Leakage-Resilient Public-Key Encryption from Obfuscation

Dana Dachman-Soled[1⋆], S. Dov Gordon[2⋆⋆], Feng-Hao Liu[3⋆⋆⋆], Adam O'Neill[4], and Hong-Sheng Zhou[5]

[1] University of Maryland, danadach@ece.umd.edu
[2] George Mason University, crypto@dovgordon.com
[3] Florida Atlantic University, fenghao.liu@fau.edu
[4] Georgetown University, adam@cs.georgetown.edu
[5] Virginia Commonwealth University, hszhou@vcu.edu

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). In particular, our compiler adapts and optimizes recent techniques of Sahai and Waters (STOC 2014) that make any encryption scheme sender-deniable. We then show that prior continual leakage resilient schemes can be upgraded to security against consecutive continual leakage without introducing new assumptions.

⋆ 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.

⋆⋆ This work was done when the author was a research scientist at Applied Communication Sciences.

⋆⋆⋆ This work was done when the author was a postdoc at the University of Maryland.

In the bounded leakage model, we develop an entirely new approach to constructing leakage-resilient encryption from obfuscation directly, based upon the public-key encryption scheme from iO and punctured pseudorandom functions due to Sahai and Waters (STOC 2014). In particular, 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.

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. [1,22,19,4]). 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. [22,4]). Hazay et al. [17] 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*[6] — allows the adversary to request *unbounded* leakage. This model was introduced by Brakerski et al. [5] and Dodis et al. [11], 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

[6] 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" [21].

of [5,11] 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. [20], with their techniques being considerably refined and generalized by Dodis et al. [12]. 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. [24]) 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" [6] for encryption schemes. In particular, our compiler uses program obfuscation — either indistinguishability obfuscation (iO) [2,14] or the public-coin differing-inputs obfuscation [18][7] — and adapts and extends techniques recently developed by Sahai and Waters [23] to achieve sender-deniable encryption. This demonstrates the applicability of the techniques of [23] to other seemingly unrelated contexts.[8] We then show that the existing CLR encryption scheme of Brakerski et al. [5] 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. [11]), 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 pseudo-random functions [23] 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. Extending these constructions to continual leakage-resilience (without introducing 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),

[7] 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. [15] do not apply to this setting.

[8] We note that the techniques of [23] have been shown useful in adaptively secure two-party and multiparty computation [16,7,9] and "only computation leaks" (OCL) circuits without trusted hardware [10]. We note that this work precedes the work of [9].

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. [5] 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 $1/258-o(1)$ [20]. Our result nearly matches the trivial upper-bound of $1/2-o(1)$.[9] 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. As we have mentioned above, Hazay et al. [17] constructed 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) [5,11] 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:
(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) [2,14] or public-coin differing-inputs obfuscation [18]. In both cases, if we start with an underlying scheme that is consecutive

[9] 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.

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 [23] 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 [5] and show that a suitable modification of it achieves 2CLR[10] with optimal $1/4 - o(1)$ leakage rate[11], under the same assumption used by [5] 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 [13,3] that generalizes the result of [5], 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 [5].

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. [11], 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 [11] 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 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) [23], 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[12], 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.

[10] Note that [5] 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.

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

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

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.

2 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 [23]. 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 2.2). Finally, we show two instantiations of the explainable update transformation: one based on indistinguishability obfuscation, and the other on differing-inputs obfuscation (Section 2.3). 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.

2.1 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, leakage resilient one-way relations [11]. We only present the PKE version for simplicity and concreteness. 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 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.*

2.2 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 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 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 1. *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* 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 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 pk^*, 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. $\boldsymbol{p} = (\mathsf{pk}, \mathsf{sk}_0, \widehat{\mathsf{sk}}_1, \ldots, \widehat{\mathsf{sk}}_{i-1})$. By an average argument, there exists a fixed

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

such that $\mathcal{D}$ can distinguish $H^{(i)}$ from $H^{(i.5)}$ conditioned on $\boldsymbol{p}^*$ 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 $\boldsymbol{p}^*$ 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 $\mathcal{D}(H^{(i)}|_{\boldsymbol{p}^*})$. On the other hand, suppose it was sampled according to $\mathcal{P}_{\mathsf{explain}}$ (as e), then $\mathcal{D}'$ will output according to $\mathcal{D}(H^{i.5}|_{\boldsymbol{p}^*})$. 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.

2.3 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 [18] (rather than the weaker indistinguishability obfuscation (iO) [2,14]), 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[13], mapping $2L_{\mathsf{sk}} + 2\kappa$ bits to κ bits. Let F_1 and F_2 be families of puncturable pseudo-random 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 $\mathsf{sk}_2 = \mathsf{PKE.Update}(\mathsf{pk}, \mathsf{sk}_1; x)$.

Fig. 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 $e = (u_1, u_2)$.

Fig. 2. Program Explain

Then we can establish the following theorem.

Theorem 2. *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 with the additional properties stated above, and $\mathcal{H}$ is a family of public-coin collision resistant hash function with the extraction property as above. Then the transformation* TransformGen *defined above is a secure explainable update transformation for* PKE *as defined in Definition 2.*

[13] 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.

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.

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 "Else 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 $\mathsf{sk}_2 = \mathsf{PKE.Update}(\mathsf{pk}^*, \mathsf{sk}_1; x)$.

Fig. 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 $\mathsf{sk}_2 = \mathsf{PKE.Update}(\mathsf{pk}^*, \mathsf{sk}_1; x)$.

Fig. 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 $e = (u_1, u_2)$.

Fig. 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.

3 2CLR from "Leakage Resilient Subspaces"

We show that the PKE scheme of Brakerski et al. [5] (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 [8] that any CLR (resp. 2CLR) PKE scheme implies a CLR "one-way relation" (OWR) [11]; to the best of our knowledge, such an implication was not previously known. Therefore, by the results of Dodis et al. [11], this translates all our results about PKE to the signature setting as well. In the full version [8] of the paper, we show that the approach of Dodis et al. [11] for constructing CLR OWRs can be extended to 2CLR one-way relations, but we achieve weaker parameters this way.

Recall that in the work [5], 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*[14] *. Then we have:*

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

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

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

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

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

[14] Note: Rk denotes rank. Here we use n as the dimension (different from [5] who used m) to avoid overloading notation.

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 [5]: 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 [5] 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) [13,3] 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*

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

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 $\Pr[\,(D, D', Z) \in \mathsf{BAD}\,] \leq \epsilon'$. Then for any set $\mathcal{S}$ and function $f\colon \mathcal{R} \times \mathcal{Z} \to \mathcal{S}$ it holds that

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

Proof. The proof is an extension of the proof of the Crooked LHL given in [3]. First, using Cauchy-Schwarz and Jensen's inequality we have

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

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.

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

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

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

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 $\Pr\left[\, f(H_k(D),z)=s\,\right] = \sum_d \Pr\left[\, D=d\,\right]\delta_{f(H_k(d),z),s}$ where $\delta_{a,b}$ is 1 if $a=b$ and 0 otherwise, and thus

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

So we have

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

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 $\Pr[(z,d,d') \in \mathsf{BAD}] \le \epsilon'$.

By a similar calculation, we have:

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

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 [5], 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 3. *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 $\Big(A, X, L(A, X \cdot T, X \cdot T'), X \cdot T'\Big)$ from $\Big(A, X, L(A, U, X \cdot T'), X \cdot T'\Big)$ 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

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

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 [5]. 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 [5], 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 [5], 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.

4 Bounded leakage-resilient encryption schemes from Obfuscation

We show that by modifying the Sahai-Waters (SW) public key encryption scheme [23] 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 the full version [8] of the paper, we will show how to further modify the construction to achieve optimal leakage rate.

On 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.

The informal theorem statement is below. We present the formal theorem and proof in the full version.

Theorem 4 (Informal.). *Under appropriate assumptions, $\mathcal{E}$ is L-leakage resilient against one-time key leakage where $L = \rho - 2\log(1/\epsilon) - L_{\mathsf{msg}}$.*

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

Key Generation: $(\mathsf{pk}, \mathsf{sk}_0) \leftarrow \mathcal{E}.\mathsf{Gen}(1^\kappa)$
Compute $k \leftarrow \mathsf{PRF}.\mathsf{Gen}(1^\kappa)$, 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 $\hat{m} = \perp$.

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

Internal (hardcoded) state: k.

On input: r

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

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

Fig. 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.

References

1. 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.
2. 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.
3. 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.
4. 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.
5. 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.

6. 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.
7. 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.
8. 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.
9. 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.
10. 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.
11. 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.
12. 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.
13. 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.
14. 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.
15. 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.
16. 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.
17. 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.
18. 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.
19. 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.
20. 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.
21. 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.
22. 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.
23. 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.

24. 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.