}) \leftarrow \mathsf{Setup}(1^\kappa)$, and $\mathsf{sk}_0 \leftarrow \mathsf{SampG}(s_G)$ and gives pk to $\mathcal{A}$. The game then proceeds as before with many leakage rounds, except that the secret keys in rounds $j \leq i$ are generated as $\mathsf{sk}_j \leftarrow \mathsf{SampG}(s_G)$, independently of all previous rounds, and in the rounds $j > i$, they are generated as $\mathsf{sk}_j \leftarrow \mathsf{Update}(\mathsf{sk}_{j-1})$. Note that Game $\mathcal{H}_{0.0}$ is the same as Game $\mathcal{H}_0$, and we define Game $\mathcal{H}_1$ to be the same as Game $\mathcal{H}_{0.q}$.

Claim 6.4 *For $i = 1, \ldots, q$, it holds that* $|\Pr[\mathcal{A} \text{ wins } |\mathcal{H}_{0.(i-1)}] - \Pr[\mathcal{A} \text{ wins } |\mathcal{H}_{0.i}]| \leq \mathsf{negl}(\kappa)$

Proof: We use the re-randomization property to argue that, for $i = 1, \ldots, q$, the winning probability of $\mathcal{A}$ is the same in Game $\mathcal{H}_{0.(i-1)}$ as in Game $\mathcal{H}_{0.i}$, up to negligible factors. We construct a reduction $\mathcal{B}$, with input $(\mathsf{pk}, s_G, \mathsf{sk}', \mathsf{sk}'')$. Here $\mathsf{sk}' \leftarrow \mathsf{SampG}(s_G)$, and sk'' is sampled based on randomly chosen b: if $b = 1$, then $\mathsf{sk}'' \leftarrow \mathsf{SampG}(s_G)$ and if $b = 0$, then $\mathsf{sk}'' \leftarrow \mathsf{Update}(\mathsf{sk}')$.

More concretely, the reduction $\mathcal{B}$ emulates a copy of $\mathcal{A}$ internally. In addition $\mathcal{B}$ emulates the view for $\mathcal{A}$: for all $j < i$, $\mathcal{B}$ generates $\mathsf{sk}_j \leftarrow \mathsf{SampG}(s_G)$, and for all $j > i + 1$, $\mathcal{B}$ generates $\mathsf{sk}_j \leftarrow \mathsf{Update}(\mathsf{sk}_{j-1})$; $\mathcal{B}$ sets $\mathsf{sk}_i := \mathsf{sk}'$ and $\mathsf{sk}_{i+1} := \mathsf{sk}''$. Upon receiving a leakage query f_j from $\mathcal{A}$, the reduction $\mathcal{B}$ returns $f_j(\mathsf{sk}_{j-1}, \mathsf{sk}_j)$ to $\mathcal{A}$.

If $\mathcal{B}$'s challenger uses sk'' which is generated through SampG then that corresponds to the view of $\mathcal{A}$ in Game $\mathcal{H}_{0.i}$ and if sk'' is generated through Update, then corresponds to Game $\mathcal{H}_{0.(i-1)}$. Therefore, if $\mathcal{A}$ is able to distinguish the two worlds, then $\mathcal{B}$ is able to break the re-randomization property. ■

Game $\mathcal{H}_2$: Game $\mathcal{H}_2$ is the same as Game $\mathcal{H}_1$, except the winning condition: now the adversary only wins if, at the end, it outputs sk^* such that $\mathsf{isGood}(\mathsf{pk}, \mathsf{sk}^*, \mathsf{dk}) = 1$.

Claim 6.5 $|\Pr[\mathcal{A} \text{ wins } |\mathcal{H}_1] - \Pr[\mathcal{A} \text{ wins } |\mathcal{H}_2]| \leq \mathsf{negl}(\kappa)$

Proof: The winning probability of $\mathcal{A}$ in Game $\mathcal{H}_2$ is at least that of Game $\mathcal{H}_1$ minus the probability that sk^* satisfies $\mathsf{Verify}(\mathsf{pk}, \mathsf{sk}^*) = 1 \wedge \mathsf{isGood}(\mathsf{pk}, \mathsf{sk}^*, \mathsf{dk}) = 0$. However, since the entire interaction between the challenger and the adversary in games $\mathcal{H}_1, \mathcal{H}_2$ can be simulated using (pk, s_G), we can use the "hardness of bad keys" property to argue that the probability of the above happening is negligible. Therefore the probability of $\mathcal{A}$ winning in Game $\mathcal{H}_2$ is at least that of Game $\mathcal{H}_1$, up to negligible factors. ■

Games $\mathcal{H}_{2.i}$ - $\mathcal{H}_3$: Let q be the total number of leakage rounds for which $\mathcal{A}$ runs. We define the Games $\mathcal{H}_{2.i}$ for $i = 0, 1, \ldots, q$ as follows. The challenger initially samples $(\mathsf{pk}, s_G, s_B, \mathsf{dk}) \leftarrow \mathsf{Gen}(1^\kappa)$ and gives pk to $\mathcal{A}$. The game then proceeds as before with many leakage rounds, except that the secret keys in rounds $j \leq i$ are generated as $\mathsf{sk}_j \leftarrow \mathsf{SampB}(s_B)$, and in the rounds $j > i$, they are generated as $\mathsf{sk}_j \leftarrow \mathsf{SampG}(s_G)$. Note that Game $\mathcal{H}_{2.0}$ is the same as Game $\mathcal{H}_2$, and we define Game $\mathcal{H}_3$ to be the same as Game $\mathcal{H}_{2.q}$.

Claim 6.6 *For $i = 1, \ldots, q$, it holds that* $|\Pr[\mathcal{A} \text{ wins } |\mathcal{H}_{2.(i-1)}] - \Pr[\mathcal{A} \text{ wins } |\mathcal{H}_{2.i}]| \leq \mathsf{negl}(\kappa)$.

Proof: We use the 2μ-Leakage Indistinguishability property to argue that, for $i = 1, \ldots, q$, the winning probability of $\mathcal{A}$ is the same in Game $\mathcal{H}_{2.(i-1)}$ as in Game $\mathcal{H}_{2.i}$, up to negligible factors. We construct a reduction $\mathcal{B}$, with input (pk, s_G, s_B) and with leakage access to sk. Here sk is sampled based on randomly chosen b: if $b = 1$, then $\mathsf{sk} \leftarrow \mathsf{SampG}(s_G)$ and if $b = 0$, then $\mathsf{sk} \leftarrow \mathsf{SampB}(s_B)$.

More concretely, the reduction $\mathcal{B}$ emulates a copy of $\mathcal{A}$ internally. In addition $\mathcal{B}$ emulates the view for $\mathcal{A}$: in each leakage-round $j < i$, $\mathcal{B}$ uses s_B to generate $\mathsf{sk}_j \leftarrow \mathsf{SampB}(s_B)$; and in round $j > i$, $\mathcal{B}$ uses s_G to generate $\mathsf{sk}_j \leftarrow \mathsf{SampG}(s_G)$. Upon receiving a leakage query f_j from $\mathcal{A}$ in leakage round j, if $j < i - 1$, $\mathcal{B}$

returns $f_j(\mathsf{sk}_{j-1}, \mathsf{sk}_j)$ to $\mathcal{A}$; if $j = i-1$, $\mathcal{B}$ defines $\hat{f}_j = f_j(\mathsf{sk}_{j-1}, \cdot)$, and then applies $\hat{f}_j$ on sk, and returns $\hat{f}_j(\mathsf{sk})$ to $\mathcal{A}$; if $j = i$, $\mathcal{B}$ defines $\hat{f}_j = f_j(\mathsf{sk}_j, \cdot)$, and then applies $\hat{f}_j$ on sk, and returns $\hat{f}_j(\mathsf{sk})$ to $\mathcal{A}$; if $j > i$, $\mathcal{B}$ returns $f_j(\mathsf{sk}_{j-1}, \mathsf{sk}_j)$ to $\mathcal{A}$. At the end, $\mathcal{B}$ outputs the value sk^* output by $\mathcal{A}$. Note that $\mathcal{B}$ made queries $\hat{f}_{i-1}$ and $\hat{f}_i$, which are 2μ bits in total.

If $\mathcal{B}$'s challenger uses a good key then that corresponds to the view of $\mathcal{A}$ in Game $\mathcal{H}_{2.i}$ and a bad key corresponds to Game $\mathcal{H}_{2.(i-1)}$. Therefore, letting b be the bit used by $\mathcal{B}$'s challenger, we have:

$$\begin{aligned}
&|\Pr[\mathcal{B} \text{ wins }] - 1/2| \\
&\quad = |\Pr[\mathsf{isGood}(\mathsf{pk}, \mathsf{sk}^*, \mathsf{dk}) = b] - 1/2| \\
&\quad = 1/2 \cdot |\Pr[\mathsf{isGood}(\mathsf{pk}, \mathsf{sk}^*, \mathsf{dk}) = 1 | b = 1] - \Pr[\mathsf{isGood}(\mathsf{pk}, \mathsf{sk}^*, \mathsf{dk}) = 1 | b = 0]| \\
&\quad = 1/2 \cdot \left|\Pr[\mathcal{A} \text{ wins } |\mathcal{H}_{2.(i-1)}] - \Pr[\mathcal{A} \text{ wins } |\mathcal{H}_{2.i}]\right|
\end{aligned}$$

■

Claim 6.7 $\Pr[\mathcal{A} \text{ wins } |\mathcal{H}_3] \leq \mathsf{negl}(\kappa)$

Proof: We now argue that probability of $\mathcal{A}$ winning Game $\mathcal{H}_3$ is negligible, by the "hardness of good keys". Notice that $\mathcal{A}$'s view in Game $\mathcal{H}_3$ can be simulated entirely just given (pk, s_B). Therefore, there is a PPT algorithm which, given (pk, s_B) as inputs, can run Game $\mathcal{H}_3$ with $\mathcal{A}$ and output sk^* such that $\mathsf{isGood}(\mathsf{pk}, \mathsf{sk}^*, \mathsf{dk}) = 1$ whenever $\mathcal{A}$ wins. So the probability of $\mathcal{A}$ winning in Game $\mathcal{H}_3$ is negligible. ■

By the hybrid argument, the probability of $\mathcal{A}$ winning in Game $\mathcal{H}_0$ is at most that of $\mathcal{A}$ winning in Game $\mathcal{H}_3$, up to negligible factors. That is, $\Pr[\mathcal{A} \text{ wins } |\mathcal{H}_0] \leq \mathsf{negl}(\kappa)$. Therefore, since the latter is negligible, the former must be negligible as well, which concludes the proof of the theorem. ■

Based on the result in [21], we have the following corollary.

Corollary 6.8 *Fix a constant $K \geq 1$, and assume that the K-linear assumption holds in the base groups of some pairing. Then, for any constant $\epsilon > 0$, there exists a μ-2CLR secure OWR scheme with relative-leakage $\frac{\mu}{|\mathsf{sk}|} \geq \frac{1}{2(K+1)} - \epsilon$.*

Furthermore, based our transformation in Section 3, we obtain a OWR scheme against continual leakage on update.

Corollary 6.9 *Under the K-linear assumption, and differing inputs obfuscation there exist μ-CLR OWRs with leak on key updates, where $\frac{\mu}{|\mathsf{sk}|} \geq \frac{1}{2(K+1)} - \epsilon$, for some negligible ϵ. The resulting scheme achieves leakage rate $\frac{1}{4(K+1)} - o(1)$.*

6.3 A Generic Construction based on PKE

In this section, we describe a generic construction of CLR-secure OWR (resp., 2CLR-secure, and CLR with leakage on key updates) from CLR-secure PKE (resp., 2CLR-secure, and CLR with leakage on key updates). A OWR requires verification of the relation to be deterministic; but a PKE does not necessarily give a OWR because there might not be a deterministic way to check the key pair $(\mathsf{pk}, \mathsf{sk})$ of a PKE. Here we present a way to check the key pair of a PKE *deterministically*, so that one can use PKE to construct OWR.

Theorem 6.10 *Let $\mathcal{E}$ be a public key encryption scheme secure in the model of CLR (respectively, of 2CLR, and of CLR with leakage on key updates) with leakage rate ρ, then for appropriate choice of polynomial $p(\cdot)$, the one way relation scheme* OWR *in Figure 22 is secure in the model of CLR (respectively, of 2CLR, and of CLR with leakage on key updates) with leakage rate ρ.*

Construction $\mathsf{OWR} = \mathsf{OWR}.\{\mathsf{Gen}, \mathsf{Update}, \mathsf{Verify}\}$

Here $\mathcal{E} = \mathcal{E}.\{\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Update}\}$ is a PKE scheme, and $p(\cdot)$ is some polynomial.

Key Generation: $(\mathsf{pk}, \mathsf{sk}_0) \leftarrow \mathsf{OWR.Gen}(1^\kappa)$

- Compute $(\mathsf{pk}^{\mathcal{E}}, \mathsf{sk}_0^{\mathcal{E}}) \leftarrow \mathcal{E}.\mathsf{Gen}(1^\kappa)$.
- Choose $p = p(\kappa)$ random messages $m_1, \ldots, m_p$ from the message space of $\mathcal{E}$.
- For $i \in [p]$, compute a random encryption $e_i = \mathcal{E}.\mathsf{Enc}(m_i)$.
- Ouput public key $\mathsf{pk} = (\mathsf{pk}^{\mathcal{E}}, (m_1, e_1), \ldots, (m_p, e_p))$ and secret key $\mathsf{sk}_0 = \mathsf{sk}_0^{\mathcal{E}}$.

Key Update: $\mathsf{sk}_{i+1} \leftarrow \mathsf{OWR.Update}(\mathsf{sk}_i)$

Set $\mathsf{sk}_{i+1} \leftarrow \mathcal{E}.\mathsf{Update}(\mathsf{sk}_i)$.

Verification: $b \leftarrow \mathsf{OWR.Verify}(\mathsf{sk}', \mathsf{pk})$

Upon receiving key pair $(\mathsf{sk}', \mathsf{pk})$, parse $\mathsf{pk} = (\mathsf{pk}^{\mathcal{E}}, (m_1, e_1), \ldots, (m_p, e_p))$.

If for all $i \in [p]$, $\mathcal{E}.\mathsf{Dec}(\mathsf{sk}', e_i) = m_i$, then set $b := 1$; otherwise, set $b := 0$.

Figure 22: The transformation of PKE to OWR.

Proof: [Sketch.] A well-known result from learning theory known as *Occam's Razor* (see for example, Kearns and Vazirani [39], Theorem 2.1)[11], says that if a class of circuits has size $|\mathcal{C}|$ and a circuit $C \in \mathcal{C}$ agrees with a target circuit $C^* \in \mathcal{C}$ on $\mathrm{poly}(\log(|\mathcal{C}|), 1/\epsilon, \log(1/\delta))$ number of random inputs, then with probability $1 - \delta$, C agrees with C^* over the uniform distribution with probability $1 - \epsilon$. In the following, we will always set $\log(1/\delta) \geq \kappa$ and so $\delta \leq 1/2^\kappa$.

Assume we have an adversary $\mathcal{A}$ breaking the security of the one-way relation, we use it to construct an adversary $\mathcal{A}'$ breaking the security of the encryption scheme $\mathcal{E}$. The class $\mathcal{C}$ consists of the circuits $\mathcal{E}.\mathsf{Dec}(\widetilde{\mathsf{sk}}, \cdot)$ for all possible sk. Clearly, $\log(|\mathcal{C}|) = |\mathsf{sk}| = \mathrm{poly}(\kappa)$. Now, C corresponds to the circuit $\mathcal{E}.\mathsf{Dec}(\mathsf{sk}', \cdot)$, where sk' is the secret key submitted by $\mathcal{A}$ such that $\mathsf{OWR.Verify}(\mathsf{sk}', \mathsf{pk}) = 1$. Furthermore, C^* is the circuit $\mathcal{E}.\mathsf{Dec}(\mathsf{sk}, \cdot)$, where sk is a real secret key. Note that C and C^* agree on $p(\kappa) = \mathrm{poly}(\log(|\mathcal{C}|), 1/\epsilon, 1/\log(\delta))$ random inputs, since $\mathcal{E}.\mathsf{Verify}(\mathsf{sk}', \mathsf{pk}) = 1$. Thus, we are guaranteed that with probability $1 - \delta$ over choice of input/output pairs (m_i, e_i) in pk, sk' decrypts correctly on a fresh random input with probability $1 - \epsilon$. We are now ready to define the adversary $\mathcal{A}'$.

$\mathcal{A}'$ internally instantiates $\mathcal{A}$, while participating externally in a leakage (resp., on consecutive two keys, and on both key and update) on encryption scheme $\mathcal{E}$. Specifically, $\mathcal{A}'$ does the following:

- Upon receiving $\mathsf{pk}^{\mathcal{E}}$ from the external experiment, do the following:
  - Choose $p = p(\kappa)$ random messages $m_1, \ldots, m_p$ from the message space of $\mathcal{E}$.
  - For $i \in [p]$, compute a random encryption $e_i = \mathcal{E}.\mathsf{Enc}(m_i)$.
  - Ouput public key $\mathsf{pk} = (\mathsf{pk}^{\mathcal{E}}, (m_1, e_1), \ldots, (m_p, e_p))$ to the internal adversary $\mathcal{A}$. Note that secret key $\mathsf{sk}_0 = \mathsf{sk}_0^{\mathcal{E}}$ is a correctly distributed secret key for this pk.
- Whenever $\mathcal{A}$ submits a leakage query f, $\mathcal{A}'$ submits the same query to its external challenger who applies it to the secret key (resp., on consecutive two keys, and on both key and update) and forwards the answer to $\mathcal{A}$.

[11] We note that the statement of Occam's Razor theorem in [39] is for the case of Boolean functions. However, the analysis can be easily extended to the non-Boolean case.

- Finally, $\mathcal{A}$ submits sk' to $\mathcal{A}'$. If there exists $i \in [p]$ such that $\mathcal{E}.\mathsf{Dec}(\mathsf{sk}', e_i) \neq m_i$, then $\mathcal{A}'$ outputs random b'.
- Otherwise, $\mathcal{A}'$ chooses two independent, uniformly random messages m_0, m_1 and submits to its external challenger.
- $\mathcal{A}'$ then receives the challenge ciphertext c^*.
- $\mathcal{A}'$ computes $m^* = \mathcal{E}.\mathsf{Dec}(\mathsf{sk}', c^*)$. If $m^* = m_0$, $\mathcal{A}'$ outputs 0. Otherwise, $\mathcal{A}'$ outputs 1.

Note that $\mathcal{A}'$ perfectly simulates $\mathcal{A}$'s view in the OWR game. Therefore, it is not hard to see that if $\mathcal{A}$ succeeds with probability $p_1 = p_1(\kappa) \geq 8/2^\kappa$, then $\mathcal{A}'$ succeeds with probability $1/2 \cdot (1-p_1) + (p_1 - \delta)(1-\epsilon)$. For $\epsilon \leq 1/7$, we have that $(p_1 - \delta)(1-\epsilon) \geq 3p_1/4$. Thus, $\mathcal{A}'$ succeeds with probability $1/2 + p_1/4$ and obtains advantage $\mathsf{Adv}_{\mathcal{A}',\mathcal{E}} = p_1/4$. This, in turn, implies that $\mathcal{A}$ must succeed with negligible $p_1(\kappa)$ probability, since otherwise we contradict the security of $\mathcal{E}$.

■

7 Continual Leakage Resilience for Digital Signatures

In the previous Section, we constructed one-way relations secure against continual leakage and leak on update. By combining our results with the work of Dodis et al. [21] we obtain a continual leakage resilient signature scheme with leakage on update (but no leakage on the randomness used for signing). By combining our results with the work of Boyle et al. [11] we obtain a continual, fully leakage resilient signature scheme with leakage on update. By "fully" leakage resilient signature scheme we mean a signature scheme which allows leakage on the randomness used for signing.

In addition to applications for constructing digital signature schemes, by combining our OWR constructions with the results of [21], we also immediately obtain constructions of identification (ID) schemes and authenticated key agreement (AKA) protocols.

7.1 Continual Leakage Model

A digital signature scheme SIG consists of three algorithms: $(\mathsf{SIG.Gen}, \mathsf{SIG.Sign}, \mathsf{SIG.Verify})$. In the continual leakage setting, we require an additional algorithm $\mathsf{SIG.Update}$ which updates the secret keys. Note that the verification key remains unchanged.

- $\mathsf{SIG.Gen}(1^\kappa) \rightarrow (\mathsf{vk}, \mathsf{sk}_0)$. The key generation algorithm takes in the security parameter κ, and outputs a secret key sk_0 and a public verification key vk.
- $\mathsf{SIG.Sign}(m, \mathsf{sk}_i) \rightarrow \sigma$. The signing algorithm takes in a message m and a secret key sk_i, and outputs a signature σ.
- $\mathsf{SIG.Verify}(\mathsf{vk}, \sigma, m) \rightarrow \{0, 1\}$. The verification algorithm takes in the verification key vk, a signature σ, and a message m. It outputs either 0 or 1.
- $\mathsf{SIG.Update}(\mathsf{sk}_{i-1}) \rightarrow \mathsf{sk}_i$. The update algorithm takes in a secret key sk_{i-1} and produces a new secret key sk_i for the *same* verification key.

Correctness. The signature scheme satisfies correctness if $\mathsf{SIG.Verify}(\mathsf{vk}, \sigma, m)$ outputs 1 whenever $\mathsf{vk}, \mathsf{sk}_0$ is produced by $\mathsf{SIG.Gen}$, and σ is produced by $\mathsf{SIG.Sign}(m, \mathsf{sk}_i)$ for some sk_i obtained by calls to $\mathsf{SIG.Update}$, starting with sk_0. (If the verification algorithm is randomized, we may relax this requirement to hold with all but negligible probability.)

Security. We define continual leakage security for signatures in terms of the following game between a challenger and an attacker (this extends the usual notion of existential unforgeability to our leakage setting). The game is parameterized by two values: the security parameter κ, and the parameter μ which controls the amount of leakage allowed. For the sake of simplicity, we assume that the signing algorithm calls the update algorithm on each invocation. Since updates in our scheme do occur with each signature, we find it more convenient to work with the simplified definition given below.

Setup Phase The game begins with a setup phase. The challenger calls $\mathsf{Gen}(1^\kappa)$ to create the signing key, sk_0, and the verification key, vk. It gives vk to the attacker. No leakage is allowed in this phase.

Query Phase. In this phase, the attacker launches a polynomial number of signing queries and leakage queries. Each time, say in the i-th query, the attacker specifies a message m_i and provides an efficiently computable leakage function f_i whose output is at most μ bits, and the challenger chooses randomness r_i, updates the secret key from sk_{i-1} to sk_i, and gives the attacker the corresponding signature for message m_i as well as the leakage response ℓ_i. In the CLR model, the leakage attack is applied on a single secret key, and the leakage response $\ell_i = f_i(\mathsf{sk}_{i-1})$. In the 2CLR model, the leakage attack is applied on consecutive two secret keys, i.e., $\ell_i = f_i(\mathsf{sk}_{i-1}, \mathsf{sk}_i)$. In the CLR with leakage on key updates, the leakage attack is applied on the current secret key and the randomness used for updating the secret key, i.e., $\ell_i = f_i(\mathsf{sk}_{i-1}, r_i)$.

Forgery Phase The attacker gives the challenger a message, m^*, and a signature σ^* such that m^* has not been previously queried. The attacker wins the game if (m^*, σ^*) passes the verification algorithm using vk.

Definition 7.1 (Continual Leakage Resilience) *We say a Digital Signature scheme is μ-CLR secure (respectively, μ-2CLR secure, or μ-CLR secure with leakage on key updates) if any* PPT *attacker only has a negligible advantage (negligible in κ) in the above game.*

7.2 NIZK and True-Simulation Extractability

Dodis et al. [21] constructed CLR-secure signature based on CLR-secure OWR and another primitive named *true-simulation extractable (tSE)* NIZK. We here recall the syntax and security properties of NIZK. We note that, the definitions below are taken from [21] for completeness.

Let R be an NP relation on pairs (y, x) with corresponding language $L_R = \{y \mid \exists x \text{ s. t. } (y, x) \in R\}$. A NIZK argument for a relation R consists of four PPT algorithms $(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify}, \mathsf{Sim})$ with syntax:

- $(\mathsf{crs}, \mathsf{tk}) \leftarrow \mathsf{Setup}(1^\kappa)$: Creates a common reference string (CRS) and a trapdoor key to the CRS.
- $\pi \leftarrow \mathsf{Prove}_{\mathsf{crs}}(y, x)$: Creates an argument that $y \in L_R$.
- $\mathsf{Sim}_{\mathsf{crs}}(y, \mathsf{tk})$: Creates a simulated argument that $y \in L_R$.
- $b \leftarrow \mathsf{Verify}_{\mathsf{crs}}(y, \pi)$: Verifies whether or not the argument π is correct.

For the sake of clarity, we write $\mathsf{Prove}, \mathsf{Verify}, \mathsf{Sim}$ without the crs in the subscript when the crs can be inferred from the context.

Definition 7.2 *We say that* $(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ *are a NIZK argument system for the relation R if the following three properties hold.*

Completeness: *For any $(y, x) \in R$, if $(\mathsf{crs}, \mathsf{tk}) \leftarrow \mathsf{Setup}(1^\kappa)$, $\pi \leftarrow \mathsf{Prove}(y, x)$, then $Verify(y, \pi) = 1$.*

Soundness: *For any* PPT *adversary $\mathcal{A}$,*

$$\Pr[\mathsf{Verify}(y, \pi^*) = 1 \wedge y \notin L_R \ : \ (\mathsf{crs}, \mathsf{tk}) \leftarrow \mathsf{Setup}(1^\kappa), (y, \pi^*) \leftarrow \mathcal{A}(\mathsf{crs})] \leq \mathsf{negl}(1^\kappa)$$

Composable Zero-Knowledge: *For any* PPT *adversary* $\mathcal{A}$ *we have* $\Pr[\mathcal{A}\text{wins }] - 1/2 \leq \mathsf{negl}(1^\kappa)$ *in the following game:*

- *The challenger samples* $(\mathsf{crs}, \mathsf{tk}) \leftarrow \mathsf{Setup}(1^\kappa)$ *and gives* $(\mathsf{crs}, \mathsf{tk})$ *to* $\mathcal{A}$.
- *The adversary* $\mathcal{A}$ *chooses* $(y, x) \in R$ *and gives these to the challenger.*
- *The challenger samples* $\pi_0 \leftarrow \mathsf{Prove}(y, x)$, $\pi_1 \leftarrow \mathsf{Sim}(y, \mathsf{tk})$, $b \leftarrow \{0, 1\}$ *and gives* π_b *to* $\mathcal{A}$.
- *The adversary* $\mathcal{A}$ *outputs a bit* b'*, and wins if* $b' = b$.

Definition 7.3 (True-Simulation Extractability [22]) *Let* $\mathsf{NIZK} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify}, \mathsf{Sim})$ *be an NIZK argument for an NP relation* R*, satisfying the completeness, soundness and zero-knowledge properties. We say that* NIZK *is* true-simulation extractable (tSE) *if:*

- *Apart from outputting a CRS and a trapdoor key,* Setup *also outputs an extraction key:* $(\mathsf{crs}, \mathsf{tk}, \mathsf{ek}) \leftarrow \mathsf{Setup}(1^\kappa)$.
- *There exists a* PPT *algorithm* $\mathsf{Ext}_{\mathsf{ek}}$ *such that for all* $\mathcal{A}$ *we have* $\Pr[\mathcal{A}\text{wins }] \leq \mathsf{negl}(1^\kappa)$ *in the following game:*
  1. *The challenger runs* $(\mathsf{crs}, \mathsf{tk}, \mathsf{ek}) \leftarrow \mathsf{Setup}(1^\kappa)$ *and gives* crs *to* $\mathcal{A}$.
  2. $\mathcal{A}^{\mathcal{SIM}_{\mathsf{tk}}()}$ *is given access to a simulation oracle* $\mathcal{SIM}_{\mathsf{tk}}()$*, which it can adaptively access. A query to the simulation oracle consists of a pair* (y, x)*. The oracle checks if* $(y, x) \in R$*. If true, it ignores* x *and outputs a simulated argument* $\mathsf{Sim}_{\mathsf{tk}}(y)$*. Otherwise, the oracle outputs* $\perp$.
  3. $\mathcal{A}$ *outputs a pair* (y^*, σ^*)*, and the challenger runs* $x^* \leftarrow \mathsf{Ext}_{\mathsf{ek}}(y^*, \sigma^*)$.

$\mathcal{A}$ *wins if* $(y, x^*) \notin R$, $\mathsf{Verify}(y^*, \sigma^*) = 1$*, and* y^* *was not part of a query to the simulation oracle.*

7.3 Construction from OWR

Next we recall Dodis et al's construction and then show that actually their construction is 2CLR-secure if the underlying OWR is 2CLR-secure. In the following, $\mathsf{OWR} := (\mathsf{OWR.Gen}(1^\kappa), \mathsf{OWR.Update}(\mathsf{sk}))$ is a 2CLR-secure one-way relation and $\mathsf{NIZK} := (\mathsf{NIZK.Setup}, \mathsf{NIZK.Prove}, \mathsf{NIZK.Verify})$ is a is tSE-NIZK for the relation

$$R = \{(y, x) \mid y = (\mathsf{pk}, m), x = \mathsf{sk} \text{ s.t. } \mathsf{Verify}(\mathsf{pk}, \mathsf{sk}) = 1\}.$$

Although input m seems useless in the above relation, looking ahead, m will play the role of the message to be signed. Note that we have the important property that when the message m changes, the statement $y = (\mathsf{pk}, m)$ also changes.

- $\mathsf{SIG.Gen}(1^\kappa)$: Output $(\mathsf{vk}, \mathsf{sk})$ where $\mathsf{vk} = (\mathsf{pk}, \mathsf{crs})$, $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{OWR.Gen}(1^\kappa)$ and $\mathsf{crs} \leftarrow \mathsf{NIZK.Setup}(1^\kappa)$.
- $\mathsf{SIG.Sign}_{\mathsf{sk}}(m)$: Output $\sigma \leftarrow \mathsf{NIZK.Prove}((\mathsf{pk}, m), \mathsf{sk})$.
- $\mathsf{SIG.Verify}_{\mathsf{vk}}(m, \sigma)$: Output $b := \mathsf{NIZK.Verify}(\mathsf{pk}, m), \sigma)$.
- $\mathsf{SIG.Update}(\mathsf{sk})$: Output $\mathsf{OWR.Update}(\mathsf{sk})$.

Theorem 7.4 *If one-way relation* OWR *is* μ*-2CLR secure and* NIZK *is true-simulation extractable, then the above signature scheme is* μ*-2CLR secure.*

Proof: The proof here is very similar to that in [21]. We prove the above theorem through a sequence of games.

Game $\mathcal{H}_0$: This is the original μ-2CLR game described in Definition 7.1, in which signing queries are answered honestly by running $\sigma \leftarrow \mathsf{NIZK.Prove}((\mathsf{pk}, m), \mathsf{sk})$ and $\mathcal{A}$ wins if she produces a valid forgery (m^*, σ^*).

Game $\mathcal{H}_1$: In this game, the signing queries are answered by generating simulated arguments, i.e., $\sigma \leftarrow \mathsf{NIZK.Sim}_{\mathsf{tk}}(\mathsf{pk}, m)$. Games $\mathcal{H}_0$ and $\mathcal{H}_1$ are indistinguishable by the zero-knowledge property of NIZK. Here the simulated arguments given to $\mathcal{A}$ as answers to signing queries are always of true statements.

Game $\mathcal{H}_2$: In this game, we modify the winning condition so that the adversary only wins if it produces a valid forgery (m^*, σ^*) and the challenger is able to extract a valid secret key sk^* for pk from (m^*, σ^*). That is, $\mathcal{A}$ wins if both $\mathsf{SIG.Verify}(m^*, \sigma^*) = 1$ and $\mathsf{OWR.Verify}(\mathsf{pk}, \mathsf{sk}^*) = 1$, where $\mathsf{sk}^* \leftarrow \mathsf{NIZK.Ext}_{\mathsf{ek}}((\mathsf{pk}, m^*), \sigma^*)$. The winning probability of $\mathcal{A}$ in Game $\mathcal{H}_2$ is at least that of Game $\mathcal{H}_1$ minus the probability that $\mathsf{NIZK.Verify}((\mathsf{pk}, m^*), \sigma^*) = 1$ and $\mathsf{OWR.Verify}(\mathsf{pk}, \mathsf{sk}^*) = 0$. By the true-simulation extractability of the argument NIZK we know that this probability is negligible. Therefore, the winning probability of $\mathcal{A}$ in Game $\mathcal{H}_2$ differs from that in Game $\mathcal{H}_1$ by a negligible amount.

We have shown that the probability that $\mathcal{A}$ wins in Game $\mathcal{H}_0$ is the same as that in Game $\mathcal{H}_2$, up to negligible factors. We now argue that the probability that $\mathcal{A}$ wins in Game $\mathcal{H}_2$ is negligible, which proves that the probability that $\mathcal{A}$ wins in Game $\mathcal{H}_0$ is negligible as well. To prove that the probability that $\mathcal{A}$ wins in Game $\mathcal{H}_2$ is negligible, we assume otherwise and show that there exists a PPT algorithm $\mathcal{B}$ that breaks the μ-2CLR security of OWR. On input pk, $\mathcal{B}$ generates $(\mathsf{crs}, \mathsf{tk}, \mathsf{ek}) \leftarrow \mathsf{NIZK.Setup}(1^\kappa)$ and emulates $\mathcal{A}$ on input $\mathsf{vk} = (\mathsf{crs}, \mathsf{pk})$. In each leakage round, $\mathcal{B}$ answers $\mathcal{A}$'s leakage queries using the leakage oracle and answers signing queries m_i by creating simulated arguments $\sigma_i \leftarrow \mathsf{NIZK.Sim}_{\mathsf{tk}}(\mathsf{pk}, m_i)$. When $\mathcal{A}$ outputs her forgery (m^*, σ^*), $\mathcal{B}$ runs $\mathsf{sk}^* \leftarrow \mathsf{NIZK.Ext}_{\mathsf{ek}}((\mathsf{pk}, m^*), \sigma^*)$ and outputs sk^*. Notice that $\Pr[\mathcal{B} \text{ wins }] = \Pr[\mathcal{A} \text{ wins }]$, so that if $\Pr[\mathcal{A} \text{ wins }]$ is non-negligible then $\mathcal{B}$ breaks the μ-consecutive two-key security of OWR. We therefore conclude that the probability that $\mathcal{A}$ wins in Game $\mathcal{H}_2$ is negligible. This concludes the proof of the theorem. ∎

Based on the result in [21], we have the following corollary.

Corollary 7.5 *Fix a constant $K \geq 1$, and assume that the K-linear assumption holds in the base groups of some pairing. Then, for any constant $\epsilon > 0$, there exists a μ-2CLR secure signature scheme with relative-leakage $\frac{\mu}{|\mathsf{sk}|} \geq \frac{1}{2(K+1)} - \epsilon$.*

Acknowledgements

We thank the anonymous reviewers for their helpful comments.

References

[1] M. Abe, M. Chase, B. David, M. Kohlweiss, R. Nishimaki, and M. Ohkubo. Constant-size structure-preserving signatures: Generic constructions and simple assumptions. In X. Wang and K. Sako, editors, *ASIACRYPT 2012*, volume 7658 of *LNCS*, pages 4–24. Springer, Dec. 2012.

[2] A. Akavia, S. Goldwasser, and V. Vaikuntanathan. Simultaneous hardcore bits and cryptography against memory attacks. In O. Reingold, editor, *TCC 2009*, volume 5444 of *LNCS*, pages 474–495. Springer, Mar. 2009.

[3] P. Ananth, D. Boneh, S. Garg, A. Sahai, and M. Zhandry. Differing-inputs obfuscation and applications. Cryptology ePrint Archive, Report 2013/689, 2013. http://eprint.iacr.org/2013/689.

[4] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. P. Vadhan, and K. Yang. On the (im)possibility of obfuscating programs. In J. Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 1–18. Springer, Aug. 2001.

[5] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. P. Vadhan, and K. Yang. On the (im)possibility of obfuscating programs. *Journal of the ACM*, 59(2):6, 2012.

[6] A. Boldyreva, S. Fehr, and A. O'Neill. On notions of security for deterministic encryption, and efficient constructions without random oracles. In D. Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 335–359. Springer, Aug. 2008.

[7] D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In M. Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 41–55. Springer, Aug. 2004.

[8] D. Boneh and B. Waters. Constrained pseudorandom functions and their applications. In K. Sako and P. Sarkar, editors, *ASIACRYPT 2013, Part II*, volume 8270 of *LNCS*, pages 280–300. Springer, Dec. 2013.

[9] E. Boyle, K.-M. Chung, and R. Pass. On extractability obfuscation. In Y. Lindell, editor, *TCC 2014*, volume 8349 of *LNCS*, pages 52–73. Springer, Feb. 2014.

[10] E. Boyle, S. Goldwasser, and I. Ivan. Functional signatures and pseudorandom functions. In H. Krawczyk, editor, *PKC 2014*, volume 8383 of *LNCS*, pages 501–519. Springer, Mar. 2014.

[11] E. Boyle, G. Segev, and D. Wichs. Fully leakage-resilient signatures. In K. G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 89–108. Springer, May 2011.

[12] Z. Brakerski, Y. T. Kalai, J. Katz, and V. Vaikuntanathan. Overcoming the hole in the bucket: Public-key cryptography resilient to continual memory leakage. In *51st FOCS*, pages 501–510. IEEE Computer Society Press, Oct. 2010.

[13] R. Canetti, C. Dwork, M. Naor, and R. Ostrovsky. Deniable encryption. In B. S. Kaliski Jr., editor, *CRYPTO'97*, volume 1294 of *LNCS*, pages 90–104. Springer, Aug. 1997.

[14] R. Canetti, S. Goldwasser, and O. Poburinnaya. Adaptively secure two-party computation from indistinguishability obfuscation. In Y. Dodis and J. B. Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 557–585. Springer, Mar. 2015.

[15] R. Canetti, H. Krawczyk, and J. B. Nielsen. Relaxing chosen-ciphertext security. In D. Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 565–582. Springer, Aug. 2003.

[16] S. Chari, C. S. Jutla, J. R. Rao, and P. Rohatgi. Towards sound approaches to counteract power-analysis attacks. In M. J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 398–412. Springer, Aug. 1999.

[17] M. Chase, M. Kohlweiss, A. Lysyanskaya, and S. Meiklejohn. Malleable proof systems and applications. In D. Pointcheval and T. Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 281–300. Springer, Apr. 2012.

[18] D. Dachman-Soled, S. D. Gordon, F.-H. Liu, A. O'Neill, and H.-S. Zhou. Leakage-resilient public-key encryption from obfuscation. 2016. Full version.

[19] D. Dachman-Soled, J. Katz, and V. Rao. Adaptively secure, universally composable, multiparty computation in constant rounds. In Y. Dodis and J. B. Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 586–613. Springer, Mar. 2015.

[20] D. Dachman-Soled, F.-H. Liu, and H.-S. Zhou. Leakage-resilient circuits revisited - optimal number of computing components without leak-free hardware. In E. Oswald and M. Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 131–158. Springer, Apr. 2015.

[21] Y. Dodis, K. Haralambiev, A. López-Alt, and D. Wichs. Cryptography against continuous memory attacks. In *51st FOCS*, pages 511–520. IEEE Computer Society Press, Oct. 2010.

[22] Y. Dodis, K. Haralambiev, A. López-Alt, and D. Wichs. Efficient public-key cryptography in the presence of key leakage. In M. Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 613–631. Springer, Dec. 2010.

[23] Y. Dodis, Y. T. Kalai, and S. Lovett. On cryptography with auxiliary input. In M. Mitzenmacher, editor, *41st ACM STOC*, pages 621–630. ACM Press, May / June 2009.

[24] Y. Dodis, A. B. Lewko, B. Waters, and D. Wichs. Storing secrets on continually leaky devices. In R. Ostrovsky, editor, *52nd FOCS*, pages 688–697. IEEE Computer Society Press, Oct. 2011.

[25] Y. Dodis, R. Ostrovsky, L. Reyzin, and A. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM J. Comput.*, 38(1):97–139, 2008.

[26] Y. Dodis and A. Smith. Correcting errors without leaking partial information. In H. N. Gabow and R. Fagin, editors, *37th ACM STOC*, pages 654–663. ACM Press, May 2005.

[27] S. Faust, T. Rabin, L. Reyzin, E. Tromer, and V. Vaikuntanathan. Protecting circuits from leakage: the computationally-bounded and noisy cases. In H. Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 135–156. Springer, May 2010.

[28] S. Garg, C. Gentry, and S. Halevi. Candidate multilinear maps from ideal lattices. In T. Johansson and P. Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 1–17. Springer, May 2013.

[29] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th FOCS*, pages 40–49. IEEE Computer Society Press, Oct. 2013.

[30] S. Garg, C. Gentry, S. Halevi, and D. Wichs. On the implausibility of differing-inputs obfuscation and extractable witness encryption with auxiliary input. In J. A. Garay and R. Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 518–535. Springer, Aug. 2014.

[31] S. Garg and A. Polychroniadou. Two-round adaptively secure MPC from indistinguishability obfuscation. In Y. Dodis and J. B. Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 614–637. Springer, Mar. 2015.

[32] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *J. ACM*, 33(4):792–807, Aug. 1986.

[33] S. Goldwasser and G. N. Rothblum. On best-possible obfuscation. In S. P. Vadhan, editor, *TCC 2007*, volume 4392 of *LNCS*, pages 194–213. Springer, Feb. 2007.

[34] J. A. Halderman, S. D. Schoen, N. Heninger, W. Clarkson, W. Paul, J. A. Calandrino, A. J. Feldman, J. Appelbaum, and E. W. Felten. Lest we remember: Cold boot attacks on encryption keys. In *USENIX Security Symposium*, pages 45–60, 2008.

[35] C. Hazay, A. López-Alt, H. Wee, and D. Wichs. Leakage-resilient cryptography from minimal assumptions. In T. Johansson and P. Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 160–176. Springer, May 2013.

[36] R. Impagliazzo, L. A. Levin, and M. Luby. Pseudo-random generation from one-way functions (extended abstracts). In *21st ACM STOC*, pages 12–24. ACM Press, May 1989.

[37] Y. Ishai, O. Pandey, and A. Sahai. Public-coin differing-inputs obfuscation and its applications. In Y. Dodis and J. B. Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 668–697. Springer, Mar. 2015.

[38] J. Katz and V. Vaikuntanathan. Signature schemes with bounded leakage resilience. In M. Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 703–720. Springer, Dec. 2009.

[39] M. J. Kearns and U. V. Vazirani. *An introduction to computational learning theory*. Massachusetts Institute of Technology, 1994.

[40] A. Kiayias, S. Papadopoulos, N. Triandopoulos, and T. Zacharias. Delegatable pseudorandom functions and applications. In A.-R. Sadeghi, V. D. Gligor, and M. Yung, editors, *ACM CCS 13*, pages 669–684. ACM Press, Nov. 2013.

[41] C.-J. Lee, C.-J. Lu, S.-C. Tsai, and W.-G. Tzeng. Extracting randomness from multiple independent sources. *IEEE Transactions on Information Theory*, 51(6):2224–2227, 2005.

[42] A. B. Lewko, M. Lewko, and B. Waters. How to leak on key updates. In L. Fortnow and S. P. Vadhan, editors, *43rd ACM STOC*, pages 725–734. ACM Press, June 2011.

[43] T. Malkin, I. Teranishi, Y. Vahlis, and M. Yung. Signatures resilient to continual leakage on memory and computation. In Y. Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 89–106. Springer, Mar. 2011.

[44] S. Micali and L. Reyzin. Physically observable cryptography (extended abstract). In M. Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 278–296. Springer, Feb. 2004.

[45] M. Naor and G. Segev. Public-key cryptosystems resilient to key leakage. In S. Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 18–35. Springer, Aug. 2009.

[46] A. Sahai and B. Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In D. B. Shmoys, editor, *46th ACM STOC*, pages 475–484. ACM Press, May / June 2014.

[47] B. Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In S. Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 619–636. Springer, Aug. 2009.

[48] B. Waters. CS 395T Special Topic: Obfuscation in Cryptography. 2014. http://www.cs.utexas.edu/~bwaters/classes/CS395T-Fall-14/outline.html.

[49] B. Waters. How to use indistinguishability obfuscation. In *Visions of Cryptography*, 2014. Talk slides available at http://www.cs.utexas.edu/~bwaters/presentations/files/how-to-use-IO.ppt.

[50] D. Wichs. Cryptographic resilience to continual information leakage. PhD Thesis, 2011. http://www.ccs.neu.edu/home/wichs/thesis.pdf.