# Cryptography Against Continuous Memory Attacks

Yevgeniy Dodis
*New York University*
*dodis@cs.nyu.edu*

Kristiyan Haralambiev
*New York University*
*kkh@cs.nyu.edu*

Adriana López-Alt
*New York University*
*lopez@cs.nyu.edu*

Daniel Wichs
*New York University*
*wichs@cs.nyu.edu*

***Abstract*—We say that a cryptographic scheme is *Continuous Leakage-Resilient* (CLR), if it allows users to refresh their secret keys, using only fresh local randomness, such that:**

- **The scheme remains functional after any number of key refreshes, although the *public key never changes*. Thus, the "outside world" is neither affected by these key refreshes, nor needs to know about their frequency.**
- **The scheme remains secure even if the adversary can *continuously* leak arbitrary information about the current secret-key, as long as the *amount* of leaked information is *bounded in between any two successive key refreshes*. There is *no* bound on the *total* amount of information that can be leaked during the lifetime of the system.**

**In this work, we construct a variety of *practical* CLR schemes, including CLR one-way relations, CLR signatures, CLR identification schemes, and CLR authenticated key agreement protocols. For each of the above, we give general constructions, and then show how to instantiate them *efficiently* using a well established assumption on bilinear groups, called the *K-Linear* assumption (for any constant K greater than or equal to 1). Our constructions are highly modular, and we develop many interesting techniques and building-blocks along the way, including: leakage-indistinguishable re-randomizable relations, homomorphic NIZKs, and leakage-of-ciphertext non-malleable encryption schemes.**

## I. Introduction

MAIN TECHNICAL CONTRIBUTION. As our main technical contribution, we design an efficiently verifiable relation $R(pk, sk)$, on public keys $pk$ (the *statement*) and secret keys $sk$ (the *witness*), which is *continuous leakage resilient* in the following sense. The user can periodically refresh the secret-key $sk$, using only some additional fresh local randomness, without affecting the public-key $pk$. The adversary first sees $pk$ and then can attack the system for *arbitrarily* many periods, where, in each period, she can adaptively learn up to $\ell$ bits of *arbitrary* information about the current witness $sk$, for some parameter $\ell < |sk|$, called the *leakage bound*. The secret-key $sk$ is then refreshed for the next period. Nevertheless, after attacking the system for any polynomial number of periods, the attacker still cannot produce a valid witness $sk^*$ such that $R(pk, sk^*) = 1$. Notice that, while there is a necessary bound on the *amount* of leakage in *each period* (learning the secret-key of any single period *in full* necessarily breaks the system), no restriction is placed on its *type*, and the *overall* amount of leakage during the attack is *unbounded*.

APPLICATIONS. We call such a relation $R$ an *$\ell$-continuous-leakage-resilient (CLR) one-way relation (OWR)* ($\ell$-CLR-OWR; see Definition 2). More generally, we say that a cryptographic primitive is $\ell$-CLR, if its secret key can be similarly refreshed using only fresh local randomness, and the usual security of this primitive holds even in the presence of a "continuous key leakage attack", as described above. Using CLR-OWRs, we design more advanced CLR primitives, including CLR signatures, CLR identification (ID) schemes, and CLR authenticated key agreement (AKA) protocols. For each of the above, we give *general* constructions, and then show how to instantiate them *efficiently* using the same number-theoretic assumption we used to derive our efficient CLR-OWR. This well established assumption is called *K-Linear* [9], [23], [33], and is believed to hold in appropriate bilinear groups for constants $K \geq 1$. In particular, the 1-linear assumption is equivalent to the DDH assumption, which only holds in some *asymmetric* bilinear groups, while 2-linear is regularly assumed to hold in many symmetric and asymmetric bilinear groups. The assumption gets *weaker* as $K$ grows. The *amount* of leakage that the schemes can tolerate in each period, measured by the ratio $\alpha = \frac{\ell}{|sk|}$ (i.e. the *fraction* of the secret-key that can leak), is given by $\alpha \approx 1/2$ under DDH, $\alpha \approx 1/3$ under 2-linear, and generally $\alpha \approx \frac{1}{K+1}$ under $K$-linear.

BIGGER PICTURE. Recently, works on various kinds of *leakage-resilient (LR) cryptography* have received a lot of attention. The goal is to design cryptographic primitives which provably resist large classes side-channel attacks, where the attacker may obtain additional unanticipated information about the secret key $sk$, not captured by the traditional definitions. Very roughly, state-of-the-art LR schemes fall into two categories. Schemes secure against "(one-time) memory attacks" [1], [29], [3], [27], [2], do not restrict the *type* of leakage that the adversary can see, but have some a-priori upper bound $\ell$ on the *overall amount* of leakage. They do not offer a method for refreshing the secret-key. In contrast, current "continuous leakage" models [11], [24], [28], [16], [30], [17] use secret-key evolution, allowing them to only restrict the amount of key leakage *per period* (as opposed to overall), but pay the price by placing additional non-trivial *restrictions* on the type of leakage. For example, a popular assumption [28], [16], [30], [17] is the "only computation leaks information" (OCLI) model of Micali and Reyzin [28], where any part of the secret-key that isn't accessed by the user during a given period *cannot* leak.

A major open question in the area, articulated, for example, by Goldwasser during her invited talk at Eurocrypt'09 [19], is to get "the best of both worlds": allowing for continuous (overall unbounded) leakage, without additionally restricting the type of leakage. In this work, we show how to achieve this for several cryptographic primitives.

RELATED PRIOR WORK. We only give a brief overview of the most relevant prior work. See the full version for a more extensive review of prior work in leakage-resilient cryptography.

The model of (one-time) memory attacks was introduced by Akavia, Goldwasser and Vaikuntanathan [1] in the context of public-key encryption (PKE), also studied in [29], [2], [14]. Signature schemes in this model were constructed in [3], [27], [14]. The only work to consider continuous leakage in the context of memory attacks is [3], which showed a general method of refreshing secret keys using an external "master refreshing key", which must be stored securely and cannot leak (essentially, the master key is used to sign a completely fresh key pair used for that period only). In contrast, our work does not assume any secure storage, and keys can be refreshed using only fresh local randomness.

Various prior works considered continuous leakage in models that restrict the *type*, as well as *amount*, of information that the adversary can learn. For example [24] considers a model where the wires of a circuit can (only) be leaked *individually*, while [17] allows the adversary to (only) observe low-complexity (e.g. $AC^0$) or noisy functions of the wires. Micali and Reyzin [28] introduced the *"only computation leaks information"* model, where each invocation of a cryptographic primitive can leak an arbitrary function of *only* the bits accessed during that invocation. Several primitives have been constructed in this setting including stream ciphers [16], [30] and signatures [17]. More recently, [26], [20] construct a general compiler that can secure *all primitives* in this setting assuming the use of some limited leak-free components. We note that all of these constructions are easily broken in the model of memory attacks, when no restrictions are placed on the type of leakage allowed to the adversary.

CONCURRENT WORK. Subsequent to our work, Brakerski et al. [10] solve the main remaining open problem, by constructing CLR public-key *encryption* (and IBE) schemes under the $K$-Linear assumption. Their techniques also yield an alternate construction of signature schemes under the $K$-Linear assumption (same as our work), with improved relative-leakage $\alpha \approx 1/K$. Prior to our work, the preliminary results of Brakerski et al. showed the feasibility of CLR primitives under a strong and non-falsifiable assumption on hash functions.

STRUCTURE OF OUR PAPER. Our main technical contribution lies in building a CLR-OWR (Sections III-V). We do so in a modular manner. First, in Section III, we show how to reduce the problem of analyzing *continuous leakage* for OWR to that of only analyzing *one-time leakage*, albeit in a more complicated primitive, which we call *leakage-indistinguishable re-randomizable relations* (LIRR). Then, in Section IV we show how to build LIRR generically from encryption schemes and NIZKs (having some well-specified additional properties). Finally, in Section V we instantiate the required components efficiently using the $K$-Linear assumption in bilinear groups to get a CLR-OWR. In Section VI, we then show how to leverage a CLR-OWR to get other more interesting CLR primitives: signatures, ID schemes, and AKA protocols. Lastly, in Section VII we mention several extensions of our results to even *stronger* models/notions of CLR security. Most importantly, we address the issue of the adversary (also) leaking on the random-coins used to refresh the secret-keys and to generate signatures.

## II. CONTINUOUS LEAKAGE & ONE-WAY RELATIONS

A one-way relation (OWR) generalizes the standard notion of a one-way function (OWF). It consists of a key-generation algorithm $(pk, sk) \leftarrow \texttt{KeyGen}()$, and a verification algorithm $\texttt{Ver}(pk, sk)$, which decides whether a secret-key $sk$ *is valid for* the public-key $pk$. The security of a OWR says that, if an adversary only sees $pk$ she should be unable to come up with any $sk^*$ for which $\texttt{Ver}(pk, sk^*) = 1$.

Of course, we can always include all of the randomness of the KeyGen algorithm in $sk$, so that $pk = \texttt{KeyGen}(sk)$ *is* a OWF. However, when defining leakage-resilient one-wayness (which we do next), this equivalence of OWR and OWF might no longer hold – by putting more information into the secret key we would also have to give the adversary more information during key-leakage attacks. Therefore, we consider OWRs, rather than OWFs, as the basic cryptographic primitive for leakage-resilience.

MODELING LEAKAGE ATTACKS. We model leakage attacks (also called memory attacks in prior work) against a secret key $sk$, by giving the adversary access to a *leakage oracle* $\mathcal{O}_{sk}^{\lambda}(\cdot)$, defined as follows:

**Definition 1** (Leakage Oracle). *A leakage oracle $\mathcal{O}_{sk}^{\lambda}(\cdot)$ is parameterized by a secret key sk and a security parameter $\lambda$. A query to the oracle consists of a description of a 1-bit probabilistic* leakage function $h : \{0,1\}^* \to \{0,1\}$. *The oracle runs $h(sk)$ for $\mathsf{poly}(\lambda)$ steps: if the computation completes it outputs $h(sk)$, else it outputs* $0$.

In our definitions of leakage-resilient primitives, the adversary can adaptively access $\mathcal{O}_{sk}^{\lambda}(\cdot)$ to learn information about the secret key $sk$ during an attack. Each query provides 1 bit of information. Our definitions will therefore place some upper bounds $\ell$ on the number of queries that the adversary can make during various stages of the attack.

LEAKAGE RESILIENT OWR. We can define a leakage-resilient OWR ($\ell$-LR-OWR) by allowing the adversary to make up to $\ell$ queries to the oracle $\mathcal{O}^{\lambda}_{sk}(\cdot)$, after seing $pk$ and before outputting $sk^*$. It was shown (e.g. in [4]) that any second-preimage resistant[1] (SPR) function $F$ automatically gives an $\ell$-LR-OWR, where $\ell$ is roughly the number of bits by which $F$ shrinks its input. This follows from a simple entropy argument: if an adversary sees $y = F(x)$ and $\ell$ bits of leakage on $x$, then $x$ still has entropy. Therefore, if the adversary outputs $x'$ such that $F(x') = y$, then, with high probability, $x' \neq x$ and hence the adversary breaks SPR.

CONTINUOUS LEAKAGE RESILIENCE. A *continuous-leakage-resilient (CLR) one-way relation (OWR)* consists of the algorithms KeyGen and Ver as before, but also includes a re-randomization algorithm $sk' \leftarrow \mathtt{ReRand}_{pk}(sk)$, which can be used to update the secret key (we will omit the subscript $pk$, when clear from context). On a high level, a OWR is continuous-leakage-resilient if an adversary can observe $\ell$ bits of leakage on each of arbitrarily many re-randomized secret keys, and still be unable to produce a valid secret key herself.

**Definition 2** (CLR-OWR). *We say that a scheme* $(\mathtt{KeyGen}, \mathtt{ReRand}, \mathtt{Ver})$ *is an* $\ell$-continuous-leakage-resilient ($\ell$-CLR) one-way relation *if it satisfies the following correctness and security properties.*

- Correctness: *For any polynomial $q = q(\lambda)$, if we sample* $(pk, sk_1) \leftarrow \mathtt{KeyGen}(1^\lambda), \{sk_{i+1} \leftarrow \mathtt{ReRand}(sk_i)\}_{i=1}^{q-1}$ *then, with overwhelming probability (w.o.p.)* $\mathtt{Ver}(pk, sk) = \mathtt{Ver}(pk, sk_1) = \ldots = \mathtt{Ver}(pk, sk_q) = 1$.
- Security: *For any PPT adversary $\mathcal{A}$, we have* $\Pr[\mathcal{A}$ *wins* $] \leq \mathsf{negl}(\lambda)$ *in the following game:*
- *Challenger chooses* $(pk, sk) \leftarrow \mathtt{KeyGen}(1^\lambda)$.
- *$\mathcal{A}$ gets pk and runs for arbitrarily many* leakage rounds. *In each round $\mathcal{A}$ makes up to $\ell$ calls to $\mathcal{O}^{\lambda}_{sk}(\cdot)$, leaking on the current secret key sk. At the end of the round, the challenger samples $sk' \leftarrow \mathtt{ReRand}(sk)$ and updates $sk := sk'$.*
- $\mathcal{A}$ wins *if it outputs $sk^*$ such that* $\mathtt{Ver}(pk, sk^*) = 1$.

WHY PRIOR LR TECHNIQUES FAIL FOR CLR. Prior works on one-time memory-leakage attacks crucially relied on an entropy argument: given the leakage and the public-key, the secret-key $sk$ still had some entropy left. For example, this was the main step of our argument for the leakage-resilience of SPR functions. However, it is unclear how to translate this type of argument to the setting of *continuous leakage-resilience*, where the total amount of information seen by the adversary is unbounded. We show a novel strategy for reasoning about continuous leakage in the next section.

[1]A function $F$ is SPR if, given a random $x$, it's hard to find any $x' \neq x$ such that $F(x) = F(x')$.

## III. CONTINUOUS LEAKAGE FROM ONE-TIME LEAKAGE

In this section, we define a new primitive, called a *leakage-indistinguishable re-randomizable relation (LIRR)*, and show that it can be used to construct a CLR-OWR. Although the definition of the new primitive is fairly complex with several security requirements, its main advantage is that it reduces the problem of *continuous*-leakage resilience for OWR to a simpler *one-time* leakage-resilience property.

A 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 $\ell$ 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 (Setup, SampG, SampB, ReRand, Ver, isGood) with the following syntax:

- $(pk, \mathsf{sam}_G, \mathsf{sam}_B, dk) \leftarrow \mathtt{Setup}(1^\lambda)$ : Outputs a *public-key pk*, a "good" sampling-key $\mathsf{sam}_G$, a "bad" sampling key $\mathsf{sam}_B$, and a *distinguishing-trapdoor dk*.
- $sk_G \leftarrow \mathtt{SampG}_{pk}(\mathsf{sam}_G)$, $sk_B \leftarrow \mathtt{SampB}_{pk}(\mathsf{sam}_B)$: These algorithms sample good/bad secret-keys using good/bad sampling keys respectively. We omit the subscript $pk$ when clear from context.
- $b = \mathtt{isGood}(pk, sk, dk)$: Uses $dk$ to distinguish good secret-keys $sk$ from bad ones.
- $sk' \leftarrow \mathtt{ReRand}_{pk}(sk)$, $b = \mathtt{Ver}(pk, sk)$. These have the same syntax as in the definition of CLR-OWR.

**Definition 3** (LIRR). *We say that the scheme* (Setup, SampG, SampB, ReRand, Ver, isGood) *is an* $\ell$-leakage-indistinguishable re-randomizable relation *($\ell$-LIRR) if it satisfies the following properties:*

- Correctness: *If* $(pk, \mathsf{sam}_G, \mathsf{sam}_B, dk) \leftarrow \mathtt{Setup}(1^\lambda)$, $sk_G \leftarrow \mathtt{SampG}(\mathsf{sam}_G)$, $sk_B \leftarrow \mathtt{SampB}(\mathsf{sam}_B)$ *then w.o.p.*
$$\mathtt{Ver}(pk, sk_G) = 1,\ \mathtt{isGood}(pk, sk_G, dk) = 1$$
$$\mathtt{Ver}(pk, sk_B) = 1,\ \mathtt{isGood}(pk, sk_B, dk) = 0$$
- Re-Randomization: *We require that* $(pk, \mathsf{sam}_G, sk_0, sk_1) \overset{c}{\approx} (pk, \mathsf{sam}_G, sk_0, sk'_1)$*, where:*
$(pk, \mathsf{sam}_G, \mathsf{sam}_B, dk) \leftarrow \mathtt{Setup}(1^\lambda)$,
$sk_0 \leftarrow \mathtt{SampG}(\mathsf{sam}_G)$, $sk_1 \leftarrow \mathtt{SampG}(\mathsf{sam}_G)$,
$sk'_1 \leftarrow \mathtt{ReRand}(sk_0)$
- Hardness of Bad Keys: *Given* $\mathsf{sam}_G$*, it's hard to produce a valid "bad key". Formally, for any PPT adversary $\mathcal{A}$:*
$$\Pr\left[\begin{array}{c}\mathtt{Ver}(pk, sk^*) = 1\\ \mathtt{isGood}(pk, sk^*, dk) = 0\end{array}\right] \leq \mathsf{negl}(\lambda)$$

- $\texttt{KeyGen}(1^\lambda)$: Sample $(pk, \mathsf{sam}_G, \cdot, \cdot) \leftarrow \texttt{Setup}(1^\lambda)$, $sk \leftarrow \texttt{SampG}(\mathsf{sam}_G)$ and output $(pk, sk)$.
- $\texttt{ReRand}, \texttt{Ver}$: Same as for LIRR.

Figure 1. Constructing CLR-OWR from a LIRR

*where probability is over*

$$(pk, \mathsf{sam}_G, \cdot, dk) \leftarrow \texttt{Setup}(1^\lambda), sk^* \leftarrow \mathcal{A}(pk, \mathsf{sam}_G)$$

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

$$\Pr[\texttt{isGood}(pk, sk^*, dk) = 1] \leq \mathsf{negl}(\lambda)$$

*where the probability is over*

$$(pk, \cdot, \mathsf{sam}_B, dk) \leftarrow \texttt{Setup}(1^\lambda), sk^* \leftarrow \mathcal{A}(pk, \mathsf{sam}_B).$$

- $\ell$-Leakage-Indistinguishability: *Given both sampling keys* $\mathsf{sam}_G, \mathsf{sam}_B$*, and* $\ell$ *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* $\left|\Pr[\mathcal{A} \text{ wins }] - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda)$ *in the following game:*
- *The challenger chooses* $(pk, \mathsf{sam}_G, \mathsf{sam}_B, dk) \leftarrow \texttt{Setup}(1^\lambda)$ *and gives* $pk, \mathsf{sam}_G, \mathsf{sam}_B$ *to* $\mathcal{A}$*.*
- *The challenger chooses* $b \leftarrow \{0, 1\}$*. If* $b = 1$ *then set* $sk \leftarrow \texttt{SampG}(\mathsf{sam}_G)$*, else* $sk \leftarrow \texttt{SampB}(\mathsf{sam}_B)$*.*
- $\mathcal{A}$ *can make up to* $\ell$ *queries to* $\mathcal{O}^\lambda_{sk}(\cdot)$*. At the end, it outputs* $sk^*$ *and wins if* $\texttt{isGood}(pk, sk^*, dk) = b$*.*

An $\ell$-LIRR can be used to construct an $\ell$-CLR-OWR, where the $\texttt{ReRand}, \texttt{Ver}$ algorithms are kept the same, while $\texttt{KeyGen}$ samples $pk$ and a "good" secret key $sk_G$ (see Figure 1). Note that the CLR-OWR completely ignores the the bad sampling algorithm $\texttt{SampB}$, the "bad" sampling key $\mathsf{sam}_B$, the distinguishing algorithm $\texttt{isGood}$, and the distinguishing key $dk$ of the LIRR. These are only used in the argument of security. Moreover, the "good" sampling key $\mathsf{sam}_G$ is only used as an intermediate step during key-generation to sample the secret-key $sk$, but is never explicitly stored afterwards.

**Theorem 4.** *Given any $\ell$-LIRR scheme, the construction in Figure 1 is a secure $\ell$-CLR-OWR.*

We argue that the above construction is secure. Assume an adversary attacks the construction and, after several leakage-rounds, produces a valid secret key $sk^*$. Since the adversary does not see any information related to the bad sampling key $\mathsf{sam}_B$, we can use the *hardness of bad keys* property to argue that $sk^*$ must be a "good" key. However, we then argue that the adversary cannot notice if we start switching good keys to bad keys in the leakage rounds. More precisely, we define several hybrid games, where each game differs from the previous by replacing a good key with a bad key in one additional leakage round. We argue that, by the *$\ell$-leakage-indistinguishability* property, the probability that the adversary produces a good key $sk^*$ as her forgery does not change between any two hybrids. Notice that this argument allows us to only analyze leakage in a single round at a time, and thus avoids the main difficulty of analyzing continuous leakage. In the last of the hybrids, the adversary only sees "bad keys", yet still manages to produce a good key $sk^*$ as her forgery. But this contradicts the *hardness of good keys* property, and proves the security of the scheme.

## IV. Constructing LIRR

We now instantiate an $\ell$-LIRR using two public-key encryption schemes and a NIZK argument system.

Review of NIZK. We remind the reader of the basic syntax of *non-interactive zero-knowledge (NIZK) arguments* [7], [31], [8]. 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 system for $R$, consists of four PPT algorithms $(\texttt{Setup}, \texttt{Prov}, \texttt{Ver}, \texttt{Sim})$ with syntax:

- $(\mathrm{CRS}, \mathrm{TK}) \leftarrow \texttt{Setup}(1^\lambda)$: Creates a common reference string (CRS) and a trapdoor key to the CRS.
- $\pi \leftarrow \texttt{Prov}_{\mathrm{CRS}}(y, x)$: Creates an argument $\pi$ for the statement $y \in L_R$, using the witness $x$.
- $\pi \leftarrow \texttt{Sim}_{\mathrm{CRS}}(y, \mathrm{TK})$: Creates a simulated argument for a statement $y$ using the trapdoor TK.
- $0/1 \leftarrow \texttt{Ver}_{\mathrm{CRS}}(y, \pi)$: Verifies whether or not the argument $\pi$ is correct.

For the sake of clarity, we write $\texttt{Prov}$ and $\texttt{Ver}$ without the CRS in the subscript, when clear from context. The definition of security for NIZK arguments is given in the full version [13]. For this work, the default notion of NIZKs includes composability (real/simulated NIZKs cannot be distinguished even given the trapdoor key TK).

Overview of Construction. We first start by describing the high level idea and the syntax of the construction. Let $\mathcal{E}_1 = (\texttt{KeyGen}^1, \texttt{Enc}^1, \texttt{Dec}^1)$, $\mathcal{E}_2 = (\texttt{KeyGen}^2, \texttt{Enc}^2, \texttt{Dec}^2)$ be two public-key encryption schemes, with perfect correctness. We define the *plaintext equality* relation for the schemes $\mathcal{E}_1, \mathcal{E}_2$ by:

$$R_{eq} \overset{\text{def}}{=} \Big\{ (y, x) \mid y = (pk_1, pk_2, c_1, c_2),\ x = (m, r_1, r_2) \\ \text{s.t. } c_1 = \texttt{Enc}^1_{pk_1}(m; r_1),\ c_2 = \texttt{Enc}^2_{pk_2}(m; r_2) \Big\}.$$

The corresponding language $L_{eq}$, is that of ciphertext pairs that encrypt the same plaintext. Let $\Pi = (\texttt{Setup}^\Pi, \texttt{Prov}^\Pi, \texttt{Ver}^\Pi, \texttt{Sim}^\Pi)$ be a NIZK argument system for $R_{eq}$. We will often omit the public-keys $pk_1, pk_2$ from the descriptions of statements $y \in L_{eq}$, when clear from context.

We will assume that the schemes $\mathcal{E}_1, \mathcal{E}_2, \Pi$ can share some common *system parameters* $\mathsf{prms} \leftarrow \texttt{ParamGen}(1^\lambda)$ (e.g. the description of some group) which are implicitly used as inputs by all of the algorithms of each of the schemes. The parameters define a common message-space $\mathcal{M}$ for the schemes $\mathcal{E}_1, \mathcal{E}_2$. The basic syntax of our construction of LIRR, except for the re-randomization algorithm, is shown in Figure 2. The main idea is to encrypt a random message

- $\texttt{Setup}(1^\lambda)$: Output $pk = (\mathsf{prms}, \textsc{crs}, pk_1, pk_2, c_1)$, $\mathsf{sam}_G = (m, r_1)$, $\mathsf{sam}_B = \textsc{tk}$, $dk = (sk_1, sk_2)$ where:
  $\mathsf{prms} \leftarrow \texttt{ParamGen}(1^\lambda)$, $(pk_1, sk_1) \leftarrow \texttt{KeyGen}^1(\mathsf{prms})$, $(pk_2, sk_2) \leftarrow \texttt{KeyGen}^2(\mathsf{prms})$, $m \leftarrow \mathcal{M}$, $c_1 \leftarrow \texttt{Enc}^1_{pk_1}(m; r_1)$, $(\textsc{crs}, \textsc{tk}) \leftarrow \texttt{Setup}^\Pi(\mathsf{prms})$
- $\texttt{SampG}(\mathsf{sam}_G)$: Output $sk_G = (c_2, \pi)$ where:
  $c_2 \leftarrow \texttt{Enc}^2_{pk_2}(m; r_2), \pi \leftarrow \texttt{Prov}^\Pi((c_1, c_2), (m, r_1, r_2))$
- $\texttt{SampB}(\mathsf{sam}_B)$: Output $sk_B = (c_2, \pi)$ where:
  $c_2 \leftarrow \texttt{Enc}^2_{pk_2}(1)$, $\pi \leftarrow \texttt{Sim}^\Pi((c_1, c_2), \textsc{tk})$
- $\texttt{Ver}(pk, sk)$: Parse $sk = (c_2, \pi)$, output $\texttt{Ver}^\Pi((c_1, c_2), \pi)$.
- $\texttt{isGood}(pk, sk, dk)$: Parse $sk = (c_2, \pi), dk = (sk_1, sk_2)$. Output 1 iff $\texttt{Dec}^1_{sk_1}(c_1) = \texttt{Dec}^2_{sk_2}(c_2)$.

Figure 2. Construction of LIRR.

$m$ using the scheme $\mathcal{E}_1$, and put the ciphertext $c_1$ in the public-key. The secret key consists of a ciphertext/proof pair $(c_2, \pi)$. In a good secret-key, $c_2$ is a new random encryption of $m$ under $\mathcal{E}_2$, and $\pi$ is a proof of plaintext-equality. In a bad secret-key, $c_2$ is just an encryption of the message 1, and $\pi$ is a simulated proof. The verification procedure just checks the proof $\pi$ against the statement $(c_1, c_2)$.

It is easy to see that the scheme satisfies the correctness property. The hardness of *bad keys*, follows directly from the soundness of the NIZK argument system. The hardness of *good keys*, on the other hand, follows from the security of the encryption scheme $\mathcal{E}_1$. In fact, we only need *one-way security* (see the full version [13] for a formal definition), which is weaker than semantic-security and only requires that an encryption of a random message is hard to invert.

**Lemma 5.** *Assume that $\mathcal{E}_1$ is a one-way secure and $\Pi$ is a sound NIZK. Then the construction in Figure 2 satisfies the* correctness, hardness of good keys *and* hardness of bad keys *properties of the LIRR definition (Definition 3).*

We are left to show (1) how to re-randomize secret keys, and (2) that the leakage-indistinguishability property holds. We do so in the next two sections, by requiring additional properties from the building-blocks $\mathcal{E}_1, \mathcal{E}_2, \Pi$.

## *A. Re-randomization: Homomorphic Encryptions & NIZKs*

We now show how to perfectly re-randomize the secret keys of our LIRR construction. Recall that a secret-key consists of a pair $(c_2, \pi)$ where $\pi$ is a proof of plaintext-equality for the statement $(c_1, c_2)$. To re-randomize the key, we need to first re-randomize the ciphertext $c_2$ into a new ciphertext $c_2^*$ with the same plaintext. We then need to update the proof $\pi$ to look like a *fresh new* proof of a *new* statement $(c_1, c_2^*)$, for which we do not know the witness! We show that this is indeed possible if the encryption schemes $\mathcal{E}_1, \mathcal{E}_2$ and the argument-system $\Pi$ are all homomorphic over some appropriate group. In particular, we define a new notion of *homomorphic NIZKs*, which is influenced by the notions of re-randomizable and malleable NIZKs from [6].

**Definition 6** (Homomorphic Encryption)**.** *We say that an encryption scheme* $(\texttt{KeyGen}, \texttt{Enc}, \texttt{Dec})$ *is* homomorphic *if the system-parameters of the scheme define groups $(\mathcal{M}, \cdot), (\mathcal{R}, +), (\mathcal{C}, \cdot)$ for the message-space, randomness-space, and ciphertext-space respectively, such that, for any $m, m' \in \mathcal{M}$ any $r, r' \in \mathcal{R}$:*

$$c = \texttt{Enc}_{pk}(m; r),\ c' = \texttt{Enc}_{pk}(m'; r') \quad \Rightarrow \quad c \cdot c' = \texttt{Enc}_{pk}(m \cdot m'; r + r')$$

It is easy to see that for any homomorphic encryption scheme and any public-key $pk$, we have $\texttt{Enc}_{pk}(1_\mathcal{M}; 0_\mathcal{R}) = 1_\mathcal{C}$ (i.e. encryption of the identity -message under identity-randomness gives an identity-ciphertext).

**Definition 7** (Homomorphic Relation)**.** *We say that a relation $R \subseteq \mathcal{Y} \times \mathcal{X}$ is* homomorphic *if $(\mathcal{Y}, \cdot), (\mathcal{X}, +)$ are groups and, for any $(y, x), (y', x') \in R$, we have $(y \cdot y', x + x') \in R$.*

**Definition 8** (Homomorphic NIZK). *We say that a NIZK argument-system* $(\texttt{Setup}, \texttt{Prov}, \texttt{Ver}, \texttt{Sim})$ *for a homomorphic-relation $R \subseteq \mathcal{Y} \times \mathcal{X}$ is itself* homomorphic *if there are groups $(\mathcal{R}, +), (\mathcal{P}, \cdot)$ for the randomness of the prover and the proofs themselves respectively, such that for any $(y, x), (y', x') \in R$, any $r, r' \in \mathcal{R}$:*

$$\pi = \texttt{Prov}(y, x; r),\ \pi' = \texttt{Prov}(y', x'; r') \quad \Rightarrow \quad \pi \cdot \pi' = \texttt{Prov}(y \cdot y', x + x'; r + r')$$

*where $\pi, \pi' \in \mathcal{P}$.*

We now connect the above definitions of homomorphic primitives to our construction in Figure 2, showing how to re-randomize the secret-keys if $\mathcal{E}_1, \mathcal{E}_2$, and $\Pi$ are homomorphic. First, we show that the plaintext-equality relation is homomorphic if we *fix* the public-keys $pk_1, pk_2$. That is, for each $pk_1, pk_2$, define the relation $R_{eq}^{(pk_1, pk_2)} \subseteq R_{eq}$ where $pk_1, pk_2$ are fixed and the statements only consist of $(c_1, c_2)$. It is easy to verify the following lemma.

**Lemma 9.** *If $\mathcal{E}_1, \mathcal{E}_2$ are homomorphic with the same message-group $\mathcal{M}$, randomness-groups $\mathcal{R}_1, \mathcal{R}_2$, and ciphertext-groups $\mathcal{C}_1, \mathcal{C}_2$ respectively, then, for any $pk_1, pk_2$, the relation $R_{eq}^{(pk_1, pk_2)}$ is a homomorphic relation over $\mathcal{Y} = \mathcal{C}_1 \times \mathcal{C}_2$ and $\mathcal{X} = \mathcal{M} \times \mathcal{R}_1 \times \mathcal{R}_2$.*

To simplify the discussion, we will say that a proof-system $\Pi$ for $R_{eq}$ is homomorphic if, for *every fixed choice* of the public-keys $pk_1, pk_2$, it is homomorphic for the (homomorphic) relations $R_{eq}^{(pk_1, pk_2)}$. Now assume that $\mathcal{E}_1, \mathcal{E}_2$ are two homomorphic encryption schemes satisfying the requirements of Lemma 9, and that $\Pi$ is a homomorphic NIZK for $R_{eq}$, with randomness-group $\mathcal{R}_3$ and proof-group $\mathcal{P}$. In Figure 3, we show how to re-randomize the secret keys of our LIRR construction from Figure 2.

The main idea is to re-randomize the ciphertext $c_2$ in the secret key (without modifying the encrypted message), by multiplying $c_2$ with a random encryption $c'$ of the identity message $1_\mathcal{M}$. We then need to update the proof $\pi$ in the

```
ReRand(sk): Parse $sk = (c, \pi)$. Choose $(r_2', r_3') \leftarrow \mathcal{R}_2 \times \mathcal{R}_3$.
  Set     $c' := \mathtt{Enc}^2_{pk_2}(1_{\mathcal{M}}; r_2')$,
          $\pi' := \mathtt{Prov}((1_{\mathcal{C}_1}, c'), (1_{\mathcal{M}}, 0_{\mathcal{R}_1}, r_2'); r_3')$
  Output $sk^* = (c \cdot c', \pi \cdot \pi')$.
```

Figure 3. Re-randomization

secret key to look like a *fresh new proof* of the *new true statement* $(c_1, c_2 \cdot c') \in L_{eq}$. We do so by multiplying $\pi$ with a random proof $\pi'$ of the true statement $(1_{\mathcal{C}_1}, c') \in L_{eq}$.

**Lemma 10.** *The re-randomization method in Figure 3 satisfies the* re-randomization of LIRR *(Definition 3).*

## B. Leakage-Indistinguishability

We are left to show the leakage-indistinguishability of our construction. To do so, we need to define a new security property, called *leakage-of-ciphertext non-malleability*, for the encryption scheme $\mathcal{E}_2$. Intuitively, this property says that, given $\ell$ bits of leakage on a *ciphertext* $c$, the adversary cannot produce a *related ciphertext* $c^*$.

**Definition 11** (LoC-NM Encryption)**.** *A public-key encryption scheme* $\mathcal{E} = (\mathtt{KeyGen}, \mathtt{Enc}, \mathtt{Dec})$ *is* $\ell$-leakage-of-ciphertext non-malleable ($\ell$-LoC NM) *if, for any PPT adversary $\mathcal{A}$, we have* $\left|\Pr[\mathcal{A} \text{ wins }] - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda)$ *in the following game:*

- *The challenger chooses* $(pk, sk) \leftarrow \mathtt{KeyGen}(1^\lambda)$ *and gives pk to* $\mathcal{A}$*.*
- $\mathcal{A}$ *chooses two messages* $m_0, m_1$*.*
- *Challenger chooses* $b \leftarrow \{0,1\}$, $c \leftarrow \mathtt{Enc}_{pk}(m_b)$ *but does not give c to* $\mathcal{A}$*.*
- $\mathcal{A}$ *can make up to* $\ell$ *queries to a leakage-oracle* $\mathcal{O}^\lambda_c(\cdot)$*, on the* ciphertext $c$.
- $\mathcal{A}$ *chooses* any $c^*$ *and gets* $m^* = \mathtt{Dec}_{sk}(c^*)$*.*
- $\mathcal{A}$ *outputs a bit* $\tilde{b}$ *and wins if* $\tilde{b} = b$*.*

REMARKS ON THE DEFINITION. Note that, in the definition, the leakage is only on the ciphertext $c$ and *not* on the secret-key $sk$. It is easy to see that *even* 1-LoC-NM security (i.e. adv. gets only 1 bit of leakage) *already* implies semantic-security, since a 1-bit leakage function can act as a distinguisher. On the other hand, the standard notion of non-malleable encryption from [15] implies $\ell$-LoC-NM security where $\ell$ approaches the message-size of the scheme. This is because an adversary that leaks less than $\ell$ bits about a ciphertext $c$ is unlikely to be able to re-produce $c$ exactly, and the decryption of any other ciphertext $c^* \neq c$ safely keeps the challenge message hidden. However, non-malleable encryption is inherently *not* homomorphic while, as we will soon see, LoC-NM encryption is simpler to achieve and can be homomorphic as well.

APPLICATION TO LEAKAGE-INDISTINGUISHABILITY. We now show that, if the scheme $\mathcal{E}_2$ in our construction is $\ell$-LoC-NM, then the construction satisfies $\ell$-leakage-indistinguishability (Definition 3). This is because the secret-key contains a ciphertext $c$, and the decision wether the secret-key is "good" or not depends on the encrypted message. Therefore, the ability to create a secret-key which is in the same category as a challenge key, requires the ability to create "related ciphertexts".

**Lemma 12.** *Assume that, in the construction of Figure 2, the scheme $\mathcal{E}_2$ is $\ell$-LoC-NM and that $\Pi$ is a secure NIZK. Then the construction satisfies the leakage-indistinguishability property of LIRR (Definition 3).*

## C. Summary of Construction and its Requirements

The following theorem follows directly from Lemmata 5, 10 and 12 and Theorem 4.

**Theorem 13.** *The construction in Figure 2, with the re-randomization procedure in Figure 3, is a secure $\ell$-LIRR as long as the components $\mathcal{E}_1, \mathcal{E}_2, \Pi$ satisfy:*

- *$\mathcal{E}_1, \mathcal{E}_2$ are homomorphic encryption schemes with perfect correctness and a common message-space $\mathcal{M}$.*
- *$\mathcal{E}_1$ is one-way secure and $\mathcal{E}_2$ is $\ell$-LoC-NM secure.*
- *$\Pi$ is a homomorphic NIZK argument system for the plaintext-equality relation $R_{eq}$.*

*Therefore, the existence of such components $\mathcal{E}_1, \mathcal{E}_2, \Pi$ implies the existence of a $\ell$-CLR-OWR.*

## V. INSTANTIATING THE COMPONENTS

In this section, we now show how to instantiate the homomorphic encryption and NIZK schemes $\mathcal{E}_1, \mathcal{E}_2, \Pi$ so as to satisfy the requirements of Theorem 13. We will do so under the *K-linear assumption* (a generalization of DDH) in Bilinear Groups. The main tool here will be the Groth-Sahai (GS) NIZK argument system, which we notice to be homomorphic. This observation is influenced by [6], who noticed that GS proofs are re-randomizable and malleable.

NUMBER THEORETIC ASSUMPTIONS. Let $\mathcal{G}$ be a group generation algorithm, which outputs $(p, \mathbb{G}, \mathbf{g}) \leftarrow \mathcal{G}(1^\lambda)$, where $\mathbb{G}$ is (some description of) a cyclic group of prime order $p$ with generator $g$.

*Decisional Diffie-Hellman (DDH).:* The DDH assumption on $\mathcal{G}$ states that

$$(\mathbb{G}, \mathbf{g}_0, \mathbf{g}_1, \mathbf{g}_0^r, \mathbf{g}_1^r) \overset{c}{\approx} (\mathbb{G}, \mathbf{g}_0, \mathbf{g}_1, \mathbf{g}_0^{r_0}, \mathbf{g}_1^{r_1})$$

where $(p, \mathbb{G}, \mathbf{g}) \leftarrow \mathcal{G}(1^\lambda)$, $\mathbf{g}_0, \mathbf{g}_1 \leftarrow \mathbb{G}$, and $r, r_0, r_1 \leftarrow \mathbb{Z}_p$.

*K-Linear [9], [23], [33].:* Let $K \geq 1$ be constant. The $K$-Linear assumption on $\mathcal{G}$ states that

$$(\mathbb{G}, \mathbf{g}_0, \mathbf{g}_1, \ldots, \mathbf{g}_K, \mathbf{g}_1^{r_1}, \mathbf{g}_2^{r_2}, \ldots, \mathbf{g}_K^{r_K}, \mathbf{g}_0^{\sum_{i=1}^K r_i}) \overset{c}{\approx}$$
$$(\mathbb{G}, \mathbf{g}_0, \mathbf{g}_1, \ldots, \mathbf{g}_K, \mathbf{g}_1^{r_1}, \mathbf{g}_2^{r_2}, \ldots, \mathbf{g}_K^{r_K}, \mathbf{g}_0^{r_0})$$

where $(p, \mathbb{G}, \mathbf{g}) \leftarrow \mathcal{G}(1^\lambda)$, $\mathbf{g}_0, \ldots, \mathbf{g}_K \leftarrow \mathbb{G}$, and $r_0, r_1, \ldots, r_K \leftarrow \mathbb{Z}_p$.

The $K$-linear assumption for $K = 1$ is the DDH assumption (in disguise). Also, $K$-linear implies $(K+1)$-linear, so the

Let $\mathsf{prms} = (p, \mathbb{G}, \mathbf{g}) \leftarrow \mathcal{G}(1^\lambda)$.

- $\mathtt{KeyGen}(\mathsf{prms})$: Choose $x \leftarrow \mathbb{Z}_p$. Set $\mathbf{f} = \mathbf{g}^x$. Output $(pk = \mathbf{f}, \quad sk = x)$.
- $\mathtt{Enc}_{pk}(\mathbf{m})$: Choose $r \leftarrow \mathbb{Z}_p$. Output $c := (\mathbf{m}\,\mathbf{g}^r, \quad \mathbf{f}^r)$.
- $\mathtt{Dec}_{sk}(c)$: Parse $c = (\mathbf{z}, \mathbf{c})$. Output $\mathbf{z}\,\mathbf{c}^{-1/x}$.

Figure 4. ElGamal Encryption (variant)

assumptions get progressively *weaker* as $K$ grows. For $K = 2$, it is called the *decisional-linear (DLIN)* assumption.

PAIRINGS. Let $\mathcal{G}_{pair}$ be a *pairing* generation algorithm, which outputs $(p, \mathbb{G}, \Gamma, \mathbb{G}_T, e, \mathbf{g}, \boldsymbol{\gamma}) \leftarrow \mathcal{G}_{pair}(1^\lambda)$, where $\mathbb{G}, \Gamma, \mathbb{G}_T$ are groups of prime order $p$, $\mathbf{g}$ is a generator of $\mathbb{G}$, and $\boldsymbol{\gamma}$ is a generator of $\Gamma$. The map $e : \mathbb{G} \times \Gamma \rightarrow \mathbb{G}_T$ is an efficiently computable *bilinear map*, which satisfies: (1) *non-degeneracy*: the element $e(\mathbf{g}, \boldsymbol{\gamma})$ generates $\mathbb{G}_T$ , and (2) *bi-linearity*: for any $a, b \in \mathbb{Z}_p$, we have $e(\mathbf{g}^a, \boldsymbol{\gamma}^b) = e(\mathbf{g}, \boldsymbol{\gamma})^{ab}$. We call $\mathbb{G}, \Gamma$ the base groups and $\mathbb{G}_T$ is the target group. If $\mathbb{G} = \Gamma$ and $\mathbf{g} = \boldsymbol{\gamma}$, we say that the pairing is *symmetric*.

OUR INSTANTIATIONS. We show how to instantiate the components $\mathcal{E}_1, \mathcal{E}_2$ and $\Pi$ assuming the $K$-linear assumption, for any constant $K \geq 1$, holds in *both base groups* $\mathbb{G}$ and $\Gamma$ of a *symmetric or asymmetric* pairing $\mathcal{G}_{pair}$. For $K = 1$, this is equivalent to assuming DDH holds in the base groups. Although, the DDH assumption is always *invalid* in the case of *symmetric* pairings ($\mathbb{G} = \Gamma$), it is believed to hold for some *asymmetric* pairings, if there is no efficiently computable linear mapping between $\mathbb{G}$ and $\Gamma$. This is also sometimes called the *External Diffie-Hellman (SXDH)* assumption [32], [9], [5], [18], [34]. For symmetric-pairings (when $\mathbb{G} = \Gamma$), it is often reasonable to assume that the $K$-linear assumption holds in $\mathbb{G}$ for $K = 2$ (and higher). Although the $K$-linear assumption gets progressively weaker as $K$ grows, there seems to be little reason to use $K > 2$ (i.e. anything other than DLIN) in practice.

For simplicity, we only concentrate on the $K = 1$ (DDH) case in the main body, and the generalization to $K > 1$ appears in the full version [13]. We note that our encryption schemes $\mathcal{E}_1, \mathcal{E}_2$ do not make use of pairing operations at all, but the NIZK argument system $\Pi$ will.

## *A. The encryption schemes $\mathcal{E}_1, \mathcal{E}_2$*

For the scheme $\mathcal{E}_1$, we can use any homomorphic one-way secure encryption. We choose to use (a slight variant of) the ElGamal encryption scheme, shown in Figure 4.

For the scheme $\mathcal{E}_2$, we need a homomorphic encryption satisfying $\ell$-LoC-NM security. We use a variant of the "Cramer-Shoup Lite" (CS-Lite) [12] scheme shown in Figure 5. We implicitly show that $\ell$-LoC-NM security can be constructed from any "1-*universal hash proof system*" (of which CS-Lite is an example), but, since we will need a scheme based on $K$-linear, we restrict ourselves to generalizations of CS-Lite.

For $n = 0$, the encryption scheme above is already semantically-secure under the DDH assumption. For $n = 1$,

Let $n \in \mathbb{Z}^+$. Let $\mathsf{prms} = (p, \mathbb{G}, \mathbf{g}_0, \mathbf{g}_1)$ where $(p, \mathbb{G}, \mathbf{g}_0) \leftarrow \mathcal{G}(1^\lambda)$ and $\mathbf{g}_1 \leftarrow \mathbb{G}$.

- $\mathtt{KeyGen}(\mathsf{prms})$: Choose $(n + 1)$ random pairs $\left\{\vec{x}_i = (x_{i,0}, x_{i,1})^\top \leftarrow \mathbb{Z}_p^2\right\}_{i=0}^n$.
  Set $\mathbf{h}_0 := \mathbf{g}_0^{x_{0,0}}\mathbf{g}_1^{x_{0,1}}, \ \mathbf{h}_1 := \mathbf{g}_0^{x_{1,0}}\mathbf{g}_1^{x_{1,1}} \ , \ldots, \ \mathbf{h}_n := \mathbf{g}_0^{x_{n,0}}\mathbf{g}_1^{x_{n,1}}$.
  Output $pk := (\mathbf{h}_0, \mathbf{h}_1, \ldots, \mathbf{h}_n)$, $sk := (\vec{x}_0, \ldots, \vec{x}_n)$.
- $\mathtt{Enc}_{pk}(\mathbf{m})$: To encrypt a message $\mathbf{m} \in \mathbb{G}$, choose $r \leftarrow \mathbb{Z}_p$ and compute $\mathbf{c}_0 := \mathbf{h}_0^r, \ \ \mathbf{c}_1 := \mathbf{h}_1^r \ \ , \ldots, \ \ \mathbf{c}_n := \mathbf{h}_n^r$.
  Output $c := (\mathbf{g}_0^r, \ \mathbf{g}_1^r, \ \mathbf{m}\,\mathbf{c}_0, \ \mathbf{c}_1, \ \ldots, \ \mathbf{c}_n)$.
- $\mathtt{Dec}_{sk}(c)$: Parse $c = (\mathbf{y}_0, \mathbf{y}_1, \mathbf{z}, \mathbf{c}_1, \ldots, \mathbf{c}_n)$. Set $\tilde{\mathbf{c}}_0 := \mathbf{y}_0^{x_{0,0}}\mathbf{y}_1^{x_{0,1}}, \ \tilde{\mathbf{c}}_1 := \mathbf{y}_0^{x_{1,0}}\mathbf{y}_1^{x_{1,1}}, \ \ldots \ , \tilde{\mathbf{c}}_n := \mathbf{y}_0^{x_{n,0}}\mathbf{y}_1^{x_{n,1}}$.
  If $\tilde{\mathbf{c}}_1 \stackrel{?}{=} \mathbf{c}_1, \ldots, \tilde{\mathbf{c}}_n \stackrel{?}{=} \mathbf{c}_n$, then output $\mathbf{z}/\tilde{\mathbf{c}}_0$. Else output $\perp$.

Figure 5. Multiple CS-Lite Scheme

we recover the original CS-Lite encryption scheme, in which the ciphertext includes an additional *verification element* $\mathbf{c}_1$, which certifies that the ciphertext is well-formed. The CS-Lite scheme is known to be CCA-1 secure under the DDH assumption, but CCA-1 security does *not*, in general, seem to imply LoC-NM security. We show that the Multiple CS-Lite scheme is $\ell$-LoC-NM secure, where the leakage-bound $\ell$ is proportional to the number of verification elements $n$.

The high level idea goes as follows. By the DDH assumption, the adversary cannot distinguish a correctly generated *challenge ciphertext* from one where $\mathbf{y}_0 = \mathbf{g}_0^{r_0}, \mathbf{y}_1 = \mathbf{g}_1^{r_1}$ are uniformly random and independent values (not a DDH tuple), and $\mathbf{c}_0, \ldots, \mathbf{c}_n$ are replaced with the corresponding $\tilde{\mathbf{c}}_i$, as computed as during decryption. In that case, the value $\tilde{\mathbf{c}}_0$ is uniformly random over the randomness of the secret vector $\vec{x}_0$, and $\mathbf{m}$ is *statistically* hidden by the ciphertext and the public-key. We only have to argue that the decryption query keeps $\mathbf{m}$ hidden. If the ciphertext $c^*$ in the decryption query includes a DDH-tuple $\mathbf{y}_0^*, \mathbf{y}_1^*$ , then the decrypted message $m^*$ is determined by the public-key alone, and does not reveal any additional information about $\vec{x}_0$ or $\mathbf{m}$. On the other hand, if $\mathbf{y}_0^*, \mathbf{y}_1^*$ are not a DDH-tuple, we argue that $c^*$ decrypts to $\perp$. Consider the values $\tilde{\mathbf{c}}_1^*, \ldots, \tilde{\mathbf{c}}_n^*$, used by the challenger to check "well-formedness" during the decryption of $c^*$. These values are uniformly random over the randomness of the secret components $\vec{x}_i$. Unfortunately, they *not* independent of the challenge-ciphertext components $\tilde{\mathbf{c}}_i$.[2] However, since the adversary only sees $\ell$ bits of leakage on the challenge ciphertext, this is not enough to guess all of the values $\tilde{\mathbf{c}}_i^*$ correctly (if $n$ is big enough), and so the ciphertext $c^*$ will decrypt to $\perp$.

**Theorem 14.** *Under the DDH assumption, the multiple CS-Lite scheme (Figure 5) with parameter $n$, is an $\ell$-LoC-NM secure encryption with leakage $\ell = n\log(p) - \lambda$. The ratio of leakage to ciphertext-size approaches 1, as $n$ grows. The scheme is homomorphic over the messages $\mathcal{M} = \mathbb{G}$, randomness $\mathcal{R} = \mathbb{Z}_p$, and ciphertexts $\mathcal{C} = \mathbb{G}^{n+3}$.*

[2]For example, if $(\mathbf{y}_0^*, \mathbf{y}_1^*) = (\mathbf{y}_0, \mathbf{y}_1)$, then $\tilde{\mathbf{c}}_i^* = \tilde{\mathbf{c}}_i$ are the same! But even if $(\mathbf{y}_0^*, \mathbf{y}_1^*) \neq (\mathbf{y}_0, \mathbf{y}_1)$, the values are still not independent.

The proof of the Theorem 14 appears in the full version, where the schemes $\mathcal{E}_1, \mathcal{E}_2$ are also generalized to get security under the $K$-linear assumption, for arbitrary values of $K$.

## *B. The NIZK System $\Pi$ : Groth-Sahai Proofs*

We consider the language of *satisfiable systems of linear equations*, over a group $\mathbb{G}$ of primer order $p$. A system of $M$ equations over $N$ variables consists of a matrix of coefficients $\mathbf{B} \in \mathbb{G}^{M \times N}$ and a vector of target values $\vec{\mathbf{c}} \in \mathbb{G}^M$:

$$\mathbf{B} = \left\{ \vec{\mathbf{b}}_i = (\mathbf{b}_{i,1}, \mathbf{b}_{i,2}, \dots, \mathbf{b}_{i,N}) \right\}_{i=1}^{M} \ , \ \vec{\mathbf{c}} = (\mathbf{c}_1, \dots, \mathbf{c}_M)$$

We say that the system $(\mathbf{B}, \vec{\mathbf{c}})$ is *satisfiable* if there exists a vector $\vec{x} = (x_1, \dots, x_N) \ \in \mathbb{Z}_p^N$ such that

$$\mathbf{b}_{i,1}^{x_1} \mathbf{b}_{i,2}^{x_2} \dots \mathbf{b}_{i,N}^{x_N} = \mathbf{c}_i \quad \text{for} \quad i \in \{1, \dots, M\}.$$

We call the vector $\vec{x}$, the *satisfying assignment* for the system $(\mathbf{B}, \vec{\mathbf{c}})$. We define the relation $R_{linear}$ as consisting of all pairs $((\mathbf{B}, \vec{\mathbf{c}}), \vec{x})$, where the system $(\mathbf{B}, \vec{\mathbf{c}})$ acts as a *statement* and the satisfying assignment $\vec{x}$ acts as a *witness*. We define the corresponding language $L_{linear}$ of satisfiable linear equations.

If we *fix* the coefficients $\mathbf{B}$, and define the relation $R^{\mathbf{B}}_{linear} = \{(\vec{\mathbf{c}}, \vec{x}) \ : \ ((\mathbf{B}, \vec{\mathbf{c}}), \vec{x}) \in R_{linear}\}$, The Groth-Sahai (GS) [21] NIZK argument system is (among other things) an argument system for the relation $R_{linear}$. We give a brief overview of the GS construction in the full version [13]. We notice that the GS NIZKs already give us a homomorphic NIZK argument system for $R_{linear}$, in the above-described sense. Therefore, the following lemma follows directly from the work of [21] (and generalizes to the $K$-linear assumption).

**Lemma 15.** *Assume the DDH assumption holds in both base groups of some pairing $\mathcal{G}_{pair}$. Then there exists a* homomorphic *NIZK argument system for the relation $R_{linear}$.*

PROOFS OF PLAINTEXT EQUALITY FOR $\mathcal{E}_1, \mathcal{E}_2$. Let $\mathcal{E}_1$ be the ElGamal encryption scheme and $\mathcal{E}_2$ be the CS-Lite encryption scheme as described in Figure 4 and Figure 5, respectively. It is relatively easy to show (see full version [13]) that the corresponding language for plaintext-equality $L_{eq}$ can be expressed in terms of a satisfiable sets of linear equations. In particular, for each statement $(\mathsf{prms}, pk, c_1, c_2)$ there exists some system $(\mathbf{B}_{eq}, \vec{\mathbf{c}}_{eq})$ such that the statement is in $L_{eq}$ if and only if the system is satisfiable. Moreover, the matrix $\mathbf{B}_{eq}$ only depends on $(\mathsf{prms}, pk)$ while the components of $(c_1, c_2)$ define $\vec{\mathbf{c}}$. Therefore, the GS scheme is a homomorphic NIZK for the plaintext-equality relation $R_{eq}$.

## *C. Parameters of Instantiation*

In the full version [13], we generalize the schemes $\mathcal{E}_1, \mathcal{E}_2$ and the NIZK system $\Pi$ to the $K$-linear assumption. If $K$ is constant, the ciphertext $c_2$ consists of $n + O(1)$ group elements. The proof $\pi$ grows linearly with $K$, and requires $Kn + O(1)$ group elements. The leakage, is $\ell = n\log(p) - \lambda$ *bits*. Therefore, the relative leakage of the scheme approaches $\frac{\ell}{|sk|} = \frac{1}{K+1}$. Recall that, in practice, only the choices $K = 1, 2$ are interesting, yielding a relative leakage of $\frac{1}{2}$ (based on DDH) and $\frac{1}{3}$ (based on DLIN) respectively.

**Theorem 16.** *Fix a constant $K \geq 1$, and assume that the $K$-linear assumption holds in the base group(s) of some pairing $\mathcal{G}_{pair}$. Then, for any constant $\epsilon > 0$, there exists an $\ell$-CLR-OWR, with relative-leakage $\frac{\ell}{|sk|} \geq \frac{1}{K+1} - \epsilon$. The public/secret keys of the scheme contain $O(1)$ group elements, and the algorithms of the scheme use $O(1)$ exponentiation/pairing operations.*

## VI. APPLICATIONS: SIGNATURES, ID SCHEMES, AKA

We now show how to use CLR-OWRs to build other CLR primitives; in particular, we define and achieve CLR security for signatures, ID schemes, and Authenticated Key Agreement (AKA). Our constructions are based on the prior works of [3], [27], [14], which (sometimes implicitly) construct "one-time" leakage-resilient versions of these primitives, from underlying "one-time" leakage-resilient OWRs. We notice that these constructions naturally extend to the continuous case. The details are given in the appendices, and we just sketch the results.

SIGNATURES. An $\ell$-CLR Signature scheme is defined analogously to a CLR-OWR: the adversary's attack consists of many *rounds*, during each of which the adversary can interleave arbitrary signature queries with up to $\ell$ leakage queries on the current secret key. At the end of each round, the key is updated using a `ReRand` procedure. Our construction is based on the the leakage-resilient signature scheme of [14], which slightly generalizes the original scheme of [27]. We start with a CLR-OWR $\mathcal{C} = (\mathtt{KeyGen}^{\mathcal{C}}, \mathtt{ReRand}, \mathtt{Ver}^{\mathcal{C}})$, and a NIZK $\Psi = (\mathtt{Setup}^{\Psi}, \mathtt{Prov}^{\Psi}, \mathtt{Ver}^{\Psi})$ for the relation

$$R_{\mathcal{C}} = \{(y, x) \mid y = (pk, m), \quad x = sk \\ \text{s.t.} \quad \mathtt{Ver}^{\mathcal{C}}(pk, sk) = 1\}.$$

The signature scheme construction is shown in Figure 6. For security, we need the NIZK system $\Psi$ to be *true-simulation extractable (tSE)*; even if an adversary sees simulated proofs of arbitrary true statements $y \in L_R$, if she produces a valid proof $\psi^*$ for a new statement $y^*$, then there is a way to *extract* a valid witness $x^*$ from $\psi^*$, so that $(y^*, x^*) \in R$. Notice that this explains the (seemingly useless) role of $m$ in the relation $R_{\mathcal{C}}$. Even if the adversary sees simulated proofs for statements that contain many different values $m$, any proof she produces for a new $m^*$ is extractable. We note that, as observed in [14], tSE NIZKs can be constructed by either (1) composing a *simulation-sound* NIZK with a CPA-secure encryption, yielding the scheme of [27], or (2) composing a standard NIZK with a CCA-secure encryption, yielding a (possibly) more efficient scheme.

On a high level, this construction preserves the (continuous) leakage-resilience security of the underlying OWR,

- $\mathtt{KeyGen}(1^\lambda)$: Run $(pk, sk) \leftarrow \mathtt{KeyGen}^{\mathcal{C}}(1^\lambda)$, $(\text{CRS}, \cdot) \leftarrow \mathtt{Setup}^{\Psi}(1^\lambda)$. Output: $vk := (pk, \text{CRS})$, $sk$.
- $\mathtt{Sign}_{sk}(m)$: Output $\sigma \leftarrow \mathtt{Prov}^{\Psi}((pk, m), sk)$.
- $\mathtt{SigVer}_{vk}(m, \sigma)$: Output $\mathtt{Ver}^{\Psi}((pk, m), \sigma)$.
- $\mathtt{ReRand}(sk)$: Run the re-randomization procedure of the CLR-OWR $\mathcal{C}$.

Figure 6. A CLR-Signature Scheme from a CLR-OWR and a tSE NIZK

since the signatures do not reveal any information about $sk$. In particular, the signatures can be simulated using the trapdoor TK for the NIZK system, without any knowledge of $sk$. Nevertheless, we can extract a valid secret-key $sk^*$ from any forgery $\sigma^*$ on a new message $m^*$. See the full version [13] for the formal definitions of CLR secure signatures, tSE NIZKs and a proof of security for the above construction.

In the full version, we also show that the above signature scheme can be instantiated *efficiently* from our efficient construction of a CLR-OWR based on the $K$-linear assumption. This requires us to use the efficient GS NIZK system to prove statements about the GS NIZK proofs themselves!

ID SCHEMES AND AKA PROTOCOLS. The CLR security of identification (ID) schemes and authenticated key agreement (AKA) protocols is defined naturally in the full version [13]. We give black-box constructions of such schemes from CLR signatures, naturally extending the prior constructions of [3], [14] in the (one-time) LR setting. In particular, we can use a CLR signature to define a simple CLR ID scheme, where the prover simply signs a random challenge chosen by the verifier. For AKA, the parties use a CLR signature scheme to set up a public-key infrastructure (PKI). Then, any pair of parties can agree on a fresh ephemeral session keys, by running a passively-secure key agreement protocol (such as the Diffie-Hellman key agreement, or its generalization to $K$-linear), and authenticating the flows of the protocol by signing them. Even if the adversary repeatedly sees (limited) leakage on many updated versions of the long-term signing keys, she will be unable to impersonate an honest user, or break the privacy of past session-keys.

SIGNATURES/ID/AKA UNDER $K$-LINEAR. The constructions of signature, ID, and AKA schemes use the same secret-key as the underlying CLR-OWR, and preserve the absolute and relative leakage of the CLR-OWR. Thus, as a corollary of our CLR-OWR construction under the $K$-linear assumption, and our constructions of ID/Signatures/AKA from CLR-OWR, we get the following result.

**Theorem 17.** *Fix any constant $K \geq 1$, and assume that the $K$-linear assumption holds in the base group(s) of some pairing $\mathcal{G}_{pair}$. Then, for any $\epsilon > 0$, there exist $\ell$-CLR Signatures, ID schemes, and AKA protocols, with relative-leakage $\frac{\ell}{|sk|} \geq \frac{1}{K+1} - \epsilon$. The above schemes are efficient: public-keys, secret-keys, communication/signatures each consist of $O(1)$ group elements and all algorithms use at most $O(1)$ group operations.*

## VII. EXTENSIONS

LEAKING RANDOM COINS OF REFRESHING. In our basic model of CLR security, we only consider leakage on the secret keys $sk$ stored in memory *in-between* refresh operations. However, the internal secret state of the system also depends on the random coins used by the refreshing algorithm and the cryptographic computation itself (e.g. signing, identifying ...). A-priori, it may be possible that even some very small amount of such leakage (say 1 bit per period) can harm security, since the total amount of leakage observed by the adversary is large. Let us start with leakage on the random coins of refreshing. Brakerski et al. [10] show that a *particular* CLR scheme is secure w.r.t. logarithmic (in the security parameter) amount of such leakage. It was observed by Brent Waters [35] that this, in-fact, holds *generically* for all CLR OWRs, Signatures, and Public-Key Encryption (PKE) schemes (and carries over to our generic constructions of ID schemes and AKA from these primitives). We formalize this observation in the full version [13]. The result shows that our schemes are all resilient to logarithmic leakage of the randomness-of-refreshing (or more, if we are willing to assume exponential exact security of the $K$-linear assumption). It remains an important open problem to allow for super-logarithmic leakage of the refreshing randomness under standard assumptions against polynomially bounded attackers.

LEAKING RANDOM COINS OF COMPUTATION. Allowing leakage of the random coins of the cryptographic computation (signing, identifying, authenticating) is perhaps even more important since, presumably, such computation occurs much more frequently than key-refreshing. In the full version [13], we show that if one starts with a CLR-OWR having relative leakage $\alpha$, one can generically construct the following schemes with leakage-of-randomness security: (1) an ID scheme in the standard model with relative leakage $\alpha$, (2) a signature scheme in the random-oracle model with relative leakage $\alpha/2$, (3) an AKA protocol in the random-oracle model with relative leakage $\alpha$. We note that constructions differ from the ones presented in Section VI. It remain an important open problem to construct (even one-time) leakage-resilient signature schemes with leakage-of-randomness security in the *standard model*.

NOISY & CONCURRENT LEAKAGE. In the full version, we also explore two additional extensions. First, we consider *noisy* leakage (previously studied in [29]), where instead of bounding the *size* of the leakage, we bound the *entropy-loss* that the leakage causes on the secret-key. In other words, we allow long leakage, as long as it does not reveal too much useful information. Second, instead of considering an adversary who leaks on the secret-keys of the device in sequential leakage-periods, we also consider the *concurrent* setting, where the adversary can leak on all the secret keys that ever were or will be used concurrently (as long as the amount of leakage is bounded on each such key). We show that our primitives are also secure in these stronger models.

## REFERENCES

[1] A. Akavia, S. Goldwasser, and V. Vaikuntanathan, "Simultaneous hardcore bits and cryptography against memory attacks," in *TCC*, ser. Lecture Notes in Computer Science, O. Reingold, Ed., vol. 5444. Springer, 2009, pp. 474–495.

[2] J. Alwen, Y. Dodis, M. Naor, G. Segev, S. Walfish, and D. Wichs, "Public-key encryption in the bounded-retrieval model," in *EUROCRYPT*, ser. Lecture Notes in Computer Science, H. Gilbert, Ed., vol. 6110. Springer, 2010, pp. 113–134.

[3] J. Alwen, Y. Dodis, and D. Wichs, "Leakage-resilient public-key cryptography in the bounded-retrieval model," in *CRYPTO*, ser. Lecture Notes in Computer Science, S. Halevi, Ed., vol. 5677. Springer, 2009, pp. 36–54.

[4] ——, "Survey: Leakage resilience and the bounded retrieval model." 2009.

[5] L. Ballard, M. Green, B. de Medeiros, and F. Monrose, "Correlation-resistant storage via keyword-searchable encryption," Cryptology ePrint Archive, Report 2005/417, 2005, http://eprint.iacr.org/.

[6] M. Belenkiy, J. Camenisch, M. Chase, M. Kohlweiss, A. Lysyanskaya, and H. Shacham, "Randomizable proofs and delegatable anonymous credentials," in *CRYPTO*, ser. Lecture Notes in Computer Science, S. Halevi, Ed., vol. 5677. Springer, 2009, pp. 108–125.

[7] M. Blum, P. Feldman, and S. Micali, "Non-interactive zero-knowledge and its applications (extended abstract)," in *STOC*. ACM, 1988, pp. 103–112.

[8] M. Blum, A. D. Santis, S. Micali, and G. Persiano, "Noninteractive zero-knowledge," *SIAM J. Comput.*, vol. 20, no. 6, pp. 1084–1118, 1991.

[9] D. Boneh, X. Boyen, and H. Shacham, "Short group signatures," in *CRYPTO*, ser. Lecture Notes in Computer Science, M. K. Franklin, Ed., vol. 3152. Springer, 2004, pp. 41–55.

[10] Z. Brakerski, Y. T. Kalai, J. Katz, and V. Vaikuntanathan, "Overcoming the hole in the bucket: Public-key cryptography resilient to continual memory leakage," FOCS, 2010, full version: http://eprint.iacr.org/2010/278.

[11] R. Canetti, Y. Dodis, S. Halevi, E. Kushilevitz, and A. Sahai, "Exposure-resilient functions and all-or-nothing transforms," in *EUROCRYPT*, ser. Lecture Notes in Computer Science, B. Preneel, Ed., vol. 1807. Springer, 2000, pp. 453–469.

[12] R. Cramer and V. Shoup, "A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack," in *CRYPTO*, ser. Lecture Notes in Computer Science, H. Krawczyk, Ed., vol. 1462. Springer, 1998, pp. 13–25.

[13] Y. Dodis, K. Haralambiev, A. López-Alt, and D. Wichs, "Cryptography against continuous memory attacks," Cryptology ePrint Archive, Report 2010/196, 2010.

[14] ——, "Efficient public-key cryptography in the presence of key leakage," Cryptology ePrint Archive, Report 2010/154, 2010.

[15] D. Dolev, C. Dwork, and M. Naor, "Non-malleable cryptography," in *STOC*. ACM, 1991, pp. 542–552.

[16] S. Dziembowski and K. Pietrzak, "Leakage-resilient cryptography," in *FOCS*. IEEE Computer Society, 2008, pp. 293–302.

[17] S. Faust, E. Kiltz, K. Pietrzak, and G. N. Rothblum, "Leakage-resilient signatures," in *TCC*, ser. Lecture Notes in Computer Science, D. Micciancio, Ed., vol. 5978. Springer, 2010, pp. 343–360.

[18] S. D. Galbraith and V. Rotger, "Easy decision-diffie-hellman groups," *LMS Journal of Computation and Mathematics*, vol. 7, p. 2004, 2004.

[19] S. Goldwasser, "Cryptography without (hardly any) secrets ?" in *EUROCRYPT*, ser. Lecture Notes in Computer Science, A. Joux, Ed., vol. 5479. Springer, 2009, pp. 369–370.

[20] S. Goldwasser and G. N. Rothblum, "How to play mental solitaire under continuous side-channels: A completeness theorem using secure hardware," to Appear at CRYPTO 2010.

[21] J. Groth and A. Sahai, "Efficient non-interactive proof systems for bilinear groups," in *EUROCRYPT*, ser. Lecture Notes in Computer Science, N. P. Smart, Ed., vol. 4965. Springer, 2008, pp. 415–432.

[22] S. Halevi, Ed., *Advances in Cryptology - CRYPTO 2009, 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings*, ser. Lecture Notes in Computer Science, vol. 5677. Springer, 2009.

[23] D. Hofheinz and E. Kiltz, "Secure hybrid encryption from weakened key encapsulation," in *CRYPTO*, ser. Lecture Notes in Computer Science, A. Menezes, Ed., vol. 4622. Springer, 2007, pp. 553–571.

[24] Y. Ishai, A. Sahai, and D. Wagner, "Private circuits: Securing hardware against probing attacks," in *CRYPTO*, ser. Lecture Notes in Computer Science, D. Boneh, Ed., vol. 2729. Springer, 2003, pp. 463–481.

[25] A. Joux, Ed., *Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings*, ser. Lecture Notes in Computer Science, vol. 5479. Springer, 2009.

[26] A. Juma and Y. Vahlis, "Protecting cryptographic keys against continual leakage," to appear at CRYPTO 2010.

[27] J. Katz and V. Vaikuntanathan, "Signature schemes with bounded leakage resilience," in *ASIACRYPT*, ser. Lecture Notes in Computer Science, M. Matsui, Ed., vol. 5912. Springer, 2009, pp. 703–720.

[28] S. Micali and L. Reyzin, "Physically observable cryptography," in *TCC*, ser. Lecture Notes in Computer Science, M. Naor, Ed., vol. 2951. Springer, 2004, pp. 278–296.

[29] M. Naor and G. Segev, "Public-key cryptosystems resilient to key leakage," in *CRYPTO*, ser. Lecture Notes in Computer Science, S. Halevi, Ed., vol. 5677. Springer, 2009, pp. 18–35.

[30] K. Pietrzak, "A leakage-resilient mode of operation," in *EUROCRYPT*, ser. Lecture Notes in Computer Science, A. Joux, Ed., vol. 5479. Springer, 2009, pp. 462–482.

[31] A. D. Santis, S. Micali, and G. Persiano, "Non-interactive zero-knowledge with preprocessing," in *CRYPTO*, ser. Lecture Notes in Computer Science, S. Goldwasser, Ed., vol. 403. Springer, 1988, pp. 269–282.

[32] M. Scott, "Authenticated id-based key exchange and remote log-in with simple token and pin number," Cryptology ePrint Archive, Report 2002/164, 2002.

[33] H. Shacham, "A cramer-shoup encryption scheme from the linear assumption and from progressively weaker linear variants," Cryptology ePrint Archive, Report 2007/074, 2007.

[34] E. R. Verheul, "Evidence that xtr is more secure than supersingular elliptic curve cryptosystems," *J. Cryptology*, vol. 17, no. 4, pp. 277–296, 2004.

[35] B. Waters, personal Communication. April 2010.