Cryptography Against Continuous Memory Attacks

Yevgeniy Dodis[*] Kristiyan Haralambiev [†] Adriana López-Alt [‡] Daniel Wichs[§]

October 1, 2010

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 of the system, 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 $K \geq 1$). 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.

[*]Computer Science Dept. NYU. Email: `dodis@cs.nyu.edu`.
[†]Computer Science Dept. NYU. Email: `kkh@cs.nyu.edu`.
[‡]Computer Science Dept. NYU. Email: `lopez@cs.nyu.edu`.
[§]Computer Science Dept. NYU. Email: `wichs@cs.nyu.edu`.

Contents

1 Introduction 3

2 Continuous Leakage Resilience for One-Way Relations (CLR-OWR) 4

3 CLR-OWR Part I: Continuous Leakage from Bounded Leakage 6

4 CLR-OWR Part II: Constructing LIRR from Generic Components 7
4.1 Syntax of Construction and Hardness Properties . . . 7
4.2 Re-randomization Using Homomorphic Encryptions and NIZKs . . . 8
4.3 Leakage-Indistinguishability . . . 9
4.4 Summary of Construction and its Requirements . . . 10

5 CLR-OWR Part III: Instantiating the Components 11
5.1 The encryption schemes $\mathcal{E}_1, \mathcal{E}_2$ . . . 11
5.2 The NIZK System Π : Groth-Sahai Proofs for Linear Equations . . . 13
5.3 Efficiency and Relative-Leakage of Instantiation with $\mathcal{E}_1, \mathcal{E}_2, \Pi$ . . . 14

6 Applications: Signatures, ID Schemes, and AKA 14

7 Extensions 15
7.1 Leakage of Computation . . . 15
7.2 Leakage of Refreshing . . . 16
7.3 Noisy Leakage . . . 17
7.4 Security with Concurrent Leakage Rounds . . . 17

A Prior Work on Leakage-Resilient Cryptography 21
A.1 Bounded-Leakage Memory Attacks . . . 21
A.2 Continuous Leakage Attack Models . . . 22

B Non-Interactive Zero Knowledge 22

C One-Way Secure Encryption 23

D Linear Algebra Notation for Prime-Order Groups 24

E The Encryption Schemes $\mathcal{E}_1, \mathcal{E}_2$ under the K-Linear Assumption 25
E.1 The scheme $\mathcal{E}_1$: Generalized ElGamal Encryption . . . 25
E.2 The scheme $\mathcal{E}_2$: Generalized "Cramer-Shoup Lite" . . . 25
E.3 The Plaintext-Equality Relation R_{eq} . . . 28

F The Proof-System Π : Groth-Sahai (GS) Proof System for Linear Equations 28
F.1 Construction . . . 29
F.2 Correctness and Security . . . 29
F.3 GS Proofs are Homomorphic . . . 30

G Putting It All Together : Efficient CLR-OWRs 31

H The Groth-Sahai (GS) Proof System for General Equations 31
H.1 System of Simple Pairing-Product Equations . . . 32

I CLR Signatures : Generic Construction from CLR-OWR 33

J CLR Signatures : Efficient Instantiation Under K-Linear **34**
J.1 Expressing $R_{\mathcal{C}}$ as a System of Pairing-Product Equations . . . 35
J.2 tSE NIZK for a System of Pairing-Product Equations . . . 35

K CLR Identification (ID) Schemes & CLR Authenticated Key Agreement (AKA) **37**
K.1 CLR ID Schemes . . . 37
K.2 CLR AKA . . . 38

L Noisy Leakage **41**
L.1 Background on Entropy . . . 41
L.2 Properties of Noisy Leakage . . . 41
L.3 Resilience to Noisy Leakage (Proof-Sketch of Theorem 7.5) . . . 42
L.4 Comparison to the Definition of [49] . . . 43

M Leakage of Computation **43**
M.1 Organization . . . 43
M.2 Background: Σ-Protocols . . . 43
M.3 Leakage of Randomness Security: Generic Constructions . . . 44
M.3.1 ID Schemes . . . 44
M.3.2 Signatures . . . 46
M.3.3 “Challenge-Response” Interactive Signatures . . . 47
M.3.4 AKA . . . 47
M.4 A Σ-Protocol for the Secret-Key of a CLR-OWR . . . 47
M.5 Summary of Results of Leakage-of-Randomness . . . 49

N Proofs Omitted from the Main Body **49**
N.1 Theorem 3.2 (LIRR $\Rightarrow$ CLR-OWR) . . . 49
N.2 Lemma 4.1 (Hardness Properties of LIRR) . . . 50
N.3 Lemma 4.5 (Homomorphic Relation) . . . 51
N.4 Lemma 4.6 (Rerandomization) . . . 51
N.5 Lemma 4.8 (LoC-NM $\Rightarrow$ LIRR) . . . 51
N.6 Theorem 7.2 (Leakage of Refreshing) . . . 52
N.6.1 Proof Intuition . . . 52
N.6.2 Formal Proof . . . 52

1 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 ℓ 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 *ℓ-continuous-leakage-resilient (CLR) one-way relation (OWR)* (ℓ-CLR-OWR; see Definition 2.3). More generally, we say that a cryptographic primitive is ℓ-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, 42, 54], 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 "bounded-leakage memory attacks" [1, 49, 3, 46, 2], do not restrict the *type* of leakage that the adversary can see, but have some a-priori upper bound ℓ on the *overall amount* of leakage. They do not offer a method for refreshing the secret-key. In contrast, current "continuous-leakage" models [13, 43, 48, 26, 50, 27] use secret-key evolution, allowing them to only bound 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 [48, 26, 50, 27] is the "only computation leaks information" (OCLI) model of Micali and Reyzin [48], 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 [31], 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 give an extended overview of leakage-resilient cryptography in Appendix A, and concentrate on only the most relevant works here. In the model of "bounded-leakage memory attacks", the most relevant schemes are the constructions of signatures, ID schemes and AKA protocols of [3, 46, 20]. As we shall see in Section 6, we will use the ideas from these works to construct advanced CLR primitives from a CLR-OWR (while constructing the latter is the bulk of this work). The work of [3] also showed a simple and general method for refreshing secret-keys with the aid of a "master refreshing key", which must be stored externally 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 only requires fresh randomness, and no additional secrets, during refreshing.

The work of [27] showed that a variant of the standard "tree-based" construction of many-time signatures from one-time signatures is secure in the "only-computation leaks information model", assuming the one-time signature scheme is secure against memory attacks (albeit, against 3 signing queries). Unfortunately, it is insecure against continuous memory attacks, since the component keys of the one-time signatures are kept in memory for many periods (even though they are not accessed), and hence can completely leak in the memory attack model.

Concurrent Work. Subsequent to our work, Brakerski et al. [11] 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 assumptions (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 3-5). We do so in a modular manner. First, in Section 3, we show how to reduce the problem of analyzing *continuous leakage* for OWR to that of only analyzing *bounded-leakage*, albeit in a more complicated primitive, which we call a *leakage-indistinguishable re-randomizable relations* (LIRR). Then, in Section 4 we show how to build LIRR generically from group-homomorphic encryptions and NIZKs, where the latter is a new notion of possibly independent interest. In addition to the encryption being homomorphic, we need it to have a certain (non-standard) bounded-leakage-resilience property, which we call *leakage-of-ciphertext non-malleability (LoC-NM)*. Finally, in Section 5 we instantiate the required homomorphic components efficiently using the K-Linear assumption in bi-linear groups. First, we show how build to *homomorphic* LoC-NM encryption, by generalizing the "Cramer-Shoup-Lite" CCA-1 encryption scheme [17]. Second, we notice that the efficient non-interactive zero-knowledge (NIZK) argument system of Groth-Sahai [38] is actually homomorphic, and can be used as the last missing block to instantiate our construction efficiently. This is the only place where we are currently "stuck" with the K-Linear assumption. In Section 6, we then show how to leverage a CLR-OWR to get other more interesting CLR primitives: signatures, ID schemes, and AKA protocols.

Extensions. Finally, in Section 7 we mention several important extensions of our results to even *stronger* models/notions of CLR security. Most importantly, in our basic model we only consider leakage on the *secret-keys* stored in memory, while in practice the adversary can also leak from the randomness of the key-refreshing and the signing algorithms. We show that our schemes can allow (at least some) leakage from these algorithms as well. For the randomness of signing, we show an alternate construction of signatures in the random-oracle model (along the lines of [3, 46]) that allows a large amount of the signing randomness to leak. For randomness of refreshing, we show that (generically) all CLR schemes (one-way relations, signatures, encryption) also allow a small amount of randomness (logarithmic in the security parameter) to leak as well. Lastly, we also show additional extensions to "concurrent leakage" and "noisy leakage" [49].

2 Continuous Leakage Resilience for One-Way Relations (CLR-OWR)

A one-way relation (OWR) consists of two efficient algorithms: a key-generation algorithm $(pk, sk) \leftarrow \texttt{KeyGen}(1^\lambda)$, and a verification algorithm $\texttt{Ver}(pk, sk)$, which decides whether a secret-key sk *is valid for* the public-key pk.

Definition 2.1 (One-Way Relation (OWR)). *We say that* (KeyGen, Ver) *is a* one-way relation *if it satisfies:*

Correctness: *If* $(pk, sk) \leftarrow \texttt{KeyGen}(1^\lambda)$*, then* $\texttt{Ver}(pk, sk) = 1$.

Security: *For any PPT attacker* $\mathcal{A}$*, we have* $\Pr\left[\texttt{Ver}(pk, sk^*) = 1 \;\middle|\; \begin{array}{c}(pk, sk) \leftarrow \texttt{KeyGen}(1^\lambda)\\ sk^* \leftarrow \mathcal{A}(pk)\end{array}\right] \leq \mathsf{negl}(\lambda)$.

A one-way relation generalizes the concept of a one-way function (OWF). Of course, we can always set sk to include all of the randomness of the KeyGen algorithm, so that $pk = \texttt{KeyGen}(sk)$ *is* a OWF. However, when defining leakage-resilient one-wayness (which we do next), this equivalence 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 2.2 (Leakage Oracle). *A leakage oracle $\mathcal{O}_{sk}^{\lambda}(\cdot)$ is parameterized by a secret key sk and a security parameter λ. 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 ℓ on the number of queries that the adversary can make during various stages of the attack.

Bounded-Leakage Resilient OWR. An ℓ-leakage-resilient OWR (ℓ-LR-OWR) is defined by modifying Definition 2.1 so that the adversary can make up to ℓ queries to the oracle $\mathcal{O}_{sk}^{\lambda}(\cdot)$, after seing pk and before outputting sk^*. It was shown in [4, 20] (and implicitly in [3, 46]) that any second-preimage resistant[1] (SPR) function F automatically gives an ℓ-LR-OWR, where ℓ 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 ℓ-bits of leakage on x, then x still has entropy. Therefore, if the adversary outputs x' such that $F(x') = y$, then, with overwhelming 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 $\texttt{KeyGen}$ and $\texttt{Ver}$ as before, but also includes a re-randomization algorithm $sk' \leftarrow \texttt{ReRand}_{pk}(sk)$, 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 ℓ bits of leakage on each of arbitrarily many re-randomized secret keys, and still be unable to produce a valid secret key herself.

Definition 2.3 (CLR-OWR). *We say that a scheme* $(\texttt{KeyGen}, \texttt{ReRand}, \texttt{Ver})$ *is an* ℓ-continuous-leakage-resilient (ℓ-CLR) one-way relation *if it satisfies the following correctness and security properties.*

Correctness: *For any polynomial $q = q(\lambda)$, if we sample $(pk, sk) \leftarrow \texttt{KeyGen}(1^\lambda)$,
$sk_1 \leftarrow \texttt{ReRand}(sk), \ldots, sk_q \leftarrow \texttt{ReRand}(sk_{q-1})$, then, with overwhelming probability (w.o.p.),
$\texttt{Ver}(pk, sk) = \texttt{Ver}(pk, sk_1) = \ldots = \texttt{Ver}(pk, sk_q) = 1$.*

Security: *For any PPT adversary $\mathcal{A}$, we have* $\Pr[\mathcal{A}$ *wins* $] \leq \mathsf{negl}(\lambda)$ *in the following game:*

- *The challenger chooses $(pk, sk) \leftarrow \texttt{KeyGen}(1^\lambda)$ and gives pk to $\mathcal{A}$.*
- *The adversary $\mathcal{A}$ runs for arbitrarily many* leakage rounds. *In each round:*
  - *The adversary makes up to ℓ calls to a leakage-oracle $\mathcal{O}_{sk}^{\lambda}(\cdot)$, with the current secret key sk.*[2]
  - *At the end of the round, the challenger samples $sk' \leftarrow \texttt{ReRand}(sk)$ and updates $sk := sk'$.*
- *The adversary* wins *if it produces a value sk^* such that $\texttt{Ver}(pk, sk^*) = 1$.*

Why Prior LR Techniques Fail for CLR. All of the prior works on bounded-leakage 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*

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

[2] This is equivalent to a definition where, in each round, the adversary asks for a single leakage-function with ℓ-bit output length.

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.

3 CLR-OWR Part I: Continuous Leakage from Bounded 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 *bounded-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 ℓ 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 ($\mathtt{Setup}$, $\mathtt{SampG}$, $\mathtt{SampB}$, $\mathtt{ReRand}$, $\mathtt{Ver}$, $\mathtt{isGood}$) 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 sam_G, a "bad" sampling key 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 3.1 (LIRR). *We say that the scheme* ($\mathtt{Setup}$, $\mathtt{SampG}$, $\mathtt{SampB}$, $\mathtt{ReRand}$, $\mathtt{Ver}$, $\mathtt{isGood}$) *is an* ℓ-leakage-indistinguishable re-randomizable relation *(*ℓ-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.*

$$\mathtt{Ver}(pk, sk_G) = 1,\ \mathtt{isGood}(pk, sk_G, dk) = 1 \quad , \quad \mathtt{Ver}(pk, sk_B) = 1,\ \mathtt{isGood}(pk, sk_B, dk) = 0$$

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:*

$$(pk, \mathsf{sam}_G, \mathsf{sam}_B, dk) \leftarrow \mathtt{Setup}(1^\lambda), \quad sk_0 \leftarrow \mathtt{SampG}(\mathsf{sam}_G), \quad sk_1 \leftarrow \mathtt{SampG}(\mathsf{sam}_G), \quad sk_1' \leftarrow \mathtt{ReRand}(sk_0)$$

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

$$\Pr\left[\begin{array}{c} \mathtt{Ver}(pk, sk^*) = 1 \\ \mathtt{isGood}(pk, sk^*, dk) = 0 \end{array} \middle| \begin{array}{c} (pk, \mathsf{sam}_G, \mathsf{sam}_B, dk) \leftarrow \mathtt{Setup}(1^\lambda) \\ sk^* \leftarrow \mathcal{A}(pk, \mathsf{sam}_G) \end{array}\right] \le \mathsf{negl}(\lambda)$$

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

$$\Pr\left[\mathtt{isGood}(pk, sk^*, dk) = 1 \middle| \begin{array}{c} (pk, \mathsf{sam}_G, \mathsf{sam}_B, dk) \leftarrow \mathtt{Setup}(1^\lambda) \\ sk^* \leftarrow \mathcal{A}(pk, \mathsf{sam}_B) \end{array}\right] \le \mathsf{negl}(\lambda)$$

ℓ-Leakage-Indistinguishability: *Given both sampling keys* $\mathsf{sam}_G, \mathsf{sam}_B$*, and* ℓ *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* $\left|\Pr[\mathcal{A} \text{ wins }] - \frac{1}{2}\right| \le \mathsf{negl}(\lambda)$ *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 a random bit* $b \leftarrow \{0,1\}$. *If* $b = 1$ *then it samples* $sk \leftarrow \texttt{SampG}(\mathsf{sam}_G)$, *and otherwise it samples* $sk \leftarrow \texttt{SampB}(\mathsf{sam}_B)$.
- *The adversary* $\mathcal{A}$ *can make up to* ℓ *queries* in total *to the leakage-oracle* $\mathcal{O}^\lambda_{sk}(\cdot)$.
- *The adversary outputs* sk^* *and wins if* $\texttt{isGood}(pk, sk^*, dk) = b$.

An ℓ-LIRR can be used to construct an ℓ-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 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 sam_G is only used as an intermediate step during key-generation to sample the secret-key sk, but is never explicitly stored afterwards.

$\texttt{KeyGen}(1^\lambda)$: 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 an LIRR ($\texttt{Setup}$, $\texttt{SampG}$, $\texttt{SampB}$, $\texttt{ReRand}$, $\texttt{Ver}$, $\texttt{isGood}$)

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 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 *ℓ-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. A formal proof of the following theorem appears in Appendix N.1.

Theorem 3.2. *Given any ℓ-LIRR scheme, the construction in Figure 1 is a secure ℓ-CLR-OWR.*

4 CLR-OWR Part II: Constructing LIRR from Generic Components

4.1 Syntax of Construction and Hardness Properties

We now instantiate an ℓ-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, 52, 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:

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

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

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:

$$R_{eq} \overset{\text{def}}{=} \left\{ (y, x) \;\middle|\; y = (pk_1, pk_2, c_1, c_2),\; x = (m, r_1, r_2) \text{ s.t. } c_1 = \mathtt{Enc}^1_{pk_1}(m; r_1),\; c_2 = \mathtt{Enc}^2_{pk_2}(m; r_2) \right\}.$$

The corresponding language L_{eq}, is that of ciphertext pairs that encrypt the same plaintext. Let $\Pi = (\mathtt{Setup}^\Pi, \mathtt{Prov}^\Pi, \mathtt{Ver}^\Pi, \mathtt{Sim}^\Pi)$ 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* $\mathsf{prms} \leftarrow \mathtt{ParamGen}(1^\lambda)$ (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 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, π). In a good secret-key, c_2 is a new random encryption of m under $\mathcal{E}_2$, and π is a proof of plaintext-equality. In a bad secret-key, c_2 is just an encryption of the message 1, and π is a simulated proof. The verification procedure just checks the proof π against the statement (c_1, c_2).

$\mathtt{Setup}(1^\lambda)$: Output $pk = (\mathsf{prms}, \text{CRS}, pk_1, pk_2, c_1)$, $\mathsf{sam}_G = (m, r_1)$, $\mathsf{sam}_B = \text{TK}$, $dk = (sk_1, sk_2)$ where:

$$\mathsf{prms} \leftarrow \mathtt{ParamGen}(1^\lambda),\quad (pk_1, sk_1) \leftarrow \mathtt{KeyGen}^1(\mathsf{prms}),\quad (pk_2, sk_2) \leftarrow \mathtt{KeyGen}^2(\mathsf{prms})$$
$$m \leftarrow \mathcal{M},\quad c_1 \leftarrow \mathtt{Enc}^1_{pk_1}(m; r_1),\quad (\text{CRS}, \text{TK}) \leftarrow \mathtt{Setup}^\Pi(\mathsf{prms})$$

$\mathtt{SampG}(\mathsf{sam}_G)$: Output $sk_G = (c_2, \pi)$ where: $c_2 \leftarrow \mathtt{Enc}^2_{pk_2}(m; r_2), \pi \leftarrow \mathtt{Prov}^\Pi((c_1, c_2), (m, r_1, r_2))$.

$\mathtt{SampB}(\mathsf{sam}_B)$: Output $sk_B = (c_2, \pi)$ where: $c_2 \leftarrow \mathtt{Enc}^2_{pk_2}(1)$, $\pi \leftarrow \mathtt{Sim}^\Pi((c_1, c_2), \text{TK})$.

$\mathtt{Ver}(pk, sk)$: Parse $sk = (c_2, \pi)$ and output $\mathtt{Ver}^\Pi((c_1, c_2), \pi)$.

$\mathtt{isGood}(pk, sk, dk)$: Parse $sk = (c_2, \pi), dk = (sk_1, sk_2)$. Output 1 iff $\mathtt{Dec}^1_{sk_1}(c_1) = \mathtt{Dec}^2_{sk_2}(c_2)$.

Figure 2: Construction of LIRR: syntax of procedures (except for re-randomization).

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 Appendix C), which is weaker than semantic-security and only requires that an encryption of a random message is hard to invert. The proof of the following lemma appears in Appendix N.2.

Lemma 4.1. *Assume that* $\mathcal{E}_1$ *is a one-way secure and* Π *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 3.1).*

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 $\mathcal{E}_1, \mathcal{E}_2, \Pi$.

4.2 Re-randomization Using Homomorphic Encryptions and 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, π) where π is a proof of plaintext-equality for the statement (c_1, c_2). Therefore, 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 π 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 Π 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 4.2 (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 4.3 (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 4.4 (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 Π 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 4.5. *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 Π 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 4.5, and that Π 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.

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

Figure 3: Re-randomization

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 π in the 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 π with a random proof π' of the true statement $(1_{\mathcal{C}_1}, c') \in L_{eq}$. The following lemma is proved in Appendix N.4.

Lemma 4.6. *The re-randomization method in Figure 3 satisfies the* re-randomization of LIRR *(Definition 3.1).*

4.3 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 ℓ bits of leakage on a *ciphertext* c, the adversary cannot produce a *related ciphertext* c^*.

Definition 4.7 (Leakage-of-Ciphertext Non-Malleable Encryption)**.** *A public-key encryption scheme* $\mathcal{E} = (\texttt{KeyGen}, \texttt{Enc}, \texttt{Dec})$ *is* ℓ-leakage-of-ciphertext non-malleable (ℓ-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* $(pk, sk) \leftarrow \texttt{KeyGen}(1^\lambda)$ *and gives* pk *to* $\mathcal{A}$*.*
- *The adversary* $\mathcal{A}$ *chooses two messages:* m_0, m_1 *and gives these to the challenger.*
- *The challenger chooses a bit* $b \leftarrow \{0,1\}$ *and computes* $c \leftarrow \texttt{Enc}_{pk}(m_b)$*, but does not give* c *to* $\mathcal{A}$*. Instead, the adversary* $\mathcal{A}$ *can make up to* ℓ *queries to a leakage-oracle* $\mathcal{O}_c^\lambda(\cdot)$*, on the* ciphertext c*.*
- *The adversary* $\mathcal{A}$ *chooses a ciphertext* c^* *and the challenger gives* $m^* = \texttt{Dec}_{sk}(c^*)$ *to* $\mathcal{A}$*.*
- *The adversary* $\mathcal{A}$ *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 [24] implies ℓ-LoC-NM security where ℓ approaches the message-size of the scheme. This is because an adversary that leaks less than ℓ 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 ℓ-LoC-NM, then the construction satisfies ℓ-leakage-indistinguishability (Definition 3.1). 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". The proof of the following lemma appears in Appendix N.5.

Lemma 4.8. *Assume that, in the construction of Figure 2, the scheme* $\mathcal{E}_2$ *is* ℓ*-LoC-NM and that* Π *is a secure NIZK. Then the construction satisfies the leakage-indistinguishability property of LIRR (Definition 3.1).*

4.4 Summary of Construction and its Requirements

The following theorem follows directly from Lemmata 4.1, 4.6 and 4.8 and Theorem 3.2.

Theorem 4.9. *The construction in Figure 2, with the re-randomization procedure in Figure 3, is a secure* ℓ*-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* ℓ*-LoC-NM secure.*
- Π *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* ℓ*-CLR-OWR.*

A stripped-down construction of (just) the CLR-OWR from the schemes $\mathcal{E}_1, \mathcal{E}_2, \Pi$, without the additional algorithms needed for LIRR, is shown in Figure 4.

$\texttt{KeyGen}(1^\lambda)$**:** Output $pk = (\mathsf{prms}, \textsc{crs}, pk_1, pk_2, c_1)$, $sk = (c_2, \pi)$ where:

$$\mathsf{prms} \leftarrow \texttt{ParamGen}(1^\lambda),\quad (pk_1, \cdot) \leftarrow \texttt{KeyGen}^1(\mathsf{prms}),\ (pk_2, \cdot) \leftarrow \texttt{KeyGen}^2(\mathsf{prms}),\ (\textsc{crs}, \cdot) \leftarrow \texttt{Setup}^\Pi(\mathsf{prms})$$
$$m \leftarrow \mathcal{M},\quad c_1 \leftarrow \texttt{Enc}^1_{pk_1}(m; r_1),\ c_2 \leftarrow \texttt{Enc}^2_{pk_2}(m; r_2), \pi \leftarrow \texttt{Prov}^\Pi((c_1, c_2), (m, r_1, r_2))$$

$\texttt{Ver}(pk, sk)$**:** Parse $sk = (c_2, \pi)$ and output $\texttt{Ver}^\Pi((c_1, c_2), \pi)$.
$\texttt{ReRand}(sk)$**:** Parse $sk = (c_2, \pi)$. Choose $(r'_2, r'_3) \leftarrow \mathcal{R}_2 \times \mathcal{R}_3$.
Set $c'_2 := \texttt{Enc}^2_{pk_2}(1_\mathcal{M}; r'_2)$, $\pi' := \texttt{Prov}((1_{\mathcal{C}_1}, c'_2), (1_\mathcal{M}, 0_{\mathcal{R}_1}, r'_2); r'_3)$. Output $sk^* = (c_2 \cdot c'_2, \pi \cdot \pi')$.

Figure 4: CLR-OWR from Components $\mathcal{E}_1, \mathcal{E}_2, \Pi$

5 CLR-OWR Part III: 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 4.9. 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, 42, 54]. 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 just the DDH assumption (in disguise). Also, K-linear implies $(K + 1)$-linear, so the 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 Γ. The map $e : \mathbb{G} \times \Gamma \to \mathbb{G}_T$ is an efficiently computable *bilinear map*, which satisfies: (1) *non-degeneracy*: the element $e(\mathbf{g}, \boldsymbol{\gamma})$ generates $\mathbb{G}_T$, and (2) *bilinearity*: 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 Π assuming the K-linear assumption, for any constant $K \geq 1$, holds in *both base groups* $\mathbb{G}$ and Γ 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 Γ. This is also sometimes called the *External Diffie-Hellman (SXDH)* assumption [53, 9, 5, 30, 55]. 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 Appendix E. 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 Π will.

5.1 The encryption schemes $\mathcal{E}_1, \mathcal{E}_2$

For the encryption scheme $\mathcal{E}_1$, we can use any homomorphic one-way secure encryption. We simply choose to use (a slight variant of) the ElGamal encryption scheme, shown in Figure 5.

For the scheme $\mathcal{E}_2$, we need a homomorphic encryption satisfying ℓ-LoC-NM security. We use a variant of the "Cramer-Shoup Lite" (CS-Lite) [16] scheme shown in Figure 6. We implicitly show that ℓ-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.

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

Figure 5: ElGamal Encryption (variant)

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 $\mathbf{h}_0 := \mathbf{g}_0^{x_{0,0}} \mathbf{g}_1^{x_{0,1}},\quad \mathbf{h}_1 := \mathbf{g}_0^{x_{1,0}} \mathbf{g}_1^{x_{1,1}}\quad, \ldots,\quad \mathbf{h}_n := \mathbf{g}_0^{x_{n,0}} \mathbf{g}_1^{x_{n,1}}$.
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 $\tilde{\mathbf{c}}_1 \overset{?}{=} \mathbf{c}_1, \ldots, \tilde{\mathbf{c}}_n \overset{?}{=} \mathbf{c}_n$, then output $\mathbf{z}/\tilde{\mathbf{c}}_0$. Else output $\perp$.

Figure 6: Multiple CS-Lite Scheme

For $n = 0$, the scheme above is already a semantically-secure encryption scheme under the DDH assumption. For $n = 1$, 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.[3] 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 ℓ-LoC-NM secure, where the leakage-bounds ℓ 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 $\tilde{\mathbf{c}}_i$.[4] However, since the adversary only sees ℓ 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 5.1. *Under the DDH assumption, the multiple CS-Lite scheme (Figure 6) with parameter n, is an ℓ-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 $\mathcal{C} = \mathbb{G}^{n+3}$.*

The proof of the above theorem appears in Appendix E, 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.

[3]The main difference between the CS-Lite scheme and the full Cramer-Shoup scheme (which is CCA-2 secure) is that the lite scheme implicitly only uses a "1-universal hash proof system" to compute the verification-element in the ciphertext, whereas the full scheme uses a more complicated "2-universal hash proof system". Of course, sice the full CS scheme is CCA-2 secure, it *cannot* be homomorphic.

[4]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.

5.2 The NIZK System Π : Groth-Sahai Proofs for Linear Equations

We consider the language of *satisfiable systems of linear equations*, over some 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 $\vec{\mathbf{c}} \in \mathbb{G}^M$:

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

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 $R^{\mathbf{B}}_{linear} = \{(\vec{\mathbf{c}}, \vec{x}) \ : \ ((\mathbf{B}, \vec{\mathbf{c}}), \vec{x}) \in R_{linear}\}$, then $R^{\mathbf{B}}_{linear}$ is a *homomorphic relation.* For simplicity, we will say that a NIZK argument system for R_{linear} is homomorphic if, for every fixed choice of $\mathbf{B}$, it is homomorphic for the (homomorphic) relations $R^{\mathbf{B}}_{linear}$.

The Groth-Sahai (GS) [38] 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 Appendix F. 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 [38] (see Appendix F for a generalization to the K-linear assumption).

Lemma 5.2. *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}. For any statement consisting of M linear equation in N unknowns, the corresponding proof π contains $2N + M$ group elements.*

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 5 and Figure 6, respectively. We now show that the corresponding language for plaintext-equality L_{eq} can be expressed in terms of a satisfiable set of linear equations.

Let $(p, \mathbb{G}, \Gamma, \mathbb{G}_T, e, \mathbf{g}_0, \gamma) \leftarrow \mathcal{G}_{pair}(1^\lambda)$ and $\mathbf{g}_1 \leftarrow \mathbb{G}$ be the common system parameters. Let pk_1, c_1 be a public-key and ciphertext of $\mathcal{E}_1$ and pk_2, c_2 be a public-key and ciphertext of $\mathcal{E}_2$ so that:

$$\begin{array}{ll} pk_1 = \mathbf{f} \in \mathbb{G}, & c_1 = (\mathbf{z}_1, \mathbf{c}_1) \in \mathbb{G}^2 \\ pk_2 = (\mathbf{h}_0, \mathbf{h}_1, \dots, \mathbf{h}_n) \in \mathbb{G}^{n+1} & c_2 = (\mathbf{y}_0, \mathbf{y}_1, \mathbf{z}_2, \mathbf{c}_{2,1}, \dots, \mathbf{c}_{2,n}) \in \mathbb{G}^{(n+3)}. \end{array}$$

Then $(c_1, c_2) \in L_{eq}$ if and only if there exist a vector $\vec{w} = (r_1, r_2) \in \mathbb{Z}_p^2$ such that:

$$\mathbf{f}^{r_1} = \mathbf{c}_1, \quad \mathbf{h}_1^{r_2} = \mathbf{c}_{2,1}, \quad \dots, \quad \mathbf{h}_n^{r_2} = \mathbf{c}_{2,n}, \quad \mathbf{g}_0^{r_2} = \mathbf{y}_0, \quad \mathbf{g}_1^{r_2} = \mathbf{y}_1, \text{ and } \quad \mathbf{h}_0^{r_2}(\mathbf{g}_0^{-1})^{r_1} = \mathbf{z}_2/\mathbf{z}_1.$$

We notice that the message $\mathbf{m}$ completely drops out, and the witness $\vec{w}$ consists only of the randomness of the two encryptions. The above system consists of $M = n + 4$ equations in $N = 2$ unknowns. Moreover, notice that only the right side of the above equations depend on the ciphertexts c_1, c_2, while the coefficients on the left are completely fixed by the public-parameters prms and the public-keys pk_1, pk_2. Therefore, for each choice of pk_1, pk_2, there exists a matrix $\mathbf{B}$ such that $R_{eq}^{(pk_1,pk_2)} = R^{\mathbf{B}}_{linear}$. This, along with the preceding paragraph, shows that the GS scheme is a homomorphic NIZK for R_{eq}, where the proofs contain $2N + M = n + 8$ group elements.

5.3 Efficiency and Relative-Leakage of Instantiation with $\mathcal{E}_1, \mathcal{E}_2, \Pi$

By plugging in the schemes $\mathcal{E}_1, \mathcal{E}_2, \Pi$ from the previous sections (based on DDH) into our abstract construction (Theorem 4.9) we get a concrete CLR-OWR instantiation. Let us briefly look at its parameters. The secret-key sk consists of a ciphertext c_2 with $n+3$ group elements, and a proof π with $n+8$ group elements, for a total of $2n+11$ group elements. The leakage, on the other hand, is $\ell = n\log(p) - \lambda$ *bits*. Assuming each group element can be represented optimally using $\log(p)$ bits, we therefore get relative leakage which approaches $\frac{\ell}{|sk|} \approx \frac{1}{2}$, as n grows.

We generalize the schemes $\mathcal{E}_1, \mathcal{E}_2$ and the NIZK system Π to the K-linear assumption in Appendix E. If K is constant, the ciphertext c_2 still consists of $n+O(1)$ group elements. However, the proof π grows linearly with K, and requires $Kn + O(1)$ group elements. Therefore, as K grows, the relative leakage of the scheme degrades, and only approaches $\frac{1}{K+1}$. 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. See Appendix G for a proof of the following theorem.

Theorem 5.3. *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 ℓ-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.*

6 Applications: Signatures, ID Schemes, and 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 Katz and Vaikuntanathan [46] and Alwen, Dodis and Wichs [3], which (sometimes implicitly) construct bounded-leakage-resilient versions of these primitives, from underlying bounded-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 ℓ-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 ℓ 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 [20], which slightly generalizes the original scheme of [46]. We start with a CLR-OWR $\mathcal{C} = (\texttt{KeyGen}^{\mathcal{C}}, \texttt{ReRand}, \texttt{Ver}^{\mathcal{C}})$, and a NIZK $\Psi = (\texttt{Setup}^{\Psi}, \texttt{Prov}^{\Psi}, \texttt{Ver}^{\Psi})$ for the relation

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

The signature scheme construction is shown in Figure 7. For security, we need the NIZK system Ψ 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 ψ^* for a new statement y^*, then there is a way to *extract* a valid witness x^* from ψ^*, 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 [20], tSE NIZKs can be constructed by either (1) composing a *simulation-sound* NIZK with a CPA-secure encryption, yielding the scheme of [46], 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, 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 σ^* on a new message m^*. See Appendix I for the formal definitions of CLR secure signatures, tSE NIZKs and a proof of security for the above construction.

```
KeyGen(1^λ): Run $(pk, sk) \leftarrow \texttt{KeyGen}^{\mathcal{C}}(1^\lambda)$,  $(\textsc{crs}, \cdot) \leftarrow \texttt{Setup}^{\Psi}(1^\lambda)$. Output: $vk := (pk, \textsc{crs})$, $sk$.
Sign_sk(m): Output $\sigma \leftarrow \texttt{Prov}^{\Psi}((pk, m), sk)$.
SigVer_vk(m, σ): Output $\texttt{Ver}^{\Psi}((pk, m), \sigma)$.
ReRand(sk): Run the re-randomization procedure of the CLR-OWR $\mathcal{C}$.
```

Figure 7: A CLR-Signature Scheme from a CLR-OWR and a tSE NIZK

In Appendix J, we also show that, amazingly, 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 Appendix K. We give black-box constructions of such schemes from CLR signatures, naturally extending the prior constructions of [3, 20] in the bounded-leakage 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 construction of signatures, ID schemes, and AKA protocols uses the same secret-key as the underlying CLR-OWR, and preserves the absolute and relative leakage of the underlying CLR-OWR. Therefore, as a corollary of our construction of CLR-OWR under the K-linear assumption, and our constructions of ID/Signatures/AKA from CLR-OWR, we get the following result.

Theorem 6.1. *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 ℓ-CLR Signatures, ID schemes, and AKA protocols, with relative-leakage $\frac{\ell}{|sk|} \geq \frac{1}{K+1} - \epsilon$. Moreover, the schemes are efficient, with public-keys, secret-keys, communication/signatures each consisting of $O(1)$ group elements and all algorithms using at most $O(1)$ group operations.*

The proof of the above theorem is divided into Appendix I (for existence of signatures), Appendix J (for the efficiency of signatures) and Appendix K (for ID schemes and AKA).

7 Extensions

In this section, we show several important extensions of our results to *stronger* notions of leakage-resilient security.

7.1 Leakage of Computation

When considering ID schemes, signatures and AKA, the secret key sk is not only stored in memory between refreshes, but also *computed upon* during the creation of a signature or the execution of the ID scheme, or AKA protocol. Taking signatures as an example, the work of Katz and Vaikuntanathan [46] already noticed that, even in the case of bounded-leakage, resilience to key-leakage attacks where the leakage functions only get sk as an input, does not necessarily guarantee security if the adversary can *also* leak the local randomness used by the signing algorithm. Following [46], we define a stronger notion of *CLR with leakage-of-computation security*, where the adversarial leakage functions in each round get the local randomness of the scheme as an input, in addition to the secret-key. In particular, in each leakage round i, the adversary can jointly leak on the current secret key sk_i, as well as on all the local-randomness r_i used to run all of the computation (signing, protocol execution of ID, AKA) during that round.

Unfortunately, as originally noted by Katz and Vaikuntanathan [46] in the case of bounded-leakage-resilience, the NIZK-based construction of signatures does *not* (in general) achieve leakage-of-computation security. On the other hand, [46] gave alternative constructions of signatures based on Σ-protocols for second-preimage resistent (SPR) hash functions (concurrently proposed by [3]), and showed that these do achieve leakage-of-computation security, albeit in the random-oracle model. A priori, it is not clear if these results extend to continuous leakage-resilience, since our constructions are *not* based on SPR functions. In Appendix M, we generalize the results of [46, 3] in several important respects, and show that they *indeed can* apply to our construction. In particular, we show *general* constructions of ID schemes (in the standard model), signatures and AKA protocols (in the random oracle model) with leakage-of-computation security. We summarize the results of Appendix M in the following theorem:

Theorem 7.1. *Fix any constant $K \geq 1$, and assume the K-linear assumption holds in the base group(s) of a pairing $\mathcal{G}_{pair}$. Let $\epsilon > 0$ be a constant, and let $\alpha = \frac{1}{K+1} - \epsilon$. The following primitives can be instantiated with* leakage-of-computation *security:*

1. *ℓ-CLR ID schemes with relative-leakage α, in the standard model.*
2. *ℓ-CLR AKA with relative-leakage α, in the random-oracle model.*
3. *ℓ-CLR signatures with relative-leakage $\alpha/2$, in the random-oracle model.*
4. *ℓ-CLR "challenge-response" interactive signatures with relative-leakage α, in the random-oracle model.*

The public-keys, secret-keys, communication, and signature size of all of these schemes consists of $O(1)$ group-elemets/exponents and all of the algorithms of the schemes require at most $O(1)$ exponentiation/pairing operations.

It remain an important open problem to construct even bounded-leakage-resilient signature schemes with leakage-of-computation security in the *standard model.*

7.2 Leakage of Refreshing

In our basic model of CLR security, we only consider leakage on the secret key *sk in-between* the refresh operations. This is already sufficient if we imagine the refreshing operation to be performed in a secure setting when no leakage occurs. However, if the adversary may also be able to leak information about the random coins of the refreshing process itself, it is not clear if security is preserved. We can consider the stronger notion of (ℓ, μ)-CLR security, where the adversary can leak up to ℓ bits of information on each secret key sk_i *in-between* refresh operations, and up to μ-bits of information on the internal state (sk_i, r_i) used *during* each refresh operation $sk_{i+1} = \mathtt{ReRand}(sk_i; r_i)$. More precisely, the adversary can learn ℓ_i bits on any secret key sk_i alone, and μ_i bits jointly on (sk_i, r_i), subject to the constraints $\mu_i \leq \mu$ and $\ell_{i+1} + \mu_i \leq \ell$ (that is, if the adversary chooses to leak many bits on the refreshing, she is constrained to leaking fewer bits on the next key). Brakerski et al. [11] show that their particular CLR PKE scheme is secure w.r.t. logarithmic (in the security parameter) leakage $\mu(\lambda) = O(\log(\lambda))$ of the refreshing randomness. It was observed by Brent Waters (personal communication [56]) that this, in-fact, holds *generically* for all CLR OWRs, Signatures, and Public-Key Encryption (PKE) schemes. It also carries over to our ID schemes and AKA protocols, which are constructed in a generic manner from signatures. For completeness, we formalize this observation in the following theorem. The proof-intuition as well as a formal proof are given in Appendix N.6.

Theorem 7.2. *Assume that a construction of a CLR-OWR (respectively: Signature, PKE) is $(\ell(\lambda), 0)$-CLR secure, and let $\mu(\lambda)$ be any function with $\mu(\lambda) = O(\log(\lambda))$. Then the* same *construction is also $(\ell(\lambda), \mu(\lambda))$-CLR secure. More generally, if the original construction is $(\ell(\lambda), 0)$-secure even against leakage-functions running in time $t(\lambda)\mathsf{poly}(\lambda)$ for some $t(\lambda)$, then the above holds for $\mu(\lambda) = \log(t(\lambda))$.*

Corollary 7.3. *Fix any constant $K \geq 1$, and assume the K-linear assumption holds in the base group(s) of a pairing $\mathcal{G}_{pair}$. Let $\epsilon > 0$ be any constant. Then, for any polynomial $\ell(\cdot)$ and any $\mu(\lambda) = O(\log(\lambda))$ there exist $(\ell(\lambda), \mu(\lambda))$-CLR OWR, Signatures, ID schemes, and AKA protocols with $\ell(\lambda)/|sk(\lambda)| = \frac{1}{K+1} - \epsilon$. Further, if we assume that the K-linear assumption is secure against adversaries running in time $t(\lambda)\mathsf{poly}(\lambda)$, then the above also holds for $\mu(\lambda) = \log(t(\lambda))$.*

It remains an important open problem to allow for super-logarithmic leakage of the refreshing randomness without making computational assumptions against super-polynomial attackers.

7.3 Noisy Leakage

In our original definition, we model leakage attacks by allowing the adversary to compute arbitrary efficient functions of the secret-key, as long as the *output lengths* of such functions are bounded. We call this *length-bounded leakage*. We now define a generalization of length-bounded leakage, called *noisy leakage*, where the adversary can learn functions with arbitrarily large output-lengths, as long as the entropy of the secret-key does not decrease significantly. We define the leakiness of a function h as follows:[5]

Definition 7.4 (ℓ-leaky function)**.** *A probabilistic leakage function $h : \{0,1\}^* \to \{0,1\}^*$ is* ℓ-leaky *if, for all $n \in \mathbb{N}$, we have $\widetilde{\mathbf{H}}_\infty(U_n \mid h(U_n)) \geq n - \ell$, where U_n is the uniform distribution over $\{0,1\}^n$.*

It is easy to show that any function whose output-length is bounded by ℓ-bits, is also ℓ-leaky. Therefore, noisy leakage provides a generalization of length-bounded leakage. We note that a similar definition of noisy leakage already appeared in [49], but the definition of "leakiness" there depended on the scheme being attacked. In contrast, our definition is independent of the scheme and hence seems easier to work with. Moreover, we show that it composes nicely so that the concatenation of an ℓ_1-leaky function and an ℓ_2-leaky function is $\ell_1 + \ell_2$ leaky. In practice, it may often be the case that physical leakage (say a read-out of electromagnetic radiation measurements) observed during a side-channel attack satisfies the noisy definition, even though it has a long description, which is not efficiently compressible, and hence does not satisfy the length-bounded definition.

In Appendix L, we show that our concrete instantiation of a CLR-OWR, based on the K-linear assumption, is secure with respect to noisy leakage. In particular, in each leakage-round, the adversary can adaptively ask for arbitrary leakage-functions h_i of the current secret-key, as long as each h_i is ℓ_i-leaky and the total leakage is at most $\sum \ell_i \leq \ell$. Moreover, we show that our constructions of signatures, ID schemes and AKA preserve noisy-leakage resilience.

Theorem 7.5. *The statement of Theorem 6.1 also holds w.r.t.* noisy*-leakage security, with the same leakage and efficiency parameters.*

7.4 Security with Concurrent Leakage Rounds

Recall that in our original definition of CLR security, the adversary runs in many *leakage-rounds*. The initial secret key sk_0 of round 0 is chosen via the `KeyGen` algorithm, and, in each subsequent round i, the secret-key sk_i is sampled via $sk_i \leftarrow \texttt{ReRand}(sk_{i-1})$. The leakage-rounds occur *sequentially*: first the adversary can leak ℓ bits on sk_0, then an additional ℓ bits on sk_1, then on sk_2 ... We call this *sequential* leakage.

We can also consider a stronger notion of security, which we call *concurrent* leakage. Firstly, for each round i, we allow the adversary to pick an arbitrary index $j \in \{0, \ldots, i-1\}$ so that the key sk_i is sampled as $sk_i \leftarrow \texttt{ReRand}(sk_j)$. Secondly, in each round i, the adversary can concurrently leak on all of the secret keys $sk_0, \ldots, sk_i$ created so far, as long as the *total* amount of leakage on each key sk_j is at most ℓ bits during the entire game. More specifically, in each round i, the adversary has access to *all* of the leakage-oracles $\mathcal{O}^\lambda_{sk_0}, \mathcal{O}^\lambda_{sk_1}, \ldots, \mathcal{O}^\lambda_{sk_i}$ *concurrently*, and can arbitrarily interleave leakage queries between them, as long as at most ℓ queries are made to each oracle over the entire game. For more advanced applications, like signature schemes, we also allow the adversary to arbitrarily interleave signature queries for the various keys sk_i.

Concurrent-leakage security may be very useful in practice. For example, we can allow for situations where there is one public-key with many (different) corresponding secret keys distributed over many parties/devices, each of which can tolerate up to ℓ bits of leakage. Moreover, the secret key in each device can be updated at will, to allow for more leakage. Lastly, any single (full) secret-key allows for the creation of arbitrarily many fresh new equivalent devices.

[5]See Appendix L for the definition of $\widetilde{\mathbf{H}}_\infty$, which denotes average-case min-entropy.

We notice that our construction of a CLR-OWR from (any) LIRR, already achieves *concurrent* security. This is because each of the keys $sk_i \leftarrow \texttt{ReRand}(sk_j)$ looks like a freshly sampled "good" key, no matter which prior key sk_j it's generated from. Moreover, the main part of the security argument that allowed us to switch a single good key for a single bad key, works even if the adversary saw all other keys in full, and hence the order of leakage does not matter. Therefore, under the K-linear assumption, we get constructions of OWRs, ID Schemes, Signatures and AKA protocols with concurrent leakage.

Theorem 7.6. *The statement of Theorem 6.1 also holds w.r.t.* concurrent*-leakage security, with the same leakage and efficiency parameters.*

References

[1] A. Akavia, S. Goldwasser, and V. Vaikuntanathan. Simultaneous hardcore bits and cryptography against memory attacks. In O. Reingold, editor, *TCC*, volume 5444 of *Lecture Notes in Computer Science*, pages 474–495. Springer, 2009.

[2] J. Alwen, Y. Dodis, M. Naor, G. Segev, S. Walfish, and D. Wichs. Public-key encryption in the bounded-retrieval model. Cryptology ePrint Archive, Report 2009/512. To Appear at Eurocrypt, 2010.

[3] J. Alwen, Y. Dodis, and D. Wichs. Leakage-resilient public-key cryptography in the bounded-retrieval model. In Halevi [40], pages 36–54.

[4] J. Alwen, Y. Dodis, and D. Wichs. Survey: Leakage resilience and the bounded retrieval model. In *ICITS*, 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*, pages 108–125, 2009.

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

[8] M. Blum, A. D. Santis, S. Micali, and G. Persiano. Noninteractive zero-knowledge. *SIAM J. Comput.*, 20(6):1084–1118, 1991.

[9] D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In *CRYPTO*, pages 41–55, 2004.

[10] V. Boyko. On the security properties of oaep as an all-or-nothing transform. In *CRYPTO*, pages 503–518, 1999.

[11] 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: ePrint 2010/278. `http://eprint.iacr.org/2010/278`.

[12] J. Camenisch, N. Chandran, and V. Shoup. A public key encryption scheme secure against key dependent chosen plaintext and adaptive chosen ciphertext attacks. In A. Joux, editor, *EUROCRYPT*, volume 5479 of *Lecture Notes in Computer Science*, pages 351–368. Springer, 2009.

[13] R. Canetti, Y. Dodis, S. Halevi, E. Kushilevitz, and A. Sahai. Exposure-resilient functions and all-or-nothing transforms. In *EUROCRYPT*, pages 453–469, 2000.

[14] R. Canetti and H. Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In B. Pfitzmann, editor, *EUROCRYPT*, volume 2045 of *Lecture Notes in Computer Science*, pages 453–474. Springer, 2001.

[15] R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In Y. Desmedt, editor, *CRYPTO*, volume 839 of *Lecture Notes in Computer Science*, pages 174–187. Springer, 1994.

[16] R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In H. Krawczyk, editor, *CRYPTO*, volume 1462 of *Lecture Notes in Computer Science*, pages 13–25. Springer, 1998.

[17] R. Cramer and V. Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In L. R. Knudsen, editor, *EUROCRYPT*, volume 2332 of *Lecture Notes in Computer Science*, pages 45–64. Springer, 2002.

[18] G. D. Crescenzo, R. J. Lipton, and S. Walfish. Perfectly secure password protocols in the bounded retrieval model. In Halevi and Rabin [41], pages 225–244.

[19] Y. Dodis, S. Goldwasser, Y. T. Kalai, C. Peikert, and V. Vaikuntanathan. Public-key encryption schemes with auxiliary inputs. In *TCC*, pages 361–381, 2010.

[20] Y. Dodis, K. Haralambiev, A. López-Alt, and D. Wichs. Efficient public-key cryptography in the presence of key leakage. Cryptology ePrint Archive, Report 2010/154, 2010. `http://eprint.iacr.org/`.

[21] Y. Dodis, Y. T. Kalai, and S. Lovett. On cryptography with auxiliary input. In *STOC*, pages 621–630, 2009.

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

[23] Y. Dodis, A. Sahai, and A. Smith. On perfect and adaptive security in exposure-resilient cryptography. In *EUROCRYPT*, pages 301–324, 2001.

[24] D. Dolev, C. Dwork, and M. Naor. Non-malleable cryptography (extended abstract). In *STOC*, pages 542–552. ACM, 1991.

[25] S. Dziembowski. Intrusion-resilience via the bounded-storage model. In Halevi and Rabin [41], pages 207–224.

[26] S. Dziembowski and K. Pietrzak. Leakage-resilient cryptography. In *FOCS*, pages 293–302, 2008.

[27] S. Faust, E. Kiltz, K. Pietrzak, and G. N. Rothblum. Leakage-resilient signatures. In *TCC*, pages 343–360, 2010.

[28] S. Faust, T. Rabin, L. Reyzin, E. Tromer, and V. Vaikuntanathan. Protecting against computationally bounded and noisy leakage. In *EUROCRYPT*, 2010. To Appear.

[29] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO*, pages 186–194, 1986.

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

[31] S. Goldwasser. Cryptography without (hardly any) secrets ? April, 2009.

[32] S. Goldwasser, Y. Kalai, C. Peikert, and V. Vaikuntanathan. Robustness of the learning with errors assumption. In *Innovations in Computer Science (ICS)*, 2010.

[33] S. Goldwasser and Y. T. Kalai. On the (in)security of the fiat-shamir paradigm. In *FOCS*, pages 102–, 2003.

[34] S. Goldwasser, Y. T. Kalai, and G. N. Rothblum. One-time programs. In D. Wagner, editor, *CRYPTO*, volume 5157 of *Lecture Notes in Computer Science*, pages 39–56. Springer, 2008.

[35] 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.

[36] J. Groth. Simulation-sound nizk proofs for a practical language and constant size group signatures. In X. Lai and K. Chen, editors, *ASIACRYPT*, volume 4284 of *Lecture Notes in Computer Science*, pages 444–459. Springer, 2006.

[37] J. Groth, R. Ostrovsky, and A. Sahai. Perfect non-interactive zero knowledge for np. In S. Vaudenay, editor, *EUROCRYPT*, volume 4004 of *Lecture Notes in Computer Science*, pages 339–358. Springer, 2006.

[38] J. Groth and A. Sahai. Efficient non-interactive proof systems for bilinear groups. In N. P. Smart, editor, *EUROCRYPT*, volume 4965 of *Lecture Notes in Computer Science*, pages 415–432. Springer, 2008.

[39] J. A. Halderman, S. D. Schoen, N. Heninger, W. Clarkson, W. Paul, J. A. Calandrino, A. J. Feldman, J. Appelbaum, and E. W. Felten. Lest we remember: cold-boot attacks on encryption keys. *Commun. ACM*, 52(5):91–98, 2009.

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

[41] S. Halevi and T. Rabin, editors. *Theory of Cryptography, Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006, Proceedings*, volume 3876 of *Lecture Notes in Computer Science*. Springer, 2006.

[42] D. Hofheinz and E. Kiltz. Secure hybrid encryption from weakened key encapsulation. In *CRYPTO*, pages 553–571, 2007.

[43] Y. Ishai, A. Sahai, and D. Wagner. Private circuits: Securing hardware against probing attacks. In *CRYPTO*, pages 463–481, 2003.

[44] A. Juma, C. Rackoff, and Y. Vahlis. Leakage resilient key proxies. To appear at CRYPTO 2010.

[45] J. Kamp and D. Zuckerman. Deterministic extractors for bit-fixing sources and exposure-resilient cryptography. In *FOCS*, pages 92–101, 2003.

[46] J. Katz and V. Vaikuntanathan. Signature schemes with bounded leakage resilience. In M. Matsui, editor, *ASIACRYPT*, volume 5912 of *Lecture Notes in Computer Science*, pages 703–720. Springer, 2009.

[47] U. M. Maurer. Unifying zero-knowledge proofs of knowledge. In *AFRICACRYPT*, pages 272–286, 2009.

[48] S. Micali and L. Reyzin. Physically observable cryptography (extended abstract). In *TCC*, pages 278–296, 2004.

[49] M. Naor and G. Segev. Public-key cryptosystems resilient to key leakage. In Halevi [40], pages 18–35.

[50] K. Pietrzak. A leakage-resilient mode of operation. In *Eurocrypt 2009, Cologne, Germany*, 2009.

[51] R. L. Rivest. All-or-nothing encryption and the package transform. In *FSE*, pages 210–218, 1997.

[52] A. D. Santis, S. Micali, and G. Persiano. Non-interactive zero-knowledge with preprocessing. In *CRYPTO*, pages 269–282, 1988.

[53] M. Scott. Authenticated id-based key exchange and remote log-in with simple token and pin number. Cryptology ePrint Archive, Report 2002/164, 2002. `http://eprint.iacr.org/`.

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

[55] E. R. Verheul. Evidence that xtr is more secure than supersingular elliptic curve cryptosystems. *J. Cryptology*, 17(4):277–296, 2004.

[56] B. Waters. Personal Communication. April 2010.

A Prior Work on Leakage-Resilient Cryptography

Below we survey the two major kinds of work on leakage-resilient cryptography: regular bounded-leakage memory attacks (and their variants) and continuous memory attacks (and their variants).

A.1 Bounded-Leakage Memory Attacks

Security against Memory Attacks. This model of leakage, sometimes called *memory-attacks*, was first proposed by Akavia, Goldwasser and Vaikuntanathan [1]. Here the attacker can learn an *arbitrary* ℓ bits of information about the secret key. This model considerably strengthens the prior model of *exposure resilient cryptography* [51, 10, 13, 23, 45], where an adversary can only learn ℓ arbitrary *physical bits* of the secret key. Back to the general model, Akavia et al. [1] constructed CPA secure PKE and IBE schemes in this model under the *learning with errors (LWE)* assumption. Naor and Segev [49] generalized the main ideas behind these constructions to show that all schemes based on *hash proof systems* (see [17]) are leakage-resilient. In particular, this resulted in efficient constructions based on the DDH and K-Linear assumptions, where the relative leakage on the secret key could be made to approach 1. Moreover, [49] showed how to also achieve CCA security in this model by giving an inefficient construction where the relative leakage can be made to approach 1, and an efficient construction with relative leakage 1/6. Recently, Dodis et al. [20] constructed efficent CCA-secure scheme in this model under the K-Linear assumption, where the relative leakage on the secret key could be made to approach 1.

The work of [2] generalizes [49] still further by showing how to construct leakage-resilient IBE schemes generically based on *identity-based hash proof systems*, with several instantiations.

Leakage-resilient signature schemes in the model of memory attacks were constructed in the random-oracle model by [3, 46], and in the standard model by [46]. The random-oracle schemes are highly-efficient but suffer from two limitations. Firstly they rely on the Fiat-Shamir [29] transform which is only known to be secure in the Random Oracle model and is not sound in general [33]. Secondly, the schemes can only tolerate leakage which approaches 1/2 of the secret key. On the other hand, the standard-model schemes allow for relative-leakage approaching 1, but are based on generic simulation-sound NIZKs and do not come with an efficient instantiation. Recently, Dodis et al. [20] constructed efficent signature scheme in this model under the K-Linear assumption, where the relative leakage on the secret key could be made to approach 1. We borrow some techniques from this work for our efficient CLR signature construction as well.

The work of [3] also constructs identification (ID) schemes and authenticated-key agreement (AKA) protocols. The parameters and/or security of these schemes were later improved by [20], to achieve strongest security notions and relative leakage approaching 1. We adapt the appropriate techniques from [3, 20] for our CLR ID schemes and CLR AKA protocols.

The work of [34] shows how to implement *any* computation securely (for a bounded number of executions), even against *leakage of all intermediate steps of the computation*, given some simple leakage-free hardware.

Extensions. We mention several related models of leakage-resilience which are strictly stronger than the memory-attacks model. Firstly, the *Bounded-Retrieval Model* [18, 25, 3, 2] imposes an additional requirement on leakage-resilient schemes, by insisting that they provide a way to "grow" the secret-key (possibly to many

Gigabytes) so as to proportionally increase the amount of tolerated leakage, but without increasing the size of the public-key, the computational-efficiency of the scheme, or the ciphertext/signature/communication lengths. The work of [3] constructs "entropic" signatures, ID schemes and AKA protocols in this setting, while the work of [2] constructs PKE and IBE schemes in this model.

Naor and Segev [49] also proposed the *noisy leakage* model, where the length of the leakage function is not bounded, although it can still "contain" at most ℓ bits of information about the secret key of the system. We consider this extension for our CLR setting in Section 7.3.

A different strengthening is the *auxiliary input model* [21, 19], where the leakage is not necessarily bounded in length, but it is (only) assumed to be computationally hard to recover the secret-key from the leakage. The work of [21] constructs symmetric-key encryption in this model, under a strengthening of the learning parity with noise (LPN) assumption, while [19] constructs public-key encryption under the DDH and LWE assumptions.

Yet another strengthening of the memory-attacks model, proposed by [32], is to require that there is a single scheme (parameterized only by the security parameter) which can tolerate essentially any amount of relative-leakage where the exact-security of the scheme degrades smoothly as the relative-leakage increases. In this model, [32] construct a symmetric-key encryption scheme.

In a different vein, Katz and Vaikuntanathan [46] considered the setting where the leakage function can also depend on the local randomness used during signature generation. We extend out CLR model to capture this improtant aspect in Section 7.1.

A.2 Continuous Leakage Attack Models

Unlike the model of bounded-leakage memory attacks, prior continuous-leakage models also restrict restrict the *type*, as well as *amount*, of information that the adversary can learn. However, schemes in such models also come with the key updating procedure, so that the overall leakage of the system is *unbounded.* The first such model was considered by Canetti et al. [13] in the context of *exposure resilient cryptography* (where, recall, the attacker can only learn ℓ physical bits of the secret). Namely, the work of [13] introduced the *gradual key exposure*, where the secret key is (deterministically) updated in a way that up to ℓ physical bits can be safely leaked in between successive refreshes. Canetti et al. [13] also showed a generic way to build a stream cipher in this model from what they called an *exposure-resilient function.*

Later, [43] studied how to implement arbitrary (stateful) computation in the setting where an adversary can observe a small *subset of the physical wires of a circuit.* Most recently, [28] study a similar problem, where the adversary can observe a low-complexity (e.g. AC^0) function of the wires. Unfortunately, these models fail to capture many meaningful side-channel attacks, such as learning the hamming-weight of the secret key bits or their parity.

In their seminal work, Micali and Reyzin [48] initiated the formal modeling of side-channel attacks under the axiom that *"only computation leaks information"* (OCLI), where each invocation of a cryptographic primitive leaks a function of *only* the bits accessed during that invocation. Several primitives have been constructed in this setting including stream ciphers [26, 50] and signatures [27]. More recently, [44, 35] construct a general compiler that can secure *all primitives* in this setting assuming the use of some limited leak-free components. On the positive side, the OCLI model only imposes a bound on the amount of information learned during each invocation of a primitive, but not on the *overall* amount of information that the attacker can get throughout the lifetime of the system. On the negative side, this model fails to capture many leakage-attacks, such as the cold-boot attack of [39], where *all* memory contents leak information, even if they were never accessed.

B Non-Interactive Zero Knowledge

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 *non-interactive zero-knowledge (NIZK) argument* for a relation R consists of four PPT algorithms $(\texttt{Setup}, \texttt{Prov}, \texttt{Ver}, \texttt{Sim})$ with syntax:

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

For the sake of clarity, we write $\texttt{Prov}, \texttt{Ver}, \texttt{Sim}$ without the $\textsc{crs}$ in the subscript when the $\textsc{crs}$ can be inferred from the context.

Definition B.1. *We say that* $(\texttt{Setup}, \texttt{Prov}, \texttt{Ver})$ *are a NIZK argument system for the relation R if the following three properties hold.*

Completeness: *For any $(y, x) \in R$, if* $(\textsc{crs}, \textsc{tk}) \leftarrow \texttt{Setup}(1^\lambda)$, $\pi \leftarrow \texttt{Prov}(y, x)$*, then* $\texttt{Ver}(y, \pi) = 1$.

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

$$\Pr\left[\begin{array}{c} \texttt{Ver}(y, \pi^*) = 1 \\ y \notin L_R \end{array} \middle| \begin{array}{c} (\textsc{crs}, \textsc{tk}) \leftarrow \texttt{Setup}(1^\lambda) \\ (y, \pi^*) \leftarrow \mathcal{A}(\textsc{crs}) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

Composable Zero-Knowledge: *For any PPT adversary $\mathcal{A}$ we have* $\left|\Pr[\mathcal{A} \textit{ wins }] - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda)$ *in the following game:*

- *The challenger samples* $(\textsc{crs}, \textsc{tk}) \leftarrow \texttt{Setup}(1^\lambda)$ *and gives* $(\textsc{crs}, \textsc{tk})$ *to* $\mathcal{A}$.
- *The adv. $\mathcal{A}$ chooses $(y, x) \in R$ and gives these to the challenger.*
- *The challenger samples* $\pi_0 \leftarrow \texttt{Prov}(y, x), \pi_1 \leftarrow \texttt{Sim}(y, \textsc{tk}), b \leftarrow \{0, 1\}$ *and gives π_b to $\mathcal{A}$.*
- *The adv. $\mathcal{A}$ outputs a bit $\tilde{b}$, and wins if $\tilde{b} = b$.*

We note that we include *composability* in our default notion of NIZK. Also, often NIZKs are defined with two different (but indistinguishable) algorithms for sampling the $\textsc{crs}$: an *honest*-$\textsc{crs}$ algorithm (which only outputs a $\textsc{crs}$ and no trapdoor $\textsc{tk}$) and a *simulated*-$\textsc{crs}$ algorithm which samples $(\textsc{crs}, \textsc{tk})$ pairs. However, in the case of NIZK *arguments* (in contrast to proofs), this distinction is irrelevant and, without loss of generality, we can just always use the *simulated*-$\textsc{crs}$ generating algorithm. (In the case of *proofs*, this distinction is important as soundness only holds unconditionally when the $\textsc{crs}$ is generated under the *honest* algorithm, without a trapdoor key.)

C One-Way Secure Encryption

Recall that an encryption scheme consists of three algorithms $\texttt{KeyGen}, \texttt{Enc}, \texttt{Dec}$ satisfying the correctness property described below:

- $(pk, sk) \leftarrow \texttt{KeyGen}(1^\lambda)$: Outputs a public/secret key pair
- $c \leftarrow \texttt{Enc}_{pk}(m)$: Given a message m and a public key pk, outputs a ciphertext c
- $m' \leftarrow \texttt{Dec}_{sk}(c)$: Given a ciphertext c and a secret key sk, outputs a message m'

Correctness: For all messages m and for all possible outputs (pk, sk) of $\texttt{KeyGen}(1^\lambda)$: $m = \texttt{Dec}_{sk}(\texttt{Enc}_{pk}(m))$.

We now define the notion of *one-way security* for encryption schemes.

Definition C.1. *An encryption scheme* $\mathcal{E} = (\texttt{KeyGen}, \texttt{Enc}, \texttt{Dec})$*, is one-way secure if, for any PPT adversary $\mathcal{A}$ we have:*

$$\Pr\left[m^* = m \mid (pk, sk) \leftarrow \texttt{KeyGen}(1^\lambda), m \leftarrow \mathcal{M}_{pk}, c \leftarrow \texttt{Enc}_{pk}(m), m^* \leftarrow \mathcal{A}(pk, c)\right] \leq \mathsf{negl}(\lambda)$$

where the public-key pk defines the message-space $\mathcal{M}_{pk}$.

It is easy to see that the standard notion of semantic-security implies one-way security if the size of the message-space $\mathcal{M}_{pk}$ is super-polynomial in the security parameter.

D Linear Algebra Notation for Prime-Order Groups

Let $\mathbb{G}$ be a group of prime order p. For clarity, we are write groups elements, such as $\mathbf{g}, \mathbf{h} \in \mathbb{G}$, in boldface to distinguish them easily from exponents such as $x, y \in \mathbb{Z}_p$. We do the same for vectors and matrices of group elements. Throughout the paper we use standard multiplicative notation when working with any group $\mathbb{G}$ of primer order p. However, when describing our schemes based on the K-linear assumption and the Groth-Sahai NIZK proof system, the presentation benefits greatly from certain linear algebra operations. Therefore, following [12], when operating on vectors or matrices of group elements, we will use linear algebra notation (defined next), which is *additive* in nature. *Note that, in this paper, all vectors are* <u>*column vectors*</u> *by default.*

Operations Between a Vector/Matrix of Group Elements and of Exponents. We define analogues of the standard linear algebra operations, where one of the operands is a vector/matrix of group elements and the other is a vector/matrix of exponents.

- The *dot-product* of the vectors $\vec{\mathbf{g}} = (\mathbf{g}_1, \dots, \mathbf{g}_n)^\top \in \mathbb{G}^n$ and $\vec{x} = (x_1, x_2, \dots, x_n)^\top \in \mathbb{Z}_p^n$ is defined as:

$$\langle \vec{x}, \vec{\mathbf{g}} \rangle = \langle \vec{\mathbf{g}}, \vec{x} \rangle \stackrel{\text{def}}{=} \prod_{i=1}^{n} \mathbf{g}_i^{x_i}.$$

- The *matrix multiplication* of $\mathbf{A} \in \mathbb{G}^{m \times n}$ and $X \in \mathbb{Z}_p^{n \times k}$ where:

$$\mathbf{A} = \{\mathbf{a}_{i,j}\}_{m \times n} = \begin{pmatrix} — & \vec{\mathbf{a}}_1^\top & — \\ — & \vec{\mathbf{a}}_2^\top & — \\ & \cdots & \\ — & \vec{\mathbf{a}}_m^\top & — \end{pmatrix} \quad , \quad X = \{x_{i,j}\}_{n \times k} = \begin{pmatrix} | & | & & | \\ \vec{x}_1 & \vec{x}_2 & \dots & \vec{x}_k \\ | & | & & | \end{pmatrix}$$

is given by

$$\mathbf{A} \cdot X \stackrel{\text{def}}{=} \{\mathbf{b}_{i,j}\}_{m \times k} \qquad \text{where} \qquad \mathbf{b}_{i,j} = \langle \vec{\mathbf{a}}_i, \vec{x}_j \rangle = \prod_{r=1}^{n} \mathbf{a}_{i,r}^{x_{r,j}}$$

For $Y = \{y_{i,j}\}_{k \times m} \in \mathbb{Z}_p^{k \times m}$, we define the matrix multiplication $Y \cdot \mathbf{A}$ analogously, so that $(Y \cdot \mathbf{A})^\top = \mathbf{A}^\top \cdot Y^\top$.

Operations Between Two Vectors/Matrices of Group Elements. We also define linear algebra operations where both operands are vectors/matrices of group elements.

- If $\mathbf{A} = \{\mathbf{a}_{i,j}\}_{m \times n}, \mathbf{B} = \{\mathbf{b}_{i,j}\}_{m \times n}$ are two matrices of *group elements*, of the same dimension, then:

$$\mathbf{A} \boxplus \mathbf{B} \stackrel{\text{def}}{=} \{\mathbf{c}_{i,j}\}_{m \times n} \qquad \text{where} \qquad \mathbf{c}_{i,j} = \mathbf{a}_{i,j}\, \mathbf{b}_{i,j}$$

is defined as the *component-wise multiplication* of the matrices. We aslo define the inverse operation:

$$\mathbf{A} \boxminus \mathbf{B} \stackrel{\text{def}}{=} \{\mathbf{c}_{i,j}\}_{m \times n} \qquad \text{where} \qquad \mathbf{c}_{i,j} = \frac{\mathbf{a}_{i,j}}{\mathbf{b}_{i,j}}$$

- Let $\mathbb{G}, \Gamma, \mathbb{G}_T$ be groups of prime order p and let $e : \mathbb{G} \times \Gamma \to \mathbb{G}_T$ be a bilinear map. Let

$$\mathbf{A} = \{\mathbf{a}_{i,j}\}_{m \times n} \in \mathbb{G}^{m \times n} \quad , \quad \boldsymbol{\Upsilon} = \{\boldsymbol{\gamma}_{i,j}\}_{n \times k} \in \Gamma^{n \times k}.$$

Then , we define the operation

$$\mathbf{A} \bullet \boldsymbol{\Upsilon} \stackrel{\text{def}}{=} \{\mathbf{t}_{i,j}\}_{m \times k} \in \mathbb{G}_T^{m \times k}, \qquad \text{where } \mathbf{t}_{i,j} = \prod_{r=1}^{n} e(\mathbf{a}_{i,r}, \boldsymbol{\gamma}_{r,j}).$$

Notice that the operands and the result of the operation all live in *different* groups.

Efficiency. All of the operations $\{+, \boxplus, \cdot, \bullet\}$ are efficient. For any $(N \times K)$, $(K \times M)$ matrices, the operations $\{\cdot, \bullet\}$ use at most $O(NKM)$ group operations (exponentiations, multiplications, pairings). For any two $(N \times M)$ matrices, the operations $\{+, \boxplus\}$ use at most $O(NM)$ group operations (multiplications).

Linear Algebra Properties. Notice that the operations $\{\boxplus, +\}$ are really the same except that $\{\boxplus\}$ acts on group elements and $\{+\}$ acts on exponents. Similarly $\{\cdot, \bullet\}$ are the same except that $\{\cdot\}$ acts on two matrices of exponents, or one matrix of exponents and one of group elements, while $\{\bullet\}$ acts on two matrices of group elements in the appropriate groups. Moreover $\{\boxplus, +\}$ satisfy the standard properties of matrix addition, while $\{\cdot, \bullet\}$ satisfy the standard properties of matrix multiplication. For example, it is easy to see that the following hold:

- Distributive property:

$$(\mathbf{A} \boxplus \mathbf{B}) \bullet \boldsymbol{\Upsilon} = (\mathbf{A} \bullet \boldsymbol{\Upsilon}) \boxplus (\mathbf{B} \bullet \boldsymbol{\Upsilon}) \qquad \text{and} \qquad \mathbf{A} \bullet (\boldsymbol{\Upsilon} \boxplus \boldsymbol{\Delta}) = (\mathbf{A} \bullet \boldsymbol{\Upsilon}) \boxplus (\mathbf{A} \bullet \boldsymbol{\Delta})$$

$$(X + Y) \cdot \mathbf{A} = (X \cdot \mathbf{A}) \boxplus (Y \cdot \mathbf{A}) \qquad \text{and} \qquad \mathbf{A} \cdot (X + Y) = (\mathbf{A} \cdot X) \boxplus (\mathbf{A} \cdot Y)$$

- Associativity:

$$(\mathbf{A} \cdot X) \bullet \boldsymbol{\Delta} = \mathbf{A} \bullet (X \cdot \boldsymbol{\Delta}) \qquad \text{and} \qquad \mathbf{B} \bullet (X \cdot \boldsymbol{\Upsilon}) = (\mathbf{B} \cdot X) \bullet \boldsymbol{\Upsilon},$$

for appropriately chosen dimensions of the matrices X, Y, $\mathbf{A}, \mathbf{B}$, and $\boldsymbol{\Upsilon}, \boldsymbol{\Delta}$ in each equation.

E The Encryption Schemes $\mathcal{E}_1, \mathcal{E}_2$ under the K-Linear Assumption

E.1 The scheme $\mathcal{E}_1$: Generalized ElGamal Encryption

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

$\mathtt{KeyGen}(\mathsf{prms})$**:** Choose $(x_1, x_2, \ldots, x_K) \leftarrow \mathbb{Z}_p^K$. Set $\mathbf{f}_1 := \mathbf{g}_0^{x_1}, \ldots, \mathbf{f}_K := \mathbf{g}_0^{x_K}$.
Output $sk = (x_1, \ldots, x_K)$, $pk = (\mathbf{f}_1, \ldots, \mathbf{f}_K)$.

$\mathtt{Enc}_{pk}(\mathbf{m})$**:** Choose $(r_1, \ldots, r_K) \leftarrow \mathbb{Z}_p^K$. Output $c := (\mathbf{m}\, \mathbf{g}_0^{\sum_{i=1}^K r_i}, \mathbf{f}_1^{r_1}, \ldots, \mathbf{f}_K^{r_K})$.

$\mathtt{Dec}_{sk}(c)$**:** Parse $c = (\mathbf{c}_0, \mathbf{c}_1, \ldots, \mathbf{c}_K)$. Output $\mathbf{c}_0 / \left(\prod_{i=1}^K \mathbf{c}_i^{1/x_i}\right)$.

Figure 8: Generalized ElGamal

For $\mathcal{E}_1$, we can use any homomorphic one-way secure encryption in bilinear groups, with security based on the K-linear assumption. We choose to use a simple generalization of the ElGamal cryptosystem, shown in Figure 5.

Notice that, with $K = 1$, we recover the scheme from Figure 5.

Theorem E.1. *For any $K \geq 1$, the generalized ElGamal scheme (Figure 8) is semantically-secure and one-way secure under the the K-linear assumption. Furthermore, it is homomorphic over the message-group $\mathcal{M} = \mathbb{G}$, randomness-group $\mathcal{R} = \mathbb{Z}_p^K$ and ciphertext-group $\mathcal{C} = \mathbb{G}^{K+1}$.*

Proof. Follows directly from the K-linear assumption. □

E.2 The scheme $\mathcal{E}_2$: Generalized "Cramer-Shoup Lite"

A generalization of the Multiple "Cramer-Shoup Lite" (CS-Lite) scheme, secure under the K-linear assumption, is shown in Figure 9. We note that similar generalizations of Cramer-Shoup to the K-linear assumption were already shown in [54, 12]. Notice that, with $K = 1$, we recover the scheme from Figure 6. Therefore, Theorem 5.1 follows as a corollary of the following theorem.

Let $K, n \in \mathbb{Z}^+$. Let $\mathsf{prms} = (p, \mathbb{G}, \mathbf{g}_0, \ldots, \mathbf{g}_K)$ where $(p, \mathbb{G}, \mathbf{g}_0) \leftarrow \mathcal{G}(1^\lambda)$ and $\mathbf{g}_1, \ldots, \mathbf{g}_K \leftarrow \mathbb{G}$.
The parameters implicitly define the $K \times (K+1)$ matrix:

$$\mathbf{A} = \begin{pmatrix} \mathbf{g}_0 & \mathbf{g}_1 & 1 & 1 & \ldots & 1 \\ \mathbf{g}_0 & 1 & \mathbf{g}_2 & 1 & \ldots & 1 \\ \mathbf{g}_0 & 1 & 1 & \mathbf{g}_3 & \ldots & 1 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{g}_0 & 1 & 1 & 1 & \ldots & \mathbf{g}_K \end{pmatrix}$$

$\mathtt{KeyGen}(\mathsf{prms})$**:** Choose $(n+1)$ random vectors $\left\{\vec{x}_i \leftarrow \mathbb{Z}_p^{(K+1)}\right\}_{i=0}^{n}$.
Set $\vec{\mathbf{h}}_0 := \mathbf{A} \cdot \vec{x}_0$, $\vec{\mathbf{h}}_1 := \mathbf{A} \cdot \vec{x}_1$, $\ldots$ $, \vec{\mathbf{h}}_n := \mathbf{A} \cdot \vec{x}_n$. Output $pk := (\vec{\mathbf{h}}_0, \vec{\mathbf{h}}_1, \ldots, \vec{\mathbf{h}}_n)$, $sk := (\vec{x}_0, \vec{x}_1 \ldots, \vec{x}_n)$.

$\mathtt{Enc}_{pk}(\mathbf{m})$**:** To encrypt a message $\mathbf{m} \in \mathbb{G}$, choose $\vec{r} \leftarrow \mathbb{Z}_p^K$ and set $\vec{\mathbf{y}} := \vec{r}^\top \cdot \mathbf{A}$
Compute $\mathbf{c}_0 := \langle \vec{\mathbf{h}}_0, \vec{r} \rangle$, $\mathbf{c}_1 := \langle \vec{\mathbf{h}}_1, \vec{r} \rangle$, $\ldots$ $, \mathbf{c}_n := \langle \vec{\mathbf{h}}_n, \vec{r} \rangle$ and $\mathbf{z} := \mathbf{c}_0 \mathbf{m}$. Output $c := (\vec{\mathbf{y}}, \mathbf{z}, \mathbf{c}_1, \ldots, \mathbf{c}_n)$.

$\mathtt{Dec}_{sk}(c)$**:** Parse $c = (\vec{\mathbf{y}}, \mathbf{z}, \mathbf{c}_1, \ldots, \mathbf{c}_n)$. Compute $\tilde{\mathbf{c}}_0 := \langle \vec{\mathbf{y}}, \vec{x}_0 \rangle$, $\tilde{\mathbf{c}}_1 := \langle \vec{\mathbf{y}}, \vec{x}_1 \rangle$, $\ldots$ $, \tilde{\mathbf{c}}_n := \langle \vec{\mathbf{y}}, \vec{x}_n \rangle$.
If $\tilde{\mathbf{c}}_1 \overset{?}{=} \mathbf{c}_1, \ldots, \tilde{\mathbf{c}}_n \overset{?}{=} \mathbf{c}_n$, then output $\mathbf{z}/\tilde{\mathbf{c}}_0$. Else output $\perp$.

Figure 9: Generalized Multiple CS-Lite Scheme

Theorem E.2. *For any $K \geq 1$, $n \geq 0$, the generalized Multiple CS-Lite scheme (Figure 9) is an ℓ-LoC-NM secure encryption under the K-linear assumption, and with leakage $\ell = n \log(p) - \lambda$. Furthermore it is a homomorphic over the message-group $\mathcal{M} = \mathbb{G}$, randomness-group $\mathcal{R} = \mathbb{Z}_p^K$, and ciphertext-group $\mathcal{C} = \mathbb{G}^{n+K+2}$.*

Proof. The scheme satisfies perfect correctness since, for each $i \in \{0, \ldots, n\}$:

$$\mathbf{c}_i = \langle \vec{\mathbf{h}}_i, \vec{r} \rangle = \langle (\mathbf{A} \cdot \vec{x}_i), \vec{r} \rangle = \vec{r}^\top \cdot (\mathbf{A} \cdot \vec{x}_i) = (\vec{r}^\top \cdot \mathbf{A}) \cdot \vec{x}_i = \vec{\mathbf{y}}^\top \cdot \vec{x}_i = \langle \vec{\mathbf{y}}, \vec{x}_i \rangle = \tilde{\mathbf{c}}_i \quad .$$

For security, we first prove two simple claims:

Claim E.3. *Let $\mathbf{g}_0, \ldots, \mathbf{g}_K$ be random generators of a group $\mathbb{G}$, and define the matrix $\mathbf{A}$ as in Figure 6. Let $\vec{\mathbf{y}}_0 \leftarrow \mathtt{rowspace}(\mathbf{A})$ and $\vec{\mathbf{y}}_1 \leftarrow \mathbb{G}^{K+1} \setminus \mathtt{rowspace}(\mathbf{A})$. Then, under the K-linear assumption over $\mathbb{G}$, the following are computationally indistinguishable: $(\mathbf{g}_0, \ldots, \mathbf{g}_K, \vec{\mathbf{y}}_0) \overset{c}{\approx} (\mathbf{g}_0, \ldots, \mathbf{g}_K, \vec{\mathbf{y}}_1)$.*

Proof. The K-linear assumption is equivalent to saying that $(\mathbf{g}_0, \ldots, \mathbf{g}_K, \vec{\mathbf{y}}_0) \overset{c}{\approx} (\mathbf{g}_0, \ldots, \mathbf{g}_K, \vec{\mathbf{y}}_{0.5})$ where $\vec{\mathbf{y}}_{0.5}$ is chosen uniformly and independently from $\mathbb{G}^{K+1}$. But, even if we fix *any* $\mathbf{g}_0, \ldots, \mathbf{g}_K$, the conditional distributions of $\vec{\mathbf{y}}_{0.5}$ and $\vec{\mathbf{y}}_1$ are statistically close. This is because $\Pr[\vec{\mathbf{y}}_{0.5} \in \mathtt{rowspace}(\mathbf{A})] = p^K/p^{K+1} = 1/p$ is negligible. ☐

Claim E.4. *Let $\mathbf{g}_0, \ldots, \mathbf{g}_K$ be any* fixed *generators of a group $\mathbb{G}$, and define the matrix $\mathbf{A}$ as in Figure 6. Fix* any *$\vec{\mathbf{y}} \in \mathbb{G}^{K+1} \setminus \mathtt{rowspace}(\mathbf{A})$. Then, over a random $\vec{x} \leftarrow \mathbb{Z}_p^{K+1}$, the distribution of $(\mathbf{A} \cdot \vec{x}, \langle \vec{\mathbf{y}}, \vec{x} \rangle)$ is exactly that of a uniformly random sample from $\mathbb{G}^{K+1}$.*

Proof. This follows from the fact that the rows of $\mathbf{A}$, together with the vector $\vec{\mathbf{y}}$, span all of $\mathbb{G}^{K+1}$ and so the linear map $f_{(\mathbf{A}, \vec{\mathbf{y}})}(\vec{x}) = (\mathbf{A} \cdot \vec{x}, \langle \vec{\mathbf{y}}, \vec{x} \rangle)$ is a bijective. ☐

Returning to the proof of Theorem E.2, we now do a series-of-games argument to show that the scheme satisfies ℓ-LoC-NM security.

Game 0: Let Game 0 be the original ℓ-LoC-NM game from Definition 3.1.

Game 1: In this game, the challenge ciphertext is computed using the decryption key. That is, instead of computing the components of the challenge ciphertext as $\mathbf{c}_i := \langle \vec{\mathbf{h}}_i, \vec{r} \rangle$, the challenger computes them as $\tilde{\mathbf{c}}_i := \langle \vec{\mathbf{y}}, \vec{x} \rangle$ and sets $\tilde{\mathbf{z}} := \tilde{\mathbf{c}}_0 \mathbf{m}_b$ (where b is the challenge bit). The challenge ciphertext is then $c = (\vec{\mathbf{y}}, \tilde{\mathbf{z}}, \tilde{\mathbf{c}}_1, \ldots, \tilde{\mathbf{c}}_n)$.

The change between Game 0 and Game 1 is only syntactic and the two games are identically distributed. This follows by the perfect correctness of decryption.

Game 2: In this game, the challenge ciphertext is computed the same way as in Game 1, *except* that the challenger samples $\vec{\mathbf{y}} \leftarrow \mathbb{G}^{K+1} \setminus \texttt{rowspace}(\mathbf{A})$ and outputs $c = (\vec{\mathbf{y}}, \tilde{\mathbf{z}}, \tilde{\mathbf{c}}_1, \ldots, \tilde{\mathbf{c}}_n)$.

Notice that in Games 1 and 2, the components $\tilde{\mathbf{c}}_i$ do not depend on $\vec{r}$, and so they are still well-defined.

We argue that Games 1 and 2 are computationally indistinguishable under the K-linear assumption (even if the adversary were given the entire ciphertext c *and* the secret key sk). This follows from Claim E.3. Therefore, the probability of $\mathcal{A}$ winning in Game 2 differs at most negligibly from that of Game 1.

Game 3: In this game, the challenger will run in exponential-time. When the adversary submits the "decryption query" $c^* = (\vec{\mathbf{y}}^*, \mathbf{z}^*, \mathbf{c}_1^*, \ldots, \mathbf{c}_n^*)$, the challenger automatically outputs $\perp$ if $\vec{\mathbf{y}}^* \notin \texttt{rowspace}(\mathbf{A})$, which it checks in exponential time. Otherwise, the challenger finds (in exponential time) the unique solution $\vec{r}$ such that $\vec{\mathbf{y}}^* = \vec{r} \cdot \mathbf{A}$, and computes $\hat{\mathbf{c}}_0^* = \langle \vec{\mathbf{h}}_0, \vec{r} \rangle, \ldots, \hat{\mathbf{c}}_n^* = \langle \vec{\mathbf{h}}_n, \vec{r} \rangle$. If $\hat{\mathbf{c}}_i^* \overset{?}{=} \mathbf{c}_i^*$ for all $i \in \{1, \ldots, n\}$, it outputs $\tilde{\mathbf{m}}^* = \mathbf{z}^* / \hat{\mathbf{c}}_0^*$ and else $\perp$.

We show that games 2 and 3 are *statistically indistinguishable*, even if the adversary $\mathcal{A}$ is computationally unbounded.

Case 1: If $\vec{\mathbf{y}}^* \in \texttt{rowspace}(\mathbf{A})$ then the challenger's response is identical in games 2 and 3 (this follows by the correctness calculation $\tilde{\mathbf{c}}_i^* = \hat{\mathbf{c}}_i^*$).

Case 2: If $\vec{\mathbf{y}}^* \notin \texttt{rowspace}(\mathbf{A})$, then we argue that the response in Game 2 is also $\perp$ with overwhelming probability. By Claim E.4, the values $\{\tilde{\mathbf{c}}_i^* = \langle \vec{\mathbf{y}}^*, \vec{x}_i \rangle\}_{i=1}^n$, used to decide if c^* is well-formed (in Game 2), are mutually uniform over the randomness of $\vec{x}_i$, even conditioned on (any) outcome of the public-key components $\vec{\mathbf{h}}_i = \mathbf{A} \cdot \vec{x}_i$ and the generators $\mathbf{g}_0, \ldots, \mathbf{g}_K$. On the other hand, the challenge-ciphertext c (as computed in Game 2) can reveal additional information about these values (e.g. if $\vec{\mathbf{y}}^* = \vec{\mathbf{y}}$ is just copied from the challenge-ciphertext, then the corresponding values $\tilde{\mathbf{c}}_i^* = \tilde{\mathbf{c}}_i$ can just be copied from the challenge-ciphertext exactly). However, the adversary only gets ℓ bits of information about the challenge ciphertext c and has to guess all n values $\tilde{\mathbf{c}}_i^*$ correctly in order for c^* not to decrypt to $\perp$. Since each value $\tilde{\mathbf{c}}_i$ is uniform over a group of size p, and the adversary is only give $\ell = n \log(p) - \lambda$ bits of correlated information, her probability of success is at most $1/2^\lambda$, which is negligible.

Hence, the only difference between games 2 and 3 is that, if $\vec{\mathbf{y}}^* \notin \texttt{rowspace}(\mathbf{A})$, then the probability that the challenger answerers with $\perp$ in Game 2 is $\geq 1 - \frac{1}{2^\lambda}$ and in game 3 it is 1. Therefore games 2 and 3 are statistically indistinguishable.

We now argue that, in Game 3, the challenger's bit b is perfectly hidden (even for a computationally unbounded adversary $\mathcal{A}$). This follows by applying Claim E.4 to the value $\tilde{\mathbf{c}}_0$, used as a one-time-pad when computing $\tilde{\mathbf{z}} = \tilde{\mathbf{c}}_0 \mathbf{m}_b$ in the challenge ciphertext $c = (\vec{\mathbf{y}}, \tilde{\mathbf{z}}, \tilde{\mathbf{c}}_1, \ldots, \tilde{\mathbf{c}}_n)$. In particular, by Claim E.4, the value $\tilde{\mathbf{c}}_0$ is uniformly random over $\mathbb{G}$, even conditioned on the observed values of the generators $\mathbf{g}_0, \ldots, \mathbf{g}_n$ and the public-key $pk = (\mathbf{h}_0, \ldots, \mathbf{h}_n)$. Moreover, the answer to the decryption query in Game 3 can now be (inefficiently) perfectly simulated given $\mathbf{g}_0, \ldots, \mathbf{g}_K, \mathbf{h}_0, \ldots, \mathbf{h}_K$ and hence does not reveal any additional information about $\tilde{\mathbf{c}}_0$. Therefore, the challenger's bit b stays perfectly (information theoretically) hidden in Game 3.

So the adversary's probability of winning in Game 3 is exactly $\frac{1}{2}$ and, by the hybrid argument, the probability of winning in Game 0 must therefore be at most negligibly close to $\frac{1}{2}$, as we wanted to show. □

We notice that, in the proof of Theorem E.2, we never made use of the fact that the adversary only submits *one* decryption query at the end of the game. Indeed, we achieve a stronger notion then just ℓ-LoC-NM security (closer to ℓ-LoC CCA-2 security). Even if the adversary can adaptively access a *decryption oracle* arbitrarily many times during the game, before and after leaking on the challenge ciphertext, as long as the leakage function cannot access it and the adversary only leaks ℓ bits, the challenger's bit stays hidden.

E.3 The Plaintext-Equality Relation R_{eq}

We now generalize the plaintext-equality relation R_{eq} to the encryption schemes $\mathcal{E}_1, \mathcal{E}_2$ based on the K-linear assumption. For any statement (pk_1, pk_2, c_1, c_2) where

$$\begin{array}{ll} pk_1 = \vec{\mathbf{f}} = (\mathbf{f}_1, \ldots, \mathbf{f}_K) \in \mathbb{G}^K, & c_1 = (\mathbf{z}_1, \mathbf{c}_{1,1}, \ldots, \mathbf{c}_{1,K}) \in \mathbb{G}^{K+1} \\ pk_2 = \left(\vec{\mathbf{h}}_0, \vec{\mathbf{h}}_1, \ldots, \vec{\mathbf{h}}_n\right) \in \mathbb{G}^{K\times(n+1)} & c_2 = (\vec{\mathbf{y}}, \mathbf{z}_2, \mathbf{c}_{2,1}, \ldots, \mathbf{c}_{2,n}) \in \mathbb{G}^{(n+K+2)}. \end{array}$$

Then $(pk_1, pk_2, c_1, c_2) \in L_{eq}$ if and only if there exist a vector $\vec{w} = (\vec{r}_1, \vec{r}_2) \in \mathbb{Z}_p^{2K}$, $\vec{r}_1 = (r_{1,1}, r_{1,2}, \ldots, r_{1,K})$ such that:

$$\mathbf{f}_1^{r_{1,1}} = \mathbf{c}_{1,1}, \quad \mathbf{f}_2^{r_{1,2}} = \mathbf{c}_{1,2} \quad, \ldots, \quad \mathbf{f}_K^{r_{1,K}} = \mathbf{c}_{1,K} \tag{1}$$

$$\vec{r}_2^{\top} \cdot \mathbf{A} = \vec{\mathbf{y}} \tag{2}$$

$$\langle \vec{\mathbf{h}}_0, \vec{r}_2 \rangle \prod_{i=0}^{K} (\mathbf{g}_0^{-1})^{r_{1,i}} = \mathbf{c}_{2,0}/\mathbf{c}_{1,0} \tag{3}$$

$$\langle \vec{\mathbf{h}}_1, \vec{r}_2 \rangle = \mathbf{c}_{2,1} \quad, \ldots, \quad \langle \vec{\mathbf{h}}_n, \vec{r}_2 \rangle = \mathbf{c}_{2,n} \tag{4}$$

That makes for $M = 2K + n + 2$ equations in $N = 2K$ unknowns.

F The Proof-System Π : Groth-Sahai (GS) Proof System for Linear Equations

In [38], Groth and Sahai give a NIZK proof system for showing that a system of equations of certain type over bilinear groups are *satisfiable*. For many applications, including the construction of LIRR, the simplest type of equations, linear equations, suffices. Recall that we say a system of linear equations $(\mathbf{B}, \vec{\mathbf{c}})$, where $\mathbf{B}$ is a $M \times N$ matrix of group elements and $\vec{\mathbf{c}}$ is a vector of M group elements in $\mathbb{G}$, is satisfiable if there exists a vector of exponents $\vec{x} \in \mathbb{Z}_p^N$ such that

$$\begin{pmatrix} \mathbf{b}_{1,1} & \mathbf{b}_{1,2} & \ldots & \mathbf{b}_{1,N} \\ \mathbf{b}_{2,1} & \mathbf{b}_{2,2} & \ldots & \mathbf{b}_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{b}_{M,1} & \mathbf{b}_{M,2} & \ldots & \mathbf{b}_{M,N} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{pmatrix} = \begin{pmatrix} \mathbf{c}_1 \\ \mathbf{c}_2 \\ \vdots \\ \mathbf{c}_M \end{pmatrix}.$$

Such vector $\vec{x}$ is called a *satisfying assignment* for the system $(\mathbf{B}, \vec{\mathbf{c}})$. The relation R_{linear} is defined for all pairs of systems of linear equations and their satisfying assignments $((\mathbf{B}, \vec{\mathbf{c}}), \vec{x})$.

For completeness, we present the Groth-Sahai (GS) NIZK proof system for linear equations in detail, including its correctness and security. Then, we observe that it is homomorphic (when $\mathbf{B}$ is fixed).

Altogether, this gives us a homomorphic NIZK proof system. Note that we use the GS NIZK proof system as a NIZK argument system, achieving only computational soundness. This can be done by running all the algorithms with a simulated CRS as described below. As a result we prove Lemma F.1 which generalizes Lemma 5.2.

For common parameters $(p, \mathbb{G}, \Gamma, \mathbb{G}_T, e, \mathbf{g}, \boldsymbol{\gamma}_0) \leftarrow \mathcal{G}_{pair}(1^\lambda)$:

$\mathtt{Setup}(1^\lambda)$: Output $\textsc{crs} = \boldsymbol{\Upsilon}$ and $\textsc{tk} = \vec{t}$, where $\boldsymbol{\gamma}_1, \ldots, \boldsymbol{\gamma}_K \leftarrow \Gamma$, $\vec{t} = (t_1, \ldots, t_K)^\top \leftarrow \mathbb{Z}_p^K$, and

$$\boldsymbol{\Upsilon} = \begin{pmatrix} \boldsymbol{\gamma}_0^{\sum_{i=1}^K t_i} & \boldsymbol{\gamma}_1^{t_1} & \boldsymbol{\gamma}_2^{t_2} & \boldsymbol{\gamma}_3^{t_3} & \ldots & \boldsymbol{\gamma}_K^{t_K} \\ \boldsymbol{\gamma}_0 & \boldsymbol{\gamma}_1 & 1 & 1 & \ldots & 1 \\ \boldsymbol{\gamma}_0 & 1 & \boldsymbol{\gamma}_2 & 1 & \ldots & 1 \\ \boldsymbol{\gamma}_0 & 1 & 1 & \boldsymbol{\gamma}_3 & \ldots & 1 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{\gamma}_0 & 1 & 1 & 1 & \ldots & \boldsymbol{\gamma}_K \end{pmatrix} \in \Gamma^{(K+1)\times(K+1)}.$$

$\mathtt{Prov}_{\mathbf{crs}}((\mathbf{B}, \vec{\mathbf{c}}), \vec{x})$: Assume $\mathbf{B} \in \mathbb{G}^{M\times N}, \vec{\mathbf{c}} \in \mathbb{G}^M, \vec{x} \in \mathbb{Z}_p^N$. Select $R \leftarrow \mathbb{Z}_p^{N\times K}$ and output $\pi = (\boldsymbol{\Delta}, \mathbf{P})$ where:

$$\boldsymbol{\Delta} = \left(\begin{array}{c|c} x_1 & \\ \vdots & R \\ x_N & \end{array}\right) \cdot \boldsymbol{\Upsilon} \quad \text{and} \quad \mathbf{P} = \mathbf{B} \cdot R$$

for $\boldsymbol{\Delta} \in \Gamma^{N\times(K+1)}$, $\mathbf{P} \in \mathbb{G}^{M\times K}$.

$\mathtt{Sim}_{\mathbf{crs}}((\mathbf{B}, \vec{\mathbf{c}}), \mathbf{tk})$: Assume $\mathbf{B} \in \mathbb{G}^{M\times N}, \vec{\mathbf{c}} \in \mathbb{G}^M$. Select $R \leftarrow \mathbb{Z}_p^{N\times K}$ and output $\pi = (\boldsymbol{\Delta}, \mathbf{P})$ where:

$$\boldsymbol{\Delta} = \left(\begin{array}{c|c} 0 & \\ \vdots & R \\ 0 & \end{array}\right) \cdot \boldsymbol{\Upsilon} \quad \text{and} \quad \mathbf{P} = \left(\begin{array}{c|c} \mathbf{c}_1 & \\ \vdots & \mathbf{B} \\ \mathbf{c}_M & \end{array}\right) \cdot \left(\frac{-t_1 \ldots -t_N}{R}\right) .$$

$\mathtt{Ver}_{\mathbf{crs}}((\mathbf{B}, \vec{\mathbf{c}}), \pi)$: Parse $\pi = (\boldsymbol{\Delta}, \mathbf{P})$ and output 1 iff

$$\mathbf{B} \bullet \boldsymbol{\Delta} = \left(\begin{array}{c|c} \mathbf{c}_1 & \\ \vdots & \mathbf{P} \\ \mathbf{c}_M & \end{array}\right) \bullet \boldsymbol{\Upsilon} .$$

Figure 10: The GS NIZK system for proving a set of linear equations is satisfiable under K-Linear.

F.1 Construction

In Figure 10, we describe the NIZK argument system of [38] for showing that a system of linear equations is satisfiable, following the presentation of [12] and using the notation from Appendix D.

F.2 Correctness and Security

Recall from Appendix D that $\cdot$ and $\bullet$ are associative operations, and both of them are distributive over $\boxplus$.

Correctness: It is easy to verify that the verification equation holds if $\mathbf{B} \cdot \vec{x} = \vec{\mathbf{c}}$:

$$\begin{aligned} \mathbf{B} \bullet \boldsymbol{\Delta} &= \mathbf{B} \bullet \left(\left(\begin{array}{c|c} \vec{x} & R \end{array} \right) \cdot \boldsymbol{\Upsilon} \right) = \left(\mathbf{B} \cdot \left(\begin{array}{c|c} \vec{x} & R \end{array} \right) \right) \bullet \boldsymbol{\Upsilon} \\ &= \left(\begin{array}{c|c} \mathbf{B} \cdot \vec{x} & \mathbf{B} \cdot R \end{array} \right) \bullet \boldsymbol{\Upsilon} = \left(\begin{array}{c|c} \vec{\mathbf{c}} & \mathbf{P} \end{array} \right) \bullet \boldsymbol{\Upsilon} \end{aligned}$$

Perfect Simulation: First notice that $\boldsymbol{\Upsilon}$ could be viewed as

$$\boldsymbol{\Upsilon} = \left(\frac{\vec{\boldsymbol{\tau}}^\top}{\boldsymbol{\Upsilon}'} \right), \text{where } \boldsymbol{\Upsilon}' = \begin{pmatrix} \boldsymbol{\gamma}_0 & \boldsymbol{\gamma}_1 & 1 & \ldots & 1 \\ \boldsymbol{\gamma}_0 & 1 & \boldsymbol{\gamma}_2 & \ldots & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{\gamma}_0 & 1 & 1 & \ldots & \boldsymbol{\gamma}_K \end{pmatrix}, \text{ and } \vec{\boldsymbol{\tau}}^\top = \vec{t}^\top \cdot \boldsymbol{\Upsilon}'.$$

Let $(\mathbf{B}, \vec{\mathbf{c}}) \in L_{linear}$ be a *true* statement with witness $\vec{x}$.

For $\pi = (\mathbf{\Delta}, \mathbf{P})$ outputted by the `Prov` algorithm, we can write:

$$\begin{aligned}
\mathbf{\Delta} &= \left(\vec{x} \mid R \right) \cdot \mathbf{\Upsilon} = \left(\vec{x} \mid R \right) \cdot \left(\frac{\vec{\boldsymbol{\tau}}^\top}{\mathbf{\Upsilon}'} \right) \\
&= \left(\vec{x} \cdot \vec{\boldsymbol{\tau}}^\top\right) \boxplus \left(R \cdot \mathbf{\Upsilon}'\right) = \left(\vec{x} \cdot \left(\vec{t}^\top \cdot \mathbf{\Upsilon}'\right)\right) \boxplus \left(R \cdot \mathbf{\Upsilon}'\right) = \left(R + \vec{x} \cdot \vec{t}^\top\right) \cdot \mathbf{\Upsilon}' \\
\mathbf{P} &= \mathbf{B} \cdot R.
\end{aligned}$$

For $\tilde{\pi} = (\tilde{\mathbf{\Delta}}, \tilde{\mathbf{P}})$ produced by `Sim` we can write:

$$\begin{aligned}
\tilde{\mathbf{\Delta}} &= \left(\vec{0} \mid R \right) \cdot \left(\frac{\vec{\boldsymbol{\tau}}^\top}{\mathbf{\Upsilon}'} \right) = \left(\vec{0} \cdot \vec{\boldsymbol{\tau}}^\top\right) \boxplus \left(R \cdot \mathbf{\Upsilon}'\right) = R \cdot \mathbf{\Upsilon}' \\
\tilde{\mathbf{P}} &= \left(\vec{\mathbf{c}} \mid \mathbf{B} \right) \cdot \left(\frac{-\vec{t}^\top}{R} \right) \\
&= \left(\vec{\mathbf{c}} \cdot -\vec{t}^\top\right) \boxplus \left(\mathbf{B} \cdot R\right) = \left((\mathbf{B} \cdot \vec{x}) \cdot -\vec{t}^\top\right) \boxplus \left(\mathbf{B} \cdot R\right) = \mathbf{B} \cdot \left(R - \vec{x} \cdot \vec{t}^\top\right)
\end{aligned}$$

Since for fixed $\vec{x}, \vec{t}$, we have the distributional equivalence (over a uniformly random R):

$$\left(R + \vec{x} \cdot \vec{t}^\top, \quad R\right) \equiv \left(R, \quad R - \vec{x} \cdot \vec{t}^\top\right)$$

it is clear that $(\mathbf{\Delta}, \mathbf{P}) \equiv (\tilde{\mathbf{\Delta}}, \tilde{\mathbf{P}})$ are also distributionally equivalent. Therefore the simulation is perfect, *even* given the trapdoor key $\vec{t}$. This proves the composable NIZK property.

Soundness To show soundness, change `Setup` to output $\mathbf{\Upsilon} = \left(\frac{\vec{\boldsymbol{\tau}}^\top}{\mathbf{\Upsilon}'} \right)$, where $\mathbf{\Upsilon}'$ is defined as above, but the first row of $\mathbf{\Upsilon}$ is chosen randomly so that $\vec{\boldsymbol{\tau}}^\top \notin \mathtt{rowspace}(\mathbf{\Upsilon}')$. This CRS is indistinguishable from a properly generated CRS by the K-linear assumption. But then $\mathbf{\Upsilon}$ is a full-rank square matrix, an can therefore be viewed as a basis of a vector space. Hence for any proof $\pi = (\mathbf{\Delta}, \mathbf{P})$, there exist unique $\vec{x}$ and R such that $\left(\vec{x} \mid R \right) \cdot \mathbf{\Upsilon} = \mathbf{\Delta}$ (as each row of $\mathbf{\Delta}$ is a vector in the vector space and has a unique representation). The left side of the verification equation is then equal to

$$\mathbf{B} \bullet \mathbf{\Delta} = \mathbf{B} \bullet \left(\left(\vec{x} \mid R \right) \cdot \mathbf{\Upsilon}\right) = \left(\mathbf{B} \cdot \vec{x} \mid \mathbf{B} \cdot R \right) \bullet \mathbf{\Upsilon}$$

which is equals the right side, $\left(\vec{\mathbf{c}} \mid \mathbf{P} \right) \bullet \mathbf{\Upsilon}$, if and only if $\mathbf{B} \cdot \vec{x} = \vec{\mathbf{c}}$ and $\mathbf{P} = \mathbf{B} \cdot R$. That is, if and only if the system of linear equations is satisfiable and π is a valid proof.

F.3 GS Proofs are Homomorphic

When the matrix $\mathbf{B}$ is fixed, it is easy to see that $R^{\mathbf{B}}_{linear}$ is a homomorphic relation, since, for $(\vec{\mathbf{c}}, \vec{x}), (\vec{\mathbf{c}}', \vec{x}') \in R^{\mathbf{B}}_{linear}$, we have

$$\mathbf{B} \cdot \vec{x} = \vec{\mathbf{c}} \quad , \quad \mathbf{B} \cdot \vec{x}' = \vec{\mathbf{c}}' \quad \Rightarrow \quad \mathbf{B} \cdot (\vec{x} + \vec{x}') = (\mathbf{B} \cdot \vec{x}) \boxplus (\mathbf{B} \cdot \vec{x}') = \vec{c} \boxplus \vec{c}'$$

so $(\vec{\mathbf{c}} \boxplus \vec{\mathbf{c}}', \vec{x} + \vec{x}') \in R^{\mathbf{B}}_{linear}$.

Let $\pi = (\mathbf{\Delta}, \mathbf{P})$ and $\pi' = (\mathbf{\Delta}', \mathbf{P}')$ be proofs for the statements $(\vec{\mathbf{c}}, \vec{x}), (\vec{\mathbf{c}}', \vec{x}') \in R^{\mathbf{B}}_{linear}$, generated honestly by `Prov` with the coins R, R' respectively. We can write $\pi \cdot \pi' = (\mathbf{\Delta}^*, \mathbf{P}^*)$ where

$$\begin{aligned}
\mathbf{\Delta}^* &= \mathbf{\Delta} \boxplus \mathbf{\Delta}' = \left(\left(\vec{x} \mid R \right) \cdot \mathbf{\Upsilon}\right) \boxplus \left(\left(\vec{x}' \mid R' \right) \cdot \mathbf{\Upsilon}\right) = \left(\vec{x} + \vec{x}' \mid R + R' \right) \cdot \mathbf{\Upsilon} \\
\mathbf{P}^* &= \mathbf{P} \boxplus \mathbf{P}' = (\mathbf{B} \cdot R) \boxplus (\mathbf{B} \cdot R') = \mathbf{B} \cdot (R + R')
\end{aligned}$$

This shows that $\pi \cdot \pi'$ is the same as a proof for the statement $(\vec{\mathbf{c}} \boxplus \vec{\mathbf{c}}', \vec{x} + \vec{x}') \in R^{\mathbf{B}}_{linear}$, generated with random coins $R + R'$ of the prover. Therefore, the GS argument system for the relation R_{linear} is homomorphic. Together with the correctness/security properties, we therefore prove the following lemma, generalizing Lemma 5.2:

Lemma F.1. *Fix a constant $K \geq 1$, and assume that the K-linear assumption holds in both base groups for some pairing $\mathcal{G}_{pair}$. Then there exists a* homomorphic *NIZK argument system for the relation R_{linear}. For any statement consisting of M linear equation in N unknowns, the corresponding proof π contains $(K+1)N + KM$ group elements. The creation/verification of π uses $O(NM)$ group operations.*

G Putting It All Together : Efficient CLR-OWRs

We now use our results from the previous sections, showing how to construct the schemes $\mathcal{E}_1, \mathcal{E}_2, \Pi$, to prove Theorem 5.3. Fix a constant $K \geq 1$ and assume that the K-linear assumption holds in both base groups $\mathbb{G}, \Gamma$ of a symmetric *or* asymmetric pairing $\mathcal{G}_{pair}$.
Taking the schemes $\mathcal{E}_1, \mathcal{E}_2, \Pi$ from the previous sections (all of which are parameterized by K) we see that:

- $\mathcal{E}_1$ is semantically-secure and therefore also one-way secure.
  In addition, it is homomorphic over the message-group $\mathcal{M} = \mathbb{G}$. (Theorem E.1)
- $\mathcal{E}_2$, parameterized by n, is ℓ-LoC-NM secure where $\ell \geq n\log(p) - \lambda$.
  In addition, it is homomorphic over the message-group $\mathcal{M} = \mathbb{G}$. (Theorem E.2)
- Π is a homomorphic NIZK for the relation R_{eq} defined by $\mathcal{E}_1, \mathcal{E}_2$. (Lemma F.1)

Therefore, by Theorem 4.9, when we instantiate our construction (Figure 2) with the schemes $\mathcal{E}_1, \mathcal{E}_2, \Pi$, we get an LIRR and therefore also a CLR-OWR (Figure 4). This proves the existence of ℓ-CLR-OWRs for arbitrarily large ℓ.

Relative Leakage. For any choice of $n \in \mathbb{N}$, the ciphertext of the scheme $\mathcal{E}_2$ consists of $n + K + 2$ group elements. From Appendix E.3, we see that the equality relation R_{eq} for $\mathcal{E}_1, \mathcal{E}_2$ consists of $M = n + 2K + 2$ equations in $N = 2K$ unknowns. Therefore, a proof π for this relation consists of

$$(K+1)2K + K(n + 2K + 2) = Kn + 4K^2 + 4K = Kn + O(1)$$

group elements. Lastly, the full secret-key of the final CLR-OWR, which contains a ciphertext c_2 of $\mathcal{E}_2$ and a proof π, consists of

$$(K+1)n + 4K^2 + 5K + 2 = (K+1)n + O(1)$$

group elements.
Assuming group elements can be represented optimally using $\log(p)$ bits[6], we therefore get relative leakage

$$\frac{\ell}{|sk|} = \frac{n\log(p) - \lambda}{((K+1)n + O(1))\log(p)} \geq \frac{\log(p)(n-1)}{((K+1)n + O(1))\log(p)} = \frac{n-1}{(K+1)n + O(1)}$$

Therefore, for any *constant* $\epsilon > 0$ there is a *constant* n such that $\frac{\ell}{|sk|} \geq \frac{1}{K+1} - \epsilon$.

Efficiency. Since K, n are always constants, the secret-key (c_2, π) consists of $Kn + 3K^2 + 3K = O(1)$ group elements. Furthermore, the creation of the ciphertext c_2 and the proof π, as well as the verification of (c_2, π), consists of $O(1)$ group operations (multiplications, exponentiations, pairings).

H The Groth-Sahai (GS) Proof System for General Equations

Note: this appendix is only used for the construction of efficient CLR-Signatures in Appendix J.

In Appendix F, we described the GS NIZK proof system for the simple case of linear equations. The full GS system [38] allows one to prove more complex systems of equations for which the variables could be both group elements and exponents. It also allows to prove the satisfiability of systems where each equation is of one of the following types: pairing product (PP) equations, multi-exponentiation (ME) in $\mathbb{G}$,

[6]We also assume, w.log. that $\log(p) \geq \lambda$.

multi-exponentiation in Γ. Note that the linear equations described in Appendix F are a simple form of the ME equations (with all constants d_i and $c_{i,j}$, defined below, equal to 0).

Next we present the language of satisfiable systems of equations containing equations of the types described above. As usual, we work in a pairing setting $(p, \mathbb{G}, \Gamma, \mathbb{G}_T, e, \mathbf{g}, \boldsymbol{\gamma}) \leftarrow \mathcal{G}_{pair}(1^\lambda)$, where $\mathbb{G}$ and Γ are the base groups of prime order p, $\mathbb{G}_T$ is the target group of order p, and e is an efficiently computable non-degenerate bilinear map $e : \mathbb{G} \times \Gamma \to \mathbb{G}_T$.

A system of equations is defined by statement elements in $\mathbb{G}, \Gamma$, and $\mathbb{Z}_p$ (described below). Different equations use (possibly but not necessarily) different statement elements. We say that a system of equations of the described types is *satisfiable* if there exist $\mathbf{x}_1, \ldots, \mathbf{x}_L \in \mathbb{G}$, $\boldsymbol{\chi}_1, \ldots, \boldsymbol{\chi}_N \in \Gamma$, $r_1, \ldots, r_{L'}$, $s_1, \ldots, s_{N'} \in \mathbb{Z}_p$ for which all equations are satisfied. We now describe each type of equation.

Pairing Product (PP) equations. A paring-product equation is defined by statement elements $\{\boldsymbol{\beta}\}_{i=1}^L \in \Gamma$, $\{\mathbf{b}_i\}_{i=1}^N \in \mathbb{G}$, and $\{(\mathbf{a}_i, \boldsymbol{\alpha}_i)\}_{i=1}^J$ for $J \in \mathbb{Z}^+$ with $J = \mathcal{O}(L+N)$. It has the form:

$$\prod_{i=1}^L e(\mathbf{x}_i, \boldsymbol{\beta}_i) \prod_{i=1}^N e(\mathbf{b}_i, \boldsymbol{\chi}_i) \prod_{i=1}^L \prod_{j=1}^N e(\mathbf{x}_i, \boldsymbol{\chi}_j) \prod_{i=1}^J e(\mathbf{a}_i, \boldsymbol{\alpha}_i) \;=\; \mathbf{1}_{\mathbb{G}_T}$$

Multi-Exponentiation (ME) equations in $\mathbb{G}$. A multi-exponentiation equation in $\mathbb{G}$ is defined by statement elements $\mathbf{a}$, $\{\mathbf{b}_i\}_{i=1}^{N'} \in \mathbb{G}$, $\{d_i, \{c_{i,j}\}_{j=1}^{N'}\}_{i=1}^L \in \mathbb{Z}_p$. It has the form:

$$\prod_{i=1}^{N'} \mathbf{b}_i^{s_i} \prod_{i=1}^L \mathbf{x}_i^{d_i} \prod_{i=1}^L \mathbf{x}_i^{\sum_{j=1}^{N'} c_{i,j} s_j} \;=\; \mathbf{a}$$

Multi-Exponentiation (ME) equations in Γ. A multi-exponentiation equation in Γ is defined by statement elements $\boldsymbol{\alpha}$, $\{\boldsymbol{\beta}_i\}_{i=1}^{L'} \in \Gamma$, $\{d_i, \{c_{i,j}\}_{j=1}^N\}_{i=1}^{L'} \in \mathbb{Z}_p$.

$$\prod_{i=1}^{L'} \boldsymbol{\beta}_i^{r_i} \prod_{i=1}^N \boldsymbol{\chi}_i^{d_i} \prod_{i=1}^N \boldsymbol{\chi}_i^{\sum_{j=1}^{L'} c_{i,j} r_j} \;=\; \boldsymbol{\alpha}$$

Theorem H.1. *[38] Under either SXDH or DLIN, a system of M PP equations and M' ME equations, in $\mathbb{G}$ or Γ, can be proved to be satisfiable in zero-knowledge. The proof size is $\mathcal{O}(M(N+L)+M'+L+N+L'+N')$.*

H.1 System of Simple Pairing-Product Equations

For our purposes, it is often nice to think of a system of simple pairing-product equations in *matrix form.* We define a system of M simple pairing-product equations over $MN + J'L$ variables as a set of four matrices: $\mathbf{B}_{\mathbb{G}} \in \mathbb{G}^{M \times L}, \mathbf{B}_\Gamma \in \Gamma^{N \times J'}, \mathbf{A}_{\mathbb{G}} \in \mathbb{G}^{M \times J}, \mathbf{A}_\Gamma \in \Gamma^{J \times J'}$.

We say that the system $(\mathbf{B}_{\mathbb{G}}, \mathbf{B}_\Gamma, \mathbf{A}_{\mathbb{G}}, \mathbf{A}_\Gamma)$ is *satisfiable* if there exist matrices $\mathbf{X}_{\mathbb{G}} \in \mathbb{G}^{M \times N}$ and $\mathbf{X}_\Gamma \in \Gamma^{L \times J'}$ such that:

$$(\mathbf{B}_{\mathbb{G}} \bullet \mathbf{X}_\Gamma) \boxplus (\mathbf{X}_{\mathbb{G}} \bullet \mathbf{B}_\Gamma) \boxplus (\mathbf{A}_{\mathbb{G}} \bullet \mathbf{A}_\Gamma) \;=\; \mathbf{J}_{\mathbb{G}_T},$$

where $\mathbf{J}_{\mathbb{G}_T} \in \mathbb{G}_T^{M \times J'}$ is the all-$\mathbf{1}_{\mathbb{G}_T}$ matrix.

We call the matrices $(\mathbf{X}_{\mathbb{G}}, \mathbf{X}_\Gamma)$ a *satisfying assignment* for the system. We define relation R_{pp} for all pairs of such systems and their satisfying assignments $((\mathbf{B}_{\mathbb{G}}, \mathbf{B}_\Gamma, \mathbf{A}_{\mathbb{G}}, \mathbf{A}_\Gamma), (\mathbf{X}_{\mathbb{G}}, \mathbf{X}_\Gamma))$. We define the corresponding language L_{pp} of satisfiable simple pairing-product equations.

It is easy to see that L_{pp} is a subset of the (general) pairing-product equations define previously (in particular, simple PP equations do not have any pairings of the type $e(\mathbf{x}_i, \boldsymbol{\chi}_j)$), for which GS is a good proof system.

I CLR Signatures : Generic Construction from CLR-OWR

In this section, we give a formal definition of continuous-leakage resilient signatures, and show that the construction in Figure 7 is secure under this definition. Because the construction in Figure 7 uses true-simulation extractable (tSE) NIZK arguments [20], we also include the definition of this primitive below.

Definition I.1. *We say that a signature scheme* $(\mathtt{KeyGen}, \mathtt{Sign}, \mathtt{SigVer}, \mathtt{ReRand})$ *is* ℓ-continuous-leakage-resilient (ℓ-CLR) *if for any PPT adversary* $\mathcal{A}$*, we have* $\Pr[\mathcal{A} \text{ wins }] \leq \mathsf{negl}(\lambda)$ *in the following game:*

- *The challenger chooses* $(vk, sk) \leftarrow \mathtt{KeyGen}(1^\lambda)$ *and gives* vk *to* $\mathcal{A}$*.*
- *The adversary* $\mathcal{A}$ *runs for arbitrarily many* leakage rounds*. In each round:*
  - *The adversary makes up to* ℓ *calls to a leakage-oracle* $\mathcal{O}^{\lambda}_{sk}(\cdot)$*, with the current secret key* sk*.*[7] *The adversary also makes queries to a signing oracle* $\mathcal{S}_{sk}(\cdot)$*. The signing oracle responds to a query* m *with a signature* σ *of* m *under the current secret key* $sk : \sigma = \mathtt{Sign}_{sk}(m)$*.*
  - *At the end of the round, the challenger samples* $sk' \leftarrow \mathtt{ReRand}(sk)$ *and updates* $sk := sk'$*.*
- *The adversary* wins *if it produces a message/signature pair* (m^*, σ^*) *such that* $\mathtt{SigVer}_{vk}(m^*, \sigma^*) = 1$ *and* m^* *was not given to* $\mathcal{S}_{sk}(\cdot)$ *as a signing query in any leakage round.*

Definition I.2 (True-Simulation Extractability [20])**.** *Let* $\Psi = (\mathtt{Setup}, \mathtt{Prov}, \mathtt{Ver}, \mathtt{Sim})$ *be an NIZK argument for an NP relation R, satisfying the completeness, soundness and zero-knowledge properties. We say that* Ψ *is* true-simulation extractable *(tSE) if:*

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

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

We are now ready to prove that the CLR signature construction in Figure 7 is secure, that is, that it complies with the requirements of Definition I.1.

Theorem I.3. *If* $\mathcal{C} = (\mathtt{KeyGen}^{\mathcal{C}}, \mathtt{ReRand}, \mathtt{Ver}^{\mathcal{C}})$ *is an* ℓ*-CLR-OWR, and* $\Psi = (\mathtt{Setup}^{\Psi}, \mathtt{Prov}, \mathtt{Ver}^{\Psi}, \mathtt{Sim}, \mathtt{Ext})$ *is a tSE-NIZK argument for relation* $R_{\mathcal{C}} = \{(y, x) \mid y = (pk, m), \quad x = sk \quad s.t. \quad \mathtt{Ver}^{\mathcal{C}}(pk, sk) = 1\}$*, then the signature scheme in Figure 7 is* ℓ*-CLR.*

Proof. We use a series-of-games argument to prove the above theorem.

Game 0: This is the original ℓ-CLR attack game described in Definition I.1, in which simulation queries are answered correctly by running $\sigma \leftarrow \mathtt{Prov}((pk, m), sk)$ and $\mathcal{A}$ wins if she produces a valid forgery (m^*, σ^*).

Game 1: In this game, the queries to the signing oracle are answered with simulated arguments: $\sigma \leftarrow \mathtt{Sim}_{\text{TK}}(pk, m)$. Games 0 and 1 are indistinguishable by the *zero-knowledge* property of the argument Ψ. Notice that the simulated arguments given to $\mathcal{A}$ as answers to signing queries are always of true statements.

[7] This is equivalent to a definition where, in each round, the adversary asks for a set of leakage functions whose output lengths sum up to at most ℓ bits.

Game 2: In this game, we modify the winning condition so that the adversary only wins if it produces a valid forgery (m^*, σ^*) *and* the challenger is able to extract a valid secret key sk^* for pk from (m^*, σ^*). That is, $\mathcal{A}$ wins if $\texttt{SigVer}(m^*, \sigma^*) = 1$ *and* $\texttt{Ver}^{\mathcal{C}}(pk, sk^*) = 1$, where $sk^* \leftarrow \texttt{Ext}_{\text{EK}}((pk, m^*), \sigma^*)$. The winning probability of $\mathcal{A}$ in Game 2 is at least that of Game 1 minus the probability that $\texttt{Ver}^{\Psi}((pk, m^*), \sigma^*) = 1 \wedge \texttt{Ver}^{\mathcal{C}}(pk, sk^*) = 0$. By the *true-simulation extractability* of the argument Ψ we know that this probability is negligible. Therefore, the winning probability of $\mathcal{A}$ in Game 2 differs from that in Game 1 by a negligible amount.

We have shown that the probability that $\mathcal{A}$ wins in Game 0 is the same as that in Game 2, up to negligible factors. We now argue that the probability that $\mathcal{A}$ wins in Game 2 is negligible, which proves that the probability that $\mathcal{A}$ wins in Game 0 is negligible as well. To prove that the probability that $\mathcal{A}$ wins in Game 2 is negligible, we assume otherwise and show that there exists a PPT algorithm $\mathcal{B}$ that breaks the *ℓ-continuous-leakage resilience* of $\mathcal{C}$. On input pk, $\mathcal{B}$ generates $(\text{CRS}, \text{TK}, \text{EK}) \leftarrow \texttt{Setup}^{\Psi}(1^{\lambda})$ and emulates $\mathcal{A}$ on input $vk = (\text{CRS}, pk)$. In each leakage round, $\mathcal{B}$ answers $\mathcal{A}$'s leakage queries using the leakage oracle $\mathcal{O}^{\lambda}_{sk}(\cdot)$ and answers signing queries m_i by creating simulated arguments $\texttt{Sim}_{\text{TK}}(pk, m_i)$. When $\mathcal{A}$ outputs her forgery (m^*, σ^*), $\mathcal{B}$ runs $sk^* \leftarrow \texttt{Ext}_{\text{EK}}((pk, m^*), \sigma^*)$ and outputs sk^*. Notice that $\Pr[\mathcal{B} \text{ wins}] = \Pr[\mathcal{A} \text{ wins}]$, so that if $\Pr[\mathcal{A} \text{ wins}]$ is non-negligible then $\mathcal{B}$ breaks the ℓ-continuous-leakage resilience of $\mathcal{C}$. We therefore conclude that the probability that $\mathcal{A}$ wins in Game 2 is negligible. This concludes the proof of the theorem. $\square$

We now know that the signature scheme in Figure 7 is CLR, but we have not proven that an instantiation of such signature scheme exists! This is because we have not shown that tSE NIZK can be achieved. Fortunately, Dodis et al. [20] show that we can indeed construct tSE NIZK arguments for any NP relation by combining any CCA-secure encryption scheme supporting labels with (standard) NIZK.

Theorem I.4. *[20] The existence of CCA-secure encryption, together with the existence of NIZK arguments for NP, implies the existence of true-simulation extractable NIZK arguments for NP.*

The following corollary follows from Theorem I.3 together with Theorem I.4.

Corollary I.5. *The existence of CCA-secure encryption, NIZK arguments for NP, and ℓ-CLR-OWRs with relative leakage $\frac{\ell}{|sk|} = \alpha$ implies the existence of ℓ-CLR Signatures with relative leakage $\frac{\ell}{|sk|} = \alpha$.*

Since NIZK proof systems for NP were shown to exist under either the DDH or DLIN assumptions (in the base groups of a pairing) by [37], and CCA-secure encryption was shown to exist under these assumptions by [16, 54], we also get the following corollary.

Corollary I.6. *Under either the DDH assumption or the DLIN assumption in the base group(s) of a pairing $\mathcal{G}_{pair}$, there exist ℓ-CLR signature schemes with relative leakage $\frac{\ell}{|sk|} \geq \frac{1}{K+1} - \epsilon$, where $K = 1$ in the case of DDH and $K = 2$ in the case of DLIN.*

J CLR Signatures : Efficient Instantiation Under K-Linear

In this section, we show an efficient instantiation of the CLR signature construction shown in Figure 7, under the K-Linear assumption. In order to show this, we will reduce our problem to showing that there exists an efficient tSE NIZK argument system (see Definition I.2) for (a special case of) the language of satisfiable systems of pairing-product equations (see Appendix H.1). We will therefore divide this section into two subsections. In Appendix J.1, we prove that if there exists an efficient tSE NIZK argument system for L_{pp}, then we have an efficient instantiation of the CLR signature scheme in Figure 7. In Appendix J.2 we show that efficient tSE NIZK argument systems for L_{pp} do indeed exist, and give a construction.

J.1 Expressing $R_{\mathcal{C}}$ as a System of Pairing-Product Equations

The construction in Figure 7 uses a CLR-OWR $\mathcal{C}$ and a tSE NIZK argument Ψ for $R_{\mathcal{C}}$. In Section 5 we showed how to instantiate $\mathcal{C}$ under K-Linear for $K = 1$ and in Appendix E we generalized the construction for $K \geq 1$. In this section we will show how to instantiate Ψ for general $K \geq 1$.

Using the instantiation of $\mathcal{C}$ given in Section 5, we can write $R_{\mathcal{C}}$ as follows:

$$R_{\mathcal{C}} = \left\{ (y, x) \ \mid \ y = ((pk_1, pk_2, \text{CRS}^{\Pi}, c_1), m) \ , \ x = (c_2, \pi) \ \text{ s.t. } \ \texttt{Ver}^{\Pi}((pk_1, pk_2, c_1, c_2), \pi) = 1 \right\},$$

where π is a Groth-Sahai argument that $c_1 = \texttt{Enc}^1_{pk_1}(w; r_1)$ and $c_2 = \texttt{Enc}^2_{pk_2}(w; r_2)$ for some values of w, r_1, r_2, and $\mathcal{E}_1$ and $\mathcal{E}_2$ are the Generalized ElGamal and Generalized Multiple CS-Lite encryption schemes, respectively. Recall from Appendix F that a Groth-Sahai argument consists of matrices $\mathbf{\Delta}, \mathbf{P}$ so that $\pi = (\mathbf{\Delta}, \mathbf{P})$. Further recall that to verify the argument π, $\texttt{Ver}^{\Pi}$ checks a system of equations. These equations are pairing-product equations of the form described above, where the statement $(\mathbf{B}_{\mathbb{G}}, \mathbf{B}_{\Gamma}, \mathbf{A}_{\mathbb{G}}, \mathbf{A}_{\Gamma})$ is defined by elements in $(\text{CRS}^{\Pi}, pk_1, pk_2, c_1, L)$, and the witness $(\mathbf{X}_{\mathbb{G}}, \mathbf{X}_{\Gamma})$ is defined by the ciphertext c_2 and the matrices $\mathbf{\Delta}, \mathbf{P}$. To see this, rewrite the verification equation as follows:

$$\mathbf{B} \bullet \mathbf{\Delta} = \left(\begin{array}{c|c} \mathbf{c}_1 & \mathbf{P} \\ \vdots & \\ \mathbf{c}_M & \end{array} \right) \bullet \mathbf{\Upsilon} \ = \ (\vec{\mathbf{c}} \bullet \boldsymbol{\tau}^{\top}) \boxplus (\mathbf{P} \bullet \mathbf{\Upsilon}')$$

where $\mathbf{\Upsilon} = \left(\dfrac{\vec{\boldsymbol{\tau}}^{\top}}{\mathbf{\Upsilon}'} \right)$. From Appendix E we see that $\vec{\mathbf{c}} = \vec{\mathbf{c}}_1 \boxplus \vec{\mathbf{c}}_2$, where $\vec{\mathbf{c}}_1$ depends solely on the ciphertext c_1 and $\vec{\mathbf{c}}_2$ depends solely on the ciphertext c_2. Therefore:

$$\mathbf{B} \bullet \mathbf{\Delta} = ((\vec{\mathbf{c}}_1 \boxplus \vec{\mathbf{c}}_2) \bullet \boldsymbol{\tau}^{\top}) \boxplus (\mathbf{P} \bullet \mathbf{\Upsilon}') = (\vec{\mathbf{c}}_1 \bullet \boldsymbol{\tau}^{\top}) \boxplus (\vec{\mathbf{c}}_2 \bullet \boldsymbol{\tau}^{\top}) \boxplus (\mathbf{P} \bullet \mathbf{\Upsilon}') = (\vec{\mathbf{c}}_1 \bullet \boldsymbol{\tau}^{\top}) \boxplus \left(\begin{array}{c|c} \mathbf{c}_{2,1} & \mathbf{P} \\ \vdots & \\ \mathbf{c}_{2,M} & \end{array} \right) \bullet \mathbf{\Upsilon}$$

Rewriting again, we get that:

$$\mathbf{J}_{\mathbb{G}_T} = (\vec{\mathbf{c}}_1 \bullet \boldsymbol{\tau}^{\top}) \boxplus \left(\begin{array}{c|c} \mathbf{c}_{2,1} & \mathbf{P} \\ \vdots & \\ \mathbf{c}_{2,M} & \end{array} \right) \bullet \mathbf{\Upsilon} \boxminus (\mathbf{B} \bullet \mathbf{\Delta}),$$

where $\mathbf{J}_{\mathbb{G}_T}$ is the matrix will all $\mathbf{1}_{\mathbb{G}_T}$ entries. This is a system of (simple) pairing product equations, where $\mathbf{B}_{\mathbb{G}}$ is defined by $\mathbf{B}, \mathbf{B}_{\Gamma} = \mathbf{\Upsilon}, \mathbf{A}_{\mathbb{G}} = \vec{\mathbf{c}}_1, \mathbf{A}_{\Gamma} = \boldsymbol{\tau}^{\top}, \mathbf{X}_{\mathbb{G}} = (\vec{\mathbf{c}_2} \mid \mathbf{P})$ and $\mathbf{X}_{\Gamma} = \mathbf{\Delta}$.

Therefore, if we show that there exists an efficient instantiation of tSE NIZK for the language L_{pp} described in Appendix H.1, then we automatically show an efficient instantiation of tSE NIZK for $R_{\mathcal{C}}$ and consequently, for the CLR signature construction in Figure 7. We show this in Appendix J.2 below.

J.2 tSE NIZK for a System of Pairing-Product Equations

Dodis et. al [20] show how to generically construct tSE NIZK from CCA-secure encryption and (standard) NIZK arguments. We give a brief description of their construction below. For full details, see [20].

Let R be any NP relation, and define

$$R' = \{ \ (y', x') \ \mid \ y' = (y, L) \ , \ x' = x \ \text{ s.t. } \ (y, x) \in R \ \}.$$

Let $\mathcal{E} = (\texttt{KeyGen}, \texttt{Enc}, \texttt{Dec})$ be a CCA-secure encryption scheme supporting labels. We write $\texttt{Enc}(\ldots : L)$ and $\texttt{Dec}(\ldots : L)$ to denote encryption and decryption under label L, respectively. Let $\Phi = (\texttt{Setup}^{\Phi}, \texttt{Prov}^{\Phi}, \texttt{Ver}^{\Phi})$ be an NIZK argument for the relation

$$R^{\Phi} = \{ \ (y^{\Phi}, x^{\Phi}) \ \mid \ y^{\Phi} = (y, c, pk, L) \ , \ x^{\Phi} = (x, r) \ , \ (y, x) \in R \wedge c = \texttt{Enc}_{pk}(x; r : L) \ \}.$$

$\texttt{Setup}(1^\lambda)$: Output $\text{CRS} = (\text{CRS}^\Phi, pk)$, $\text{TK} = \text{TK}^\Phi$, $\text{EK} = sk$ where
$(pk, sk) \leftarrow \texttt{KeyGen}(1^\lambda)$, $(\text{CRS}^\Phi, \text{TK}^\Phi) \leftarrow \texttt{Setup}^\Phi(1^\lambda)$.
$\texttt{Prov}((y, L), x; r)$: Output $\varphi = (c, \phi)$ where $c \leftarrow \texttt{Enc}_{pk}(x; r : L)$, $\phi \leftarrow \texttt{Prov}^\Phi((y, c, pk, L), (x, r))$.
$\texttt{Ver}((y, L), \varphi)$: Parse $\varphi = (c, \phi)$ and return $\texttt{Ver}^\Phi((y, c, pk, L), \phi)$.

Figure 11: Construction of tSE NIZK from CCA-secure encryption and (standard) NIZK

Then the argument system described in Figure 11 is a tSE NIZK argument system for the relation R'.

We will use the tSE NIZK construction of [20] described above to give an efficient instantiation of tSE NIZK for L_{pp}. We therefore need a CCA-secure encryption scheme $\mathcal{E} = (\texttt{KeyGen}, \texttt{Enc}, \texttt{Dec})$ and an NIZK argument Φ for the relation

$$R^\Phi = \left\{ \begin{array}{c} (y, x) \ \mid \ y = ((\mathbf{B}_\mathbb{G}, \mathbf{B}_\Gamma, \mathbf{A}_\mathbb{G}, \mathbf{A}_\Gamma), c, pk, L) \ , \ x = ((\mathbf{X}_\mathbb{G}, \mathbf{X}_\Gamma), r) \ \text{ s.t.} \\ \\ ((\mathbf{B}_\mathbb{G}, \mathbf{B}_\Gamma, \mathbf{A}_\mathbb{G}, \mathbf{A}_\Gamma), (\mathbf{X}_\mathbb{G}, \mathbf{X}_\Gamma)) \in R_{pp} \wedge c = \texttt{Enc}_{pk}((\mathbf{X}_\mathbb{G}, \mathbf{X}_\Gamma); r : L) \end{array} \right\}$$

For CCA-secure encryption, we use the Linear Cramer-Shoup scheme from [54], modified to support labels as in [12]. The scheme is described in Figure 12. We need two instances of the Linear Cramer-Shoup scheme, one for elements in $\mathbb{G}$ and another for elements in Γ. This is because the components of $\mathbf{X}_\mathbb{G}$ are elements in $\mathbb{G}$, while the components of $\mathbf{X}_\Gamma$ are elements in Γ. We encrypt each component of $\mathbf{X}_\mathbb{G}$ and each component of $\mathbf{X}_\Gamma$ separately, but all using the same label L. [8]

Let $K \in \mathbb{Z}^+$. Let $\mathsf{prms} = (H, p, \mathbb{G}, \mathbf{g}_0, \ldots, \mathbf{g}_K)$ where $(p, \mathbb{G}, \mathbf{g}_0) \leftarrow \mathcal{G}(1^\lambda)$, $\mathbf{g}_1, \ldots, \mathbf{g}_K \leftarrow \mathbb{G}$, and H is a collision-resistant hash function. The parameters implicitly define the $K \times (K+1)$ matrix:

$$\mathbf{A} = \begin{pmatrix} \mathbf{g}_0 & \mathbf{g}_1 & 1 & 1 & \ldots & 1 \\ \mathbf{g}_0 & 1 & \mathbf{g}_2 & 1 & \ldots & 1 \\ \mathbf{g}_0 & 1 & 1 & \mathbf{g}_3 & \ldots & 1 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{g}_0 & 1 & 1 & 1 & \ldots & \mathbf{g}_K \end{pmatrix}$$

$\texttt{KeyGen}(\mathsf{prms})$**:** Choose 3 random vectors $\vec{v}, \vec{w}, \vec{x} \leftarrow \mathbb{Z}_p^{(K+1)}$. Set $\vec{\mathbf{d}} := \mathbf{A} \cdot \vec{v}$, $\vec{\mathbf{e}} := \mathbf{A} \cdot \vec{w}$, $\vec{\mathbf{h}} := \mathbf{A} \cdot \vec{x}$.
Output $pk := (\vec{\mathbf{d}}, \vec{\mathbf{e}}, \vec{\mathbf{h}})$, $sk := (\vec{v}, \vec{w}, \vec{x})$.

$\texttt{Enc}_{pk}(\mathbf{m} : L)$**:** To encrypt a message $\mathbf{m} \in \mathbb{G}$ under label L, choose $\vec{r} \leftarrow \mathbb{Z}_p^K$ and set $\vec{\mathbf{y}} := \vec{r}^\top \cdot \mathbf{A}$.
Compute $\mathbf{a} := \langle \vec{\mathbf{h}}, \vec{r} \rangle$, $\mathbf{z} := \mathbf{am}$ and $\mathbf{c} := \langle \vec{\mathbf{d}} \boxplus (\vec{\mathbf{e}} \cdot t), \vec{r} \rangle$, where $t = H(\vec{\mathbf{y}}, \mathbf{z}, L)$. Output $c := (\vec{\mathbf{y}}, \mathbf{z}, \mathbf{c})$.

$\texttt{Dec}_{sk}(c)$**:** Parse $c = (\vec{\mathbf{y}}, \mathbf{z}, \mathbf{c})$. Compute $\tilde{\mathbf{c}} := \langle \vec{\mathbf{y}}, \vec{v} + t\vec{w} \rangle$. If $\tilde{\mathbf{c}} \stackrel{?}{=} \mathbf{c}$, then output $\mathbf{z} / \langle \vec{\mathbf{y}}, \vec{x} \rangle$. Else output $\perp$.

Figure 12: The Linear Cramer-Shoup Scheme (with Labels)

To instantiate the NIZK argument Φ, first notice that because of the form of the ciphertext components of a Linear Cramer-Shoup encryption, the sub-language

$$L_{CS} = \{ \ (y, x) \ \mid \ y = (c, pk, L) \ , \ x = (\mathbf{w}, \vec{r}) \ \text{ s.t. } c = \texttt{Enc}_{pk}(\mathbf{w}; \vec{r} : L) \}$$

can be expressed as a system of $(K+2)$ multi-exponentiation equations:

$$\mathbf{y}_i = \mathbf{g}_0^{\sum_{i=1}^K r_i} \mathbf{g}_i^{r_i} \quad \text{for } i = 1, \ldots, K \qquad , \qquad \mathbf{z} = \mathbf{w} \prod_{i=1}^K \mathbf{h}_i^{r_i} \quad , \quad \mathbf{c} = \prod_{i=1}^K (\mathbf{d}_i, \mathbf{e}_i^t)^{r_i}$$

[8] We note that encrypting each component of a plaintext separately does not necessarily conserve CCA-security. This problem can be solved by using a strong one-time signature to sign the resulting ciphertexts. More precisely, the encryptor generates a signature key pair (vk, sk), attaches vk to the label when encrypting each component, and signs all resulting ciphertexts. In our instantiation, we can use the strong one-time signature of [36], secure under the Discrete Log assumption (implied by K-Linear). Because of the small signature size of this scheme, namely 2 elements in $\mathbb{Z}_p$, this does not affect the overall efficiency of our scheme.

Therefore, the language L_Φ can be expressed as a system of equations containing a combination of pairing-product equations, multi-exponentiation equations in $\mathbb{G}$, and multi-exponentiation equations in Γ. As described in Appendix H, the Groth-Sahai argument system [38] can be used to prove that such a system is satisfiable. Moreover, such a proof has size $O(1)$ and requires $O(1)$ exponentiation/pairing operations.

Theorem J.1. *Fix a constant $K \geq 1$ and assume that the K-Linear assumption holds in the base group(s) of some pairing $\mathcal{G}_{pair}$. Let $(\mathbf{B}_\mathbb{G}, \mathbf{B}_\Gamma, \mathbf{A}_\mathbb{G}, \mathbf{A}_\Gamma)$ be a system of M pairing-product equations with N unknowns. Then there exists a tSE NIZK argument system for proving that $(\mathbf{B}_\mathbb{G}, \mathbf{B}_\Gamma, \mathbf{A}_\mathbb{G}, \mathbf{A}_\Gamma) \in L_{pp}$ that uses proofs of size $O(1)$ and requires $O(1)$ exponentiation/pairing operations.*

Combining the generic construction of CLR signatures in Figure 7 with Theorem 5.3 and Theorem J.1, yields the following theorem, which proves (part of) Theorem 6.1.

Theorem J.2. *Fix any $K \geq 1$, 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 exists an ℓ-CLR signature scheme with relative leakage $\frac{\ell}{|sk|} \geq \frac{1}{K+1} - \epsilon$.*

K CLR Identification (ID) Schemes & CLR Authenticated Key Agreement (AKA)

K.1 CLR ID Schemes

In this section, we construct a CLR Identification (ID) scheme from our construction of CLR signatures. We begin by giving a definition of CLR ID schemes.

Definition K.1 (CLR-ID). *We say that an ID scheme* $(\texttt{KeyGen}, \texttt{ReRand}, \mathcal{P}, \mathcal{V})$ *is CLR if it satisfies the following properties:*

Correctness: *In an interaction between an honest prover and an honest verifier* $\{\mathcal{P}(pk, sk) \rightleftharpoons \mathcal{V}(pk)\}$*, the verifier* $\mathcal{V}$ *always* accepts *the execution.*

Security: *The winning probability of any PPT adversary* $\mathcal{A}$ *is* $\Pr[\mathcal{A} \text{ wins }] \leq \mathsf{negl}(\lambda)$ *in the following game:*

Initialization: *The challenger chooses* $(pk, sk) \leftarrow \texttt{KeyGen}(1^\lambda)$ *and gives* pk *to* $\mathcal{A}$.

Learning Stage: *The learning stage proceeds in arbitrary many leakage rounds. In each round, the adversary* $\mathcal{A}$ *can arbitrarily interleave the following actions:*

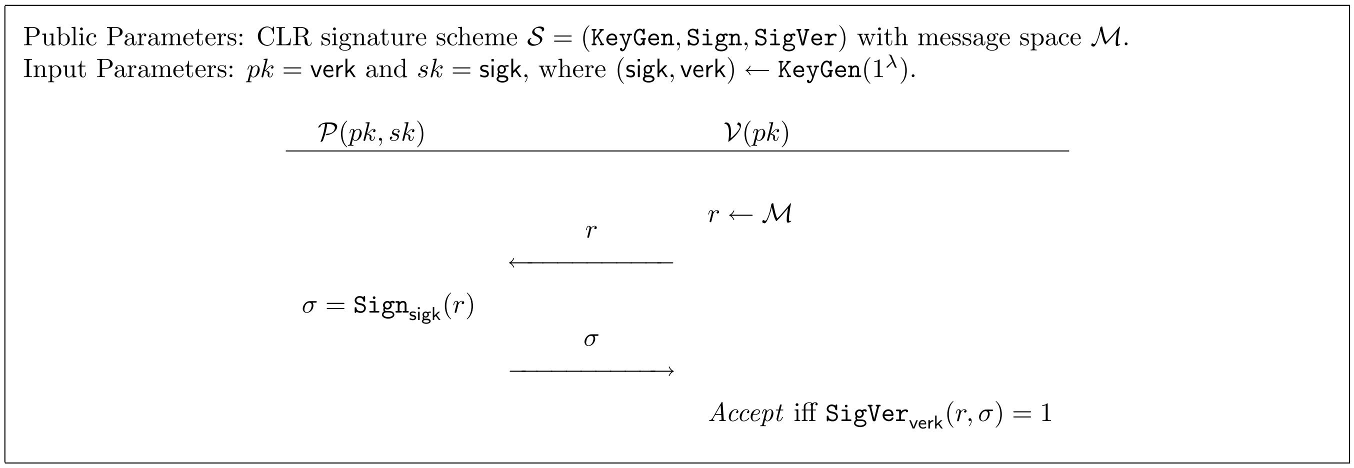
- *Protocol Executions: The adversary* $\mathcal{A}$ *can execute arbitrarily many protocols with the honest prover* $\mathcal{P}(sk)$*, who uses fresh random coins for each execution.*
- *Leakage Queries: In each round, the adversary* $\mathcal{A}$ *can make up to* ℓ *queries to the oracle* $\mathcal{O}^\lambda_{sk}$*, on the current secret key* sk.

At the end of each round, the challenger computes $sk' \leftarrow \texttt{ReRand}(sk)$ *and updates* $sk := sk'$.

Impersonation Stage: *The adversary* $\mathcal{A}$ *interacts with an honest verifier* $\mathcal{V}(pk)$*, by taking the role of the prover (there is* no *leakage at this point). We say that* $\mathcal{A}$ *wins if* $\mathcal{V}$ accepts *the execution.*

Construction. Our CLR ID scheme naturally extends the construction of [3] to the continuous-leakage setting. In the ID scheme, the prover simply uses a CLR signature scheme to sign a message randomly chosen by the verifier. The scheme is described in Figure 13.

Theorem K.2. *Let* $\mathcal{S} = (\texttt{KeyGen}, \texttt{Sign}, \texttt{SigVer})$ *be an* ℓ*-CLR signature scheme with message space* $\mathcal{M}$ *such that* $|\mathcal{M}| = 2^{\omega(\lambda)}$. *Then the ID scheme of Figure 13 is an* ℓ*-CLR ID scheme.*


Public Parameters: CLR signature scheme $\mathcal{S} = (\texttt{KeyGen}, \texttt{Sign}, \texttt{SigVer})$ with message space $\mathcal{M}$.
Input Parameters: $pk = \mathsf{verk}$ and $sk = \mathsf{sigk}$, where $(\mathsf{sigk}, \mathsf{verk}) \leftarrow \texttt{KeyGen}(1^\lambda)$.

$\mathcal{P}(pk, sk)$
$\mathcal{V}(pk)$
$r \leftarrow \mathcal{M}$
r
$\sigma = \texttt{Sign}_{\mathsf{sigk}}(r)$
σ
Accept iff $\texttt{SigVer}_{\mathsf{verk}}(r, \sigma) = 1$

Figure 13: CLR-ID Scheme from CLR Signatures

Proof. The correctness property is trivial. For security, assume there exists an adversary $\mathcal{A}$ breaking the security of the CLR ID scheme with non-negligible probability ϵ. We build $\mathcal{B}$ that breaks the CLR security of the signature scheme as follows. During the learning stage, in every leakage round, $\mathcal{B}$ answers to $\mathcal{A}$'s leakage queries by using the leakage oracle $\mathcal{O}^{\lambda}_{\mathsf{sigk}}$. In every protocol execution, $\mathcal{B}$ simulates the prover by using the signature oracle $\mathcal{S}_{\mathsf{sigk}}(\cdot)$ to responds to $\mathcal{A}$'s challenges. During the impersonation stage, $\mathcal{B}$ samples $r^* \leftarrow \mathcal{M}$ and sends r^* to $\mathcal{A}$. $\mathcal{A}$ responds with a signature σ^*, and $\mathcal{B}$ outputs the message/signature pair (r^*, σ^*).

Let $q(\lambda)$ be an upper bound on the number of protocol executions $\mathcal{A}$ performs during the impersonation stage, where q is some polynomial. Let $r_1, \ldots, r_{q(\lambda)}$ be the challenges that $\mathcal{A}$ sent during the learning stage. Let E be the event that $r^* = r_i$ for some $i = 1, \ldots q(\lambda)$. Then:

$$\Pr[\mathcal{B} \text{ succeeds}] = \Pr[\neg E \wedge \mathcal{A} \text{ succeeds}] = \Pr[\mathcal{A} \text{ succeeds}] - \Pr[\mathcal{A} \text{ succeeds} \mid E] \Pr[E] = \epsilon - \Pr[E]$$

And we have that

$$\Pr[E] \leq \sum_{i=1}^{q(\lambda)} \Pr[r^* = r_i] = \frac{q(\lambda)}{2^{\omega(\lambda)}} \leq negl(\lambda),$$

since q is a polynomial. Therefore, if ϵ is non-negligible, then so is $\Pr[\mathcal{B} \text{ succeeds}]$. Since we assumed that $\mathcal{S}$ was a CLR signature scheme, then we must have that $\epsilon \leq negl(\lambda)$. This completes the proof. ☐

Combining Theorem K.2 with Theorem J.2 yields the following theorem, which proves (part of) Theorem 6.1.

Theorem K.3. *Fix any $K \geq 1$, 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 exists an ℓ-CLR ID scheme with relative leakage $\frac{\ell}{|sk|} \geq \frac{1}{K+1} - \epsilon$.*

K.2 CLR AKA

Using our CLR signature scheme from Figure 7 (and instantiating it as described in Appendix J), we construct a CLR authenticated key agreement (AKA) scheme. Our scheme achieves *perfect forward security* in the unauthenticated-links model with erasures [14] (with the modifications of [3]). We refer the reader to [14, 3] for a detailed description of the model and definitions of security, but give a high level of the problem and our solution below. Note, however, that we slightly modify the security definitions from [14, 3] to make them applicable to the continuous-leakage setting.

Model and Security Definitions. We consider the problem of exchanging a cryptographic key between two parties, Alice and Bob, in the presence of a "man-in-the-middle" adversary. Moreover, Alice and Bob not only want to be sure that the privacy of the key is preserved, but also that they are indeed communicating with each other (and not with a malicious third party). In the continuous-leakage setting, the "man-in-the-middle" adversary not only has the ability to read and intercept messages between the Alice and Bob (and between Alice and/or Bob and other parties), but is also able to learn unbounded and arbitrary information about Alice and Bob's long-term secret keys. On the other hand, Alice and Bob are able to periodically and inconspicuously *update* their long-term secret keys, so that the adversary is limited to learn at most ℓ bits of information about any one particular secret key (though it might learn an unbounded amount of information in total).

We model this scenario more formally by considering an adversary that plays against concurrent sessions of the protocol between n players $\mathcal{P}_1, \ldots, \mathcal{P}_n$, with corresponding long-term secret keys $sk_1, \ldots, sk_n$. The adversary can schedule key exchanges between players and choose the players that will participate in each session. She also has the ability to corrupt players, learn their ephemeral states, and learn arbitrary information about their long-term secret keys. This last ability is modeled by providing the adversary with n leakage oracles $\mathcal{O}^{\lambda}_{sk_1}, \ldots, \mathcal{O}^{\lambda}_{sk_n}$. The goal of the adversary is to learn the session key of a test session of her choice, between players $\mathcal{P}_i$ and $\mathcal{P}_j$ also of her choice. We do not allow the adversary to corrupt $\mathcal{P}_i$ or $\mathcal{P}_j$ or learn their ephemeral states during the test session, and we restrict her access to the n leakage oracles as follows:

- The adversary can only perform queries to the leakage oracles *before* the test session (but not during).
- Each leakage oracle $\mathcal{O}^{\lambda}_{sk}$ maintains an internal counter that keeps track of how many queries the adversary has asked the oracle. When the counter reaches ℓ, the oracle $\mathcal{O}^{\lambda}_{sk}$ automatically updates sk and sets the internal counter to 0. This models leakage rounds in the continuous-leakage setting, allowing the adversary to call each leakage oracle an unbounded number of times, but only learn ℓ bits of information about any specific secret key sk. Note that modifying the leakage oracles in this way is the only change we make to the model and security definitions of [14, 3].

Finally, we require *perfect forward security*, which guarantees that the session key remains secure even if the adversary learns the entire long-term secret keys of $\mathcal{P}_i, \mathcal{P}_j$ *after* the test session has been completed and the session key has been deleted from memory.

Generic Construction. Our construction takes any passively-secure key agreement protocol achieving perfect forward security (such as the Diffie-Hellman key exchange) and authenticates it using our CLR signature scheme from Figure 7. For simplicity, we assume that the passively-secure key agreement is a 2-round protocol (which is the case for the most widely used protocols). We use the following five algorithms to denote a 2-round key agreement protocol, and informally describe the correctness and passive security properties:

- $\mathsf{prms} \leftarrow \mathtt{Setup}(1^{\lambda})$: Outputs public parameters prms.
- $(\alpha, s_1) \leftarrow \mathtt{RoundOne}(\mathsf{prms})$: Takes in public parameters prms and outputs a value α and an ephemeral state s_1.
- $(\beta, s_2) \leftarrow \mathtt{RoundTwo}(\mathsf{prms}, \alpha)$: Takes in prms, and α, and outputs a value β and an ephemeral state s_2
- $k_1 := \mathtt{KeyOne}(\mathsf{prms}, s_1, \alpha, \beta)$: Takes in prms, the values α, β and the ephemeral state s_1 and outputs a key k_1.
- $k_2 := \mathtt{KeyTwo}(\mathsf{prms}, s_2, \alpha, \beta)$: Takes in prms, the values α, β and the ephemeral state s_2 and outputs a key k_2.

Correctness guarantees that $k = \mathtt{KeyOne}(\mathsf{prms}, s_1, \alpha, \beta) = \mathtt{KeyTwo}(\mathsf{prms}, s_2, \alpha, \beta)$ where $\mathsf{prms} \leftarrow \mathtt{Setup}(1^{\lambda})$, $(\alpha, s_1) \leftarrow \mathtt{RoundOne}(\mathsf{prms})$, and $(\beta, s_2) \leftarrow \mathtt{RoundTwo}(\mathsf{prms}, \alpha)$. *Passive security* guarantees that k is indistinguishable from a randomly chosen key.

Using the notation described above, we present our generic CLR-AKA scheme in Figure 14. Our construction uses a CLR signature scheme $\mathcal{S} = (\mathtt{KeyGen}, \mathtt{Sign}, \mathtt{SigVer})$ and passively-secure key agreement protocol $\Pi = (\mathtt{Setup}, \mathtt{RoundOne}, \mathtt{RoundTwo}, \mathtt{KeyOne}, \mathtt{KeyTwo})$ achieving perfect forward security.

Theorem K.4. *Let $\mathcal{S} = (\mathtt{KeyGen}, \mathtt{Sign}, \mathtt{SigVer})$ be an ℓ-CLR signature scheme and let $\Pi = (\mathtt{Setup}, \mathtt{RoundOne}, \mathtt{RoundTwo}, \mathtt{KeyOne}, \mathtt{KeyTwo})$ be a passively-secure key agreement protocol achieving perfect forward security.*

Public Parameters: $\mathsf{prms} \leftarrow \mathtt{Setup}(1^\lambda)$
Common Input: signature verification keys $(\mathsf{verk}_i, \mathsf{verk}_j)$

Initiator $\mathcal{P}_i(\mathsf{sigk}_i)$		Responder $\mathcal{P}_j(\mathsf{sigk}_j)$
$(\alpha, s_1) \leftarrow \mathtt{RoundOne}(\mathsf{prms})$ register session $(\mathcal{P}_j, \alpha)$	$\xrightarrow{\mathcal{P}_i, \alpha}$	
	$\xleftarrow{\mathcal{P}_j, \beta, \sigma_j}$	$(\beta, s_2) \leftarrow \mathtt{RoundTwo}(\mathsf{prms}, \alpha; r_2)$ $\sigma_j = \mathtt{Sign}_{\mathsf{sigk}_j}(\mathcal{P}_i, \mathcal{P}_j, \alpha, \beta)$ register session $(\mathcal{P}_i, \alpha, \beta)$
$\sigma_i = \mathtt{Sign}_{\mathsf{sigk}_i}(\mathcal{P}_j, \mathcal{P}_i, \alpha, \beta)$ output peer $= \mathcal{P}_j$, sid $= (\alpha, \beta)$, output session key $k_i \leftarrow \mathtt{KeyOne}(s_1, \alpha, \beta)$ delete s_1	$\xrightarrow{\mathcal{P}_i, \sigma_i}$	
mark session complete		output peer $=\mathcal{P}_j$, sid$=(\alpha, \beta)$ output session key $k_j \leftarrow \mathtt{KeyTwo}(s_2, \alpha, \beta)$ delete s_2 mark session complete

Sanity Checks:

1. If σ_j is not a valid signature of $(\mathcal{P}_i, \mathcal{P}_j, \alpha, \beta)$, under verk_j then $\mathcal{P}_i$ ignores the message. Similarly for σ_i and $\mathcal{P}_j$.
2. If $\mathcal{P}_i$ receives a round-2 message σ_j of $(\mathcal{P}_i, \mathcal{P}_j, \alpha, \beta)$ but has not registered a session $(\mathcal{P}_j, \alpha)$ then ignore the message. Similarly $\mathcal{P}_j$ and σ_i.

Figure 14: CLR-AKA Scheme from CLR Signatures and Passively-Secure Key Agreement with Perfect Forward Security

Then the protocol in Figure 14 is an ℓ-CLR authenticated key agreement protocol with perfect forward security in the unauthenticated-links model.

The proof of Theorem K.4 is analogous to the one presented in [3] for the protocol $\mathsf{eSig\text{-}DH}$. Informally, it satisfies *perfect forward security* because after the session is completed, learning the signing keys $\mathsf{sigk}_i, \mathsf{sigk}_j$ does not give the adversary any new information about the key that was exchanged.

Instantiation under K-Linear. Let $\mathcal{E} = (\mathtt{KeyGen}, \mathtt{Enc}, \mathtt{Dec})$ be a CPA-secure encryption scheme and consider the following 2-round passively-secure key agreement protocol: $\mathtt{RoundOne}$ runs $(pk, sk) \leftarrow \mathtt{KeyGen}$ and outputs $(\alpha := pk, s_1 := sk)$, $\mathtt{RoundTwo}$ samples random k, computes $c \leftarrow \mathtt{Enc}_{pk}(k)$, deletes the randomness used for encryption, and outputs $(\beta := c, s_2 := k)$. $\mathtt{KeyOne}$ computes $k := \mathtt{Dec}_{sk}(c)$, deletes sk and outputs k, and $\mathtt{KeyTwo}$ simply outputs k. Correctness follows directly from the correctness of $\mathcal{E}$ and passive security follows from the CPA-security of $\mathcal{E}$. Perfect forward security is trivially achieved because sk is deleted by $\mathtt{KeyOne}$ and the randomness used in encryption is deleted by $\mathtt{RoundTwo}$ as soon as the ciphertext is created. Therefore, there are no long-term secret keys to be revealed.

Notice that we can instantiate the above key agreement construction under the K-linear assumption using the Generalized ElGamal encryption scheme described in Section E.1. This, combined with Theorem K.4 and Theorem J.2 yields the following theorem, which proves (part of) Theorem 6.1.

Theorem K.5. *Fix any $K \geq 1$, 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 exists an ℓ-CLR AKA scheme achieving perfect forward security with relative leakage $\frac{\ell}{|sk|} \geq \frac{1}{K+1} - \epsilon$.*

L Noisy Leakage

L.1 Background on Entropy

The *min-entropy* of a random variable X is

$$\mathbf{H}_\infty(X) \stackrel{\text{def}}{=} -\log(\max_x \Pr[X = x]).$$

This is a standard notion of entropy used in cryptography, since it measures the worst-case predictability of X. We also review a generalization from [22], called *average conditional min-entropy* defined by

$$\widetilde{\mathbf{H}}_\infty(X|Z) \stackrel{\text{def}}{=} -\log\left(\mathop{\mathbb{E}}_{z\leftarrow Z}\left[\ \max_x \Pr[X = x|Z = z]\ \right]\right) = -\log\left(\mathop{\mathbb{E}}_{z\leftarrow Z}\left[2^{-\mathbf{H}_\infty(X|Z=z)}\right]\right).$$

This measures the worst-case predictability of X by an adversary that may observe a correlated variable Z. Sometimes, it is easier to talk about predictability rather than entropy, and we define

$$\mathbf{Pred}(X \mid Z) \stackrel{\text{def}}{=} 2^{-\widetilde{\mathbf{H}}_\infty(X \mid Z)} = \mathop{\mathbb{E}}_{z\leftarrow Z}\left[\ \max_x \Pr[X = x|Z = z]\ \right] \tag{5}$$

It is easy to see that for any (unbounded) adversary $\mathcal{A}$ that "sees" the outcome of Z and tries to predict the outcome of a correlated variable X, we have $\Pr[\mathcal{A}(Z) = X] \leq \mathbf{Pred}(X \mid Z)$. Conversely, the optimal unbounded strategy $\mathcal{A}$ does achieve $\Pr[\mathcal{A}(Z) = X] = \mathbf{Pred}(X \mid Z)$. We will also use the following lemma.

Lemma L.1 ([22])**.** *Let X, Y, Z be random variables where Y takes on values in a set of size at most 2^ℓ. Then $\widetilde{\mathbf{H}}_\infty(X|(Y,Z)) \geq \widetilde{\mathbf{H}}_\infty((X,Y)|Z) - \ell \geq \widetilde{\mathbf{H}}_\infty(X|Z) - \ell$ and, in particular, $\widetilde{\mathbf{H}}_\infty(X|Y) \geq \mathbf{H}_\infty(X) - \ell$.*

The above lemma shows that, if $h : \{0,1\}^* \to \{0,1\}^\ell$ is a length-bounded function, then for any random variable X, we have $\widetilde{\mathbf{H}}_\infty(X \mid h(X)) \geq \mathbf{H}_\infty(X) - \ell$. In particular, if f is length-bounded with ℓ bit output, then f is ℓ-leaky, as defined in Definition 7.4.

L.2 Properties of Noisy Leakage

We now show that our definition of noisy leakage (Definition 7.4) has several nice properties which make it particularly easy to work with.

First we show that, if a function h decreases the entropy of the *uniform* distribution by at most ℓ bits, then it decreases the entropy of *every* distribution by at most ℓ bits. In other words, the uniform distribution always gives us the *worst possible* entropy degradation.

Lemma L.2. *Assume a probabilistic function $h : \{0,1\}^* \to \{0,1\}^*$ is ℓ-leaky. Then, for* every *random variable X over $\{0,1\}^n$, we have $\widetilde{\mathbf{H}}_\infty(X \mid h(X)) \geq \mathbf{H}_\infty(X) - \ell$.*

Proof. Assume otherwise, that X is some distribution over $\{0,1\}^n$ such that $\mathbf{H}_\infty(X) = t$ and $\widetilde{\mathbf{H}}_\infty(X \mid h(X)) < t - \ell$. Also assume that h uses m random coins. That means there is some predictor P for which $\Pr[P(h(X; U_m)) = X] > 2^{\ell - t}$. Call a pair $(x, r) \in \{0,1\}^{n+m}$ "good" if $P(h(x; r)) = x$. Then there must be $> 2^{\ell+m}$ good pairs, since the probability of any given pair occurring is $\leq 2^{-(t+m)}$. Therefore $\Pr[P(h(U_n; U_m)) = U_n] = \Pr[(U_n, U_m) \text{ is "good"}] > 2^{\ell+m}/2^{n+m} = 2^{\ell-n}$. Hence $\widetilde{\mathbf{H}}_\infty(U_n \mid h(U_n)) < n - \ell$, which contradicts h being ℓ-leaky.

□

Next we show that any strategy which adaptively applies many ℓ_i-leaky functions, only learns $\sum_i \ell_i$ bits of information at the end.

Lemma L.3. *Let X be an arbitrary random variable over $\{0,1\}^n$. Assume that $\mathcal{A}$ is an arbitrary (unbounded) adversarial strategy which adaptively chooses probabilistic functions $h_i : \{0,1\}^* \to \{0,1\}^*$ and learns $h_i(X)$. Assume each h_i is ℓ_i-leaky, and that the function h_i always satisfy $\sum_i \ell_i \leq \ell$. Let $\mathcal{A}(x)$ be the view os $\mathcal{A}$ at the end of the above game with input x. Then $\widetilde{\mathbf{H}}_\infty(X \mid \mathcal{A}(X)) \geq \mathbf{H}_\infty(X) - \ell$.*

Proof. Let us first prove the lemma in the case that $\mathcal{A}$ can only ask for two functions h_1, h_2, where h_1 is ℓ_1-leaky and h_2 is $(\ell-\ell_1)$-leaky, but can be chosen adaptively based on the answer to h_1. The view of $\mathcal{A}$ consists of $(z_1, z_2) = (h_1(X), h_2(X))$. For any value z_1 define the conditional distribution $X_{z_1} = (X \mid h_1(X) = z_1)$. We then have:

$$\begin{aligned}
\mathbf{Pred}(X \mid \mathcal{A}(X)) &= \mathop{\mathbb{E}}_{(z_1,z_2)\leftarrow\mathcal{A}(X)} \max_{x\in\{0,1\}^n} \Pr[X = x \mid \mathcal{A}(X) = (z_1, z_2)] \\
&\leq \mathop{\mathbb{E}}_{z_1\leftarrow h_1(X)} \max_{h_2} \mathop{\mathbb{E}}_{z_2\leftarrow h_2(X_{z_1})} \max_{x} \Pr[X = x \mid h_1(X) = z_1, h_2(X) = z_2] \\
&= \mathop{\mathbb{E}}_{z_1\leftarrow h_1(X)} \max_{h_2} \left(\mathop{\mathbb{E}}_{z_2\leftarrow h_2(X_{z_1})} \max_{x} \Pr[X_{z_1} = x \mid h_2(X_{z_1}) = z_2] \right) \\
&= \mathop{\mathbb{E}}_{z_1\leftarrow h_1(X)} \max_{h_2} \mathbf{Pred}(X_{z_1} \mid h_2(X_{z_1})) \\
&\leq \mathop{\mathbb{E}}_{z_1\leftarrow h_1(X)} \mathbf{Pred}(X_{z_1}) 2^{\ell-\ell_1} \qquad (6) \\
&= 2^{\ell-\ell_1} \mathbf{Pred}(X \mid h_1(X)) \\
&\leq 2^{\ell} \mathbf{Pred}(X) \qquad (7)
\end{aligned}$$

where (6) (respectively (7)) follows from the fact that h_2 is $\ell - \ell_1$-leaky (respectively, h_1 is ℓ_1-leaky) and Lemma L.2. To generalize to the case where $\mathcal{A}$ can ask for many functions, we just use induction on the number of functions m. We just showed the case of $m = 1, 2$. Assume the lemma holds for m, and we show that it holds for $m + 1$. The main idea is that, after seing the answer to the first ℓ_1-leaky query $h_1(X)$, we can define the rest of the strategy of $\mathcal{A}$ as a function $\mathcal{A}_2(X)$, which consists of m adaptively chosen queries $h_2, \ldots, h_{m+1}$. By the inductive hypothesis, we see that $\mathcal{A}_2$ is $(\ell - \ell_1)$-leaky. The theorem then follows from the $m = 2$ case. □

L.3 Resilience to Noisy Leakage (Proof-Sketch of Theorem 7.5)

DEFINITIONS. Defining resilience to noisy leakage is slightly more difficult then length-bounded leakage, since there is no efficient procedure to check how leaky a function h is. We define the noisy-leakage oracle $\mathcal{NO}^{\lambda}_{sk}(\cdot)$, analogously to Definition 2.2, except that the query functions $h : \{0,1\}^* \to \{0,1\}^*$ can have arbitrary output lengths (not just 1 bit). In the definition of CLR primitives, we will not restrict the *number of queries* to the oracle, but rather the sum of the "leakiness" of the queries. For example, we define a noisy CLR-OWR as follows.

Definition L.4 (Noisy-Leakage Security for CLR-OWR)**.** *We say that an ℓ-CLR-OWR is* secure against noisy leakage, *if it satisfies the definition of CLR-OWR (Definition 2.3) with the following modifications:*

1. *In each leakage round, the adversary can adaptively make an* arbitrary number *of oracle-calls to $\mathcal{NO}^{\lambda}_{sk}(\cdot)$.*
2. *The adversary only wins if, in each leakage-round i, the set of queries $h_{i,1}, \ldots, h_{i,m_i}$ made to $\mathcal{NO}^{\lambda}_{sk}(\cdot)$ satisfy $\sum_{j=1}^{m_i} \ell_{i,j} \leq \ell$, where each $h_{i,j}$ is $\ell_{i,j}$-leaky.*
   (Notice that the winning condition cannot be checked efficiently.)

We can define the other noisy CLR primitives (signatures, ID schemes, AKA) analogously. We can also analogously define noisy variants of our building blocks for the CLR-OWR construction: a noisy LIRR and a noisy LoC-NM encryption. To prove Theorem 7.5, we notice that our web of reductions preserves noisy leakage. In particular, it it easy to extend our results to get the following:

Theorem 4.9 noisy LoC-NM encryption	$\Rightarrow$	noisy LIRR.
Theorem 3.2 noisy LIRR	$\Rightarrow$	noisy CLR-OWR.
Theorem I.3: noisy CLR-OWR	$\Rightarrow$	noisy CLR-signatures
Theorem K.2, Theorem K.4: noisy CLR-signature	$\Rightarrow$	noisy CLR-ID schemes and noisy CLR-AKA

Therefore, to prove Theorem 7.5, we only need to show that the construction of the encryption scheme $\mathcal{E}_2$ is ℓ-LoC-NM w.r.t. noisy leakage. But this follows almost directly from the proof of Theorem E.2.

The only place in that proof where the nature of the leakage is used, is in arguing the indistinguishability of Games 2 and 3. In particular, it is used in the analysis of case 2, arguing that a decryption query with $\vec{\mathbf{y}}^* \notin \texttt{rowspace}(\mathbf{A})$ is likely to decrypt to $\perp$ in Game 2. To do so, we used the fact that the correct verification elements $\tilde{\mathbf{c}}_i^*$ corresponding to $\vec{\mathbf{y}}^*$ are uniformly random given all other information except for the leakage. Then, the nature of the leakage was only used to argue that $\widetilde{\mathbf{H}}_\infty(\tilde{\mathbf{c}}_1^*, \ldots, \tilde{\mathbf{c}}_n^* \mid \text{ leakage }) \geq \lambda$, and so the adversary can only predict these values with negligible probability. In the case of length bounded leakage, the above inequality follows from Lemma L.1. For the case of noisy-leakage, the above inequality still holds by Lemma L.3.

L.4 Comparison to the Definition of [49]

Naor and Segev [49], introduced the concept of noisy leakage, but with a slightly different definition. That is, if a scheme has distribution PK, SK on public/secret keys than a leakage-function h is ℓ-leaky if $\widetilde{\mathbf{H}}_\infty(SK \mid PK, h(SK)) \geq \widetilde{\mathbf{H}}_\infty(SK \mid PK) - \ell$. We find this definition less robust since the "leakiness" of a function h depends on the scheme within which this function is analyzed, rather than just on the function itself. However, by Lemma L.2, our measure of noisy leakiness is always more pessimistic and so security under the [49] definition is a stronger notion. Interestingly, our schemes do *not* satisfy it. In particular, our secret-key sk for CLR-OWR does not, in itself, have large statistical entropy but only pseudo-entropy. Therefore, the identity function $h(sk) = sk$, would be legal leakage for our scheme according to the definition of [49], but *not* according to our Definition 7.4.

M Leakage of Computation

M.1 Organization

In Appendix M.2, we give some background on Σ-protocols. In Appendix M.3, we then give general constructions of CLR security for ID schemes, signatures and AKA from CLR-OWRs and Σ-protocols, and show that these constructions satisfy *leakage-of-computation* security. Lastly, in Appendix M.4, we show how to *efficiently* instantiate Σ-protocols for the concrete relations we need, using our concrete instantiation of a CLR-OWR.

This gives us several alternative constructions of the above primitives, differing from those in Appendix I and Appendix K. The main advantage is that the current constructions satisfy *leakage-of-computation* security. Interestingly, they are also *more efficient*. The main disadvantage, for the signatures and AKA protocols, is that they are only proven secure in the *random oracle model*. Also, the relative-leakage of the signature scheme (but not the ID scheme and AKA protocol) is halved.

M.2 Background: Σ-Protocols

Definition. Let R be an NP relation consisting of *instance, witness* pairs $(y, x) \in R$ and let $L_R = \{y \mid \exists x, (y, x) \in R\}$ be the corresponding *language* of $\mathcal{R}$. A Σ-protocol for R is a protocol between a PPT ITM prover $\mathcal{P}(y, x)$ and a PPT ITM verifier $\mathcal{V}(x)$, which proceeds in three rounds where: (1) the prover $\mathcal{P}(y, x)$ sends an initial message a, (2) the verifier $\mathcal{V}(y)$ sends a uniformly random challenge c from some space C (3) the prover $\mathcal{P}(y, x)$ sends a response z. The verifier $\mathcal{V}(x)$ either *accepts* or *rejects* the conversation by computing some predicate $\texttt{Ver}(y, a, c, z)$ of the instance y and the conversation (a, c, z). We require that Σ-protocols satisfy the following three properties:

Perfect Completeness: For any $(y, x) \in R$, the execution $\{\mathcal{P}(y, x) \rightleftharpoons \mathcal{V}(y)\}$ is always accepting.

Special Soundness: There is an efficient extractor algorithm such that, given an instance y and two accepting conversations for y: (a, c, z), (a, c', z') with the same first message a but different challenges

$c \neq c'$, the algorithm outputs x' such that $(y, x') \in R$.

Perfect Honest Verifier Zero Knowledge (HVZK): There is a PPT simulator $\mathcal{S}$ such that, for any $(y, x) \in R$, the simulator $\mathcal{S}(y)$ produces conversations (a, c, z) which are *identically distributed* to the conversations produced by an execution with an *honest* verifier $\{\mathcal{P}(y, x) \rightleftharpoons \mathcal{V}(y)\}$.

Although it often makes sense to talk about Σ-protocols with even a 1-bit challenge space $C = \{0, 1\}$, we will (from now on) assume that the ***challenge-space is super-polynomial*** in the security parameter: $|C| = 2^{\omega(\log(n))}$. This can always be made to hold, using parallel repetition.

PROPERTIES OF Σ-PROTOCOLS. As was shown in [15], the HVZK property implies *witness indistinguishability*: For any adversarial verifier $\mathcal{V}^*$ any $(y, x), (y, x') \in R$ the conversations:

$$\{\mathcal{P}(y, x) \rightleftharpoons \mathcal{V}^*(y)\} \quad \equiv \quad \{\mathcal{P}(y, x') \rightleftharpoons \mathcal{V}^*(y)\}$$

are identically distributed. This also implies that, for any joint distribution (Y, X) over R and any adversary $\mathcal{V}^*$, if we condition on any outcome $Y = y$, then the random variables X and $Z = (\text{view of } \mathcal{V}^*)$ are independently distributed.

Although *computationally-indistinguishable HVZK* Σ-protocols are known to exists for all of NP, this is unlikely to be the case with perfect HVZK Σ-protocols [15], which is our default notion. Therefore, we will need to explicitly give a Σ-protocol for our concrete relations later.

M.3 Leakage of Randomness Security: Generic Constructions

Let $\mathcal{C} = (\texttt{KeyGen}, \texttt{ReRand}, \texttt{Ver}^{\mathcal{C}})$ be a CLR-OWR. Let $R_c = \{(pk, sk) \mid \texttt{Ver}^{\mathcal{C}}(pk, sk) = 1\}$ be a relation on valid pk, sk pairs of the CLR-OWR, and let $\Sigma = (\mathcal{P}, \mathcal{V})$ be a Σ-protocol for the relation $R_{\mathcal{C}}$.[9] In the rest of the section, we give generic constructions of several CLR primitives with leakage-of-computation security, from $\mathcal{C}$ and Σ.

M.3.1 ID Schemes

Definition M.1 (CLR-ID with Leakage of Randomness)**.** *We say that an ID scheme* $(\texttt{KeyGen}, \texttt{ReRand}, \mathcal{P}, \mathcal{V})$ *is a CLR-ID with leakage-of-computation security if the regular correctness property holds (see Definition K.1) and the winning probability of any PPT adversary* $\mathcal{A}$ *is* $\Pr[\mathcal{A} \text{ wins }] \leq \mathsf{negl}(\lambda)$ *in the following game:*

Initialization: *The challenger chooses* $(pk, sk) \leftarrow \texttt{KeyGen}(1^\lambda)$ *and gives pk to* $\mathcal{A}$*. It sets r to be the empty string.*

Learning Stage: *The learning stage proceeds in arbitrary many leakage rounds. In each round, the adversary* $\mathcal{A}$ *can arbitrarily interleave the following actions:*

- *Protocol Executions: The adv.* $\mathcal{A}$ *can execute arbitrarily many protocols with the honest prover* $\mathcal{P}(sk; r_i)$*, who uses fresh random coins* r_i*. In the beginning of each such execution, the challenger sets* $r := r||r_i$ *(i.e. appends* r_i *to r).*
- *Leakage Queries: In each round, the adv.* $\mathcal{A}$ *can make up to* ℓ *queries to the oracle* $\mathcal{O}^\lambda_{sk,r}$*, on the current secret key sk and all the randomness r used so far in the current round.*

At the end of each round, the challenger resets r to the empty string, computes $sk' \leftarrow \texttt{ReRand}(sk)$ *and updates* $sk := sk'$*.*

Impersonation Stage: *The adversary* $\mathcal{A}$ *interacts with an honest verifier* $\mathcal{V}(pk)$*, by taking the role of the prover (there is* no *leakage at this point). We say that* $\mathcal{A}$ *wins if* $\mathcal{V}$ accepts *the execution.*

[9]Recall, we assume that a Σ-protocol is *perfect* HVZK.

We want to say that, if $\mathcal{C} = (\texttt{KeyGen}, \texttt{ReRand}, \texttt{Ver}^{\mathcal{C}}), \Sigma = (\mathcal{P}, \mathcal{V})$ are define as before, then ($\texttt{KeyGen}$, $\texttt{ReRand}$, $\mathcal{P}$, $\mathcal{V}$) is a secure CLR-ID scheme with leakage-of-computation security. Unfortunately, we do not know how to prove this statement in general (even without leakage-of-computation security). The problem is that executions of a Σ-protocol cannot be efficiently simulated, even though they do not (information theoretically) reveal any additional information about the secret key. The way to get around this problem is to make a stronger assumption on $\mathcal{C}$, that it satisfies CLR security with respect to *noisy leakage* (Definition 7.4). That way, we can actually include all of the protocol executions as outputs of a leakage function on sk.

Theorem M.2. *Assume that $\mathcal{C} = (\texttt{KeyGen}, \texttt{ReRand}, \texttt{Ver}^{\mathcal{C}})$ is an $(\ell+1)$-CLR-OWR with noisy-leakage security and that $\Sigma = (\mathcal{P}, \mathcal{V})$ is a Σ-protocol for the relation $R_{\mathcal{C}}$. Then $\mathcal{I} = (\texttt{KeyGen}, \mathcal{P}, \mathcal{V})$ is an ℓ-CLR-secure ID scheme with leakage-of-computation security.*

Proof. Let $\mathcal{A}$ be an adversary against the ID scheme, with noticeable probability of winning. We construct an adversary $\mathcal{B}$ for the CLR-OWR as follows. We can define the leakage function $h_{\mathcal{A},pk}(x)$ which does the following: (1) checks that $\texttt{Ver}^{\mathcal{C}}(pk, x) = 1$ and, if not, it outputs $\perp$, (2) otherwise, it simply runs $\mathcal{A}$ for a single leakage round, using $sk = x$ to perfectly simulate the the prover $\mathcal{P}(pk, sk; r)$ and the leakage-oracle $\mathcal{O}^{\lambda}_{sk,r}$, outputting the entire view of $\mathcal{A}$ at the end. We first prove the following:

Claim M.3. *For any fixed pk and algorithm $\mathcal{A}$, the function $h_{\mathcal{A},pk}(x)$ is $(\ell+1)$-leaky.*

Proof. Let X be any random variable and let us define the random-variables $B = \texttt{Ver}^{\mathcal{C}}(pk, X)$ to be a binary indicator variable for whether X is a valid secret-key for pk, P to be the outcome of all interactions between $\mathcal{A}$ and the prover $\mathcal{P}(pk, X; R)$ with uniform randomness R, and L to be the responses to all of $\mathcal{A}$'s the leakage-oracle queries to $\mathcal{O}^{\lambda}_{X,R}$ on the secret-key X and randomness R. Then

$$\begin{aligned}
& \mathbf{Pred}(X|h_{\mathcal{A},pk}(X)) \\
= \; & \mathbf{Pred}(X|h_{\mathcal{A},pk}(X), B=1)\Pr[B=1] + \mathbf{Pred}(X|h_{\mathcal{A},pk}(X), B=0)\Pr[B=0] && (8)\\
\leq \; & \mathbf{Pred}(X|P, L, B=1)\Pr[B=1] + \mathbf{Pred}(X|B=0)\Pr[B=0] && (9)\\
\leq \; & 2^{\ell}\mathbf{Pred}(X|P, B=1)\Pr[B=1] + \mathbf{Pred}(X|B=0)\Pr[B=0] && (10)\\
\leq \; & 2^{\ell}\mathbf{Pred}(X|B=1)\Pr[B=1] + \mathbf{Pred}(X|B=0)\Pr[B=0] && (11)\\
\leq \; & 2^{\ell}\mathbf{Pred}(X|B) \leq \mathbf{Pred}(X)2^{\ell+1} && (12)
\end{aligned}$$

Where (8) follows by the definition of predictability and the laws of expectation, (9) follows by decomposing the output of the function $h_{\mathcal{A},pk}(x)$ depending on the outcome of B, (10) follows since the queries to the leakage oracle are cumulatively at most ℓ-leaky (and Lemma L.3), (11) follows since interaction with the prover $\mathcal{P}$ is perfectly witness indistinguishable (and hence P reveals 0 information conditioned on $B = 1$) and (12) follows by the definition of predictability again.

By substituting $X = U_n$, we get $\widetilde{\mathbf{H}}_{\infty}(U_n|h_{\mathcal{A},pk}(U_n)) = -\log(\mathbf{Pred}(U_n|h_{\mathcal{A},pk}(U_n)) = n - (\ell+1)$, which proves the claim. □

So, the reduction $\mathcal{B}$ breaking the CLR-OWR simply takes the description of $\mathcal{A}$ (including it's random-coins and current state) at the beginning of each leakage-round and queries its oracle $\mathcal{O}_{sk}$ with the leakage-function $h_{\mathcal{A},pk}$. It then feeds the output to its copy of $\mathcal{A}$ to progress to the next leakage round. This goes on until $\mathcal{A}$ moves to the impersonation stage.

When that occurs, $\mathcal{B}$ simply runs as an honest verifier with $\mathcal{A}$ to get a conversation (a, c, z). It then rewinds $\mathcal{A}$ to the challenge phase and sends a fresh challenge c'. Using the standard rewinding argument, there is a noticeable probability that $\mathcal{B}$ recovers two valid conversations $(a, c, z)(a, c', z')$ with $c \neq c'$, and therefore a witness sk^* such that $\texttt{Ver}^{\mathcal{C}}(pk, sk^*) = 1$. Therefore $\mathcal{B}$ wins the CLR-OWR attack game with noticeable probability. □

M.3.2 Signatures

Definition M.4 (CLR-Sig with Leakage of Randomness)**.** *We say that a signature scheme* ($\mathtt{KeyGen}$, $\mathtt{Sign}$, $\mathtt{SigVer}$, $\mathtt{ReRand}$) *is ℓ-CLR with* leakage-of-computation security *if for any PPT adversary $\mathcal{A}$, we have* $\Pr[\mathcal{A}$ *wins* $] \le \mathsf{negl}(\lambda)$ *in the following game:*

- *The challenger chooses $(vk, sk) \leftarrow \mathtt{KeyGen}(1^\lambda)$ and gives vk to $\mathcal{A}$.*
- *The adversary $\mathcal{A}$ runs for arbitrarily many* leakage rounds*. In the beginning of each round r is initialized to be the empty string. Then, in each round:*
  - *The adversary makes up to ℓ calls to a leakage-oracle $\mathcal{O}_{sk,r}$, with the current secret key sk and the randomness r used so far. The adversary also makes queries to a signing oracle $\mathcal{S}_{sk}(\cdot)$. The signing oracle responds to a query m with the signature $\sigma = \mathtt{Sign}_{sk}(m; r_i)$ under fresh random coins r_i, and the challenger updates $r = r||r_i$.*
  - *At the end of the round, the challenger samples $sk' \leftarrow \mathtt{ReRand}(sk)$ and updates $sk := sk'$. It resets r to the empty string.*
- *The adversary* wins *if it produces a message/signature pair (m^*, σ^*) such that $\mathtt{SigVer}_{vk}(m^*, \sigma^*) = 1$ and m^* was not given to $\mathcal{S}_{sk}(\cdot)$ as a signing query in any leakage round.*

Let $\mathcal{C} = (\mathtt{KeyGen}, \mathtt{ReRand}, \mathtt{Ver}^{\mathcal{C}})$ be a CLR-OWR and $\Sigma = (\mathcal{P}, \mathcal{V})$ be a Σ-protocol for $R_{\mathcal{C}}$. We can then use the Fiat-Shamir [29] transform to define a signature scheme $\mathcal{S} = (\mathtt{KeyGen}, \mathtt{Sign}, \mathtt{SigVer}, \mathtt{ReRand})$ as shown in Figure 15.

Let $H : \{0,1\}^* \to C$ be a Random Oracle mapping arbitrary messages to the challenge-space C of the Σ-protocol.

$\mathtt{KeyGen}$, $\mathtt{ReRand}$: Same as for the OWR.

$\mathtt{Sign}_{sk}(m)$: Use the prover $\mathcal{P}(pk, sk)$ to compute the first message a of the protocol. Compute the challenge $c := H(a, m)$ and give c to $\mathcal{P}(pk, sk)$ to get a last message z. Output (a, z).

$\mathtt{SigVer}_{pk}(m, \sigma)$: Parse $\sigma = (a, z)$. Compute $c = H(a, m)$. Output 1 if the conversation (a, c, z) is accepting for pk.

Figure 15: Fiat-Shamir Transform [29]: Signatures from Σ-Protocols

Note on Leakage in the RO Model. As noted by [46], when analyzing leakage in the RO model, the leakage functions $h(\cdot)$ should have access to the random-oracle. This captures the essence of the random-oracle model, where *all parties*, honest or adversarial, have access to the oracle. We assume this to be the case in the following theorem.

Theorem M.5. *Assume that $\mathcal{C} = (\mathtt{KeyGen}, \mathtt{ReRand}, \mathtt{Ver}^{\mathcal{C}})$ is an $(\ell + 1)$-CLR-OWR with noisy-leakage security and concurrent-leakage-round security. Assume that $\Sigma = (\mathcal{P}, \mathcal{V})$ is a Σ-protocol for the relation $R_{\mathcal{C}}$. Then the Fiat-Shamir scheme $\mathcal{S} = (\mathtt{KeyGen}, \mathtt{Sign}, \mathtt{SigVer}, \mathtt{ReRand})$, as defined in Figure 15 is an $\ell/2$-CLR-secure (existentially-unforgeable) signature scheme with leakage-of-computation security.*

Proof. Assume that $\mathcal{A}$ is an attacker on the signature scheme with noticeable success probability. Also, assume that $\mathcal{A}$ runs for at most q_L leakage round and makes at most q_H calls to the random oracle H (including calls made by the leakage functions). We define an attacker $\mathcal{B}$ against the concurrent/noisy-leakage security of the CLR-OWR. The attacker $\mathcal{B}$ works as follows. It chooses a $2(q_H)$-wise independent function f_k (keyed by a random k) with range $\mathcal{C}$[10] and a random $c' \in \mathcal{C}$. It uses these to define a leakage-function $h_{\mathcal{A},pk,k,c'}(sk_1, \dots, sk_{q_L})$ which concurrently leaks on q_L versions of the CLR-OWR secret keys. The function works as follows:

Check that Secret Keys Are Legal: If $\mathtt{Ver}(pk, sk_i) = 0$ for any i, output $\perp$.

[10]We can also replace f_k by an appropriate PRF.

Run $\mathcal{A}$: Run the adversary $\mathcal{A}$. Answer all random-oracle queries z via $f_k(z)$. Answer all signing queries on m honestly by running the honest prover $\mathcal{P}$ with challenge $c = f_k(a, m)$ secret key sk_i and uniform randomness r_j. Let the set P be all the answers to the signing queries. Answer all leakage queries on the secret-keys sk_i and randomness r_j honestly. Let L be the answers to all leakage queries.

Rewind Once: At the end, $\mathcal{A}$ outputs some value $(m_1^*, \sigma_1^* = (a_1^*, z_1^*))$. Rewind all the way to the beginning and run $\mathcal{A}$ again, answering random-oracle queries, signing queries and leakage queries as before, except that, on random-oracle query (m^*, a^*) output c'. If $\mathcal{A}$ makes any new signing/leakage queries, update the set L, P accordingly. At the end $\mathcal{A}$ outputs some value $(m_2^*, \sigma_2^* = (a_2^*, z_2^*))$.

Output: Output the responses to the leakage/signing queries L, P.

Now, when $\mathcal{B}$ gets L, P, it can use them to run $\mathcal{A}$ the same way as was done inside the leakage function (using L, P instead of the secret-keys), eventually getting $(m_1^*, \sigma_1^* = (a_1^*, z_1^*))$, $(m_2^*, \sigma_2^* = (a_2^*, z_2^*))$. Using the standard rewinding argument, there is a noticeable probability that $(a_1^*, c_1^* = f_k(m_1^*, a_1^*), z_1^*)$ and $(a_2^*, c_2^* = c', z_2^*)$ are two accepting conversations with $a_1^* = a_2^*$ and $c_1^* \neq c_2^*$. In this case $\mathcal{B}$ can come up with a valid secret key sk using the special soundness property.

So all we are left to prove is that $h_{\mathcal{A},pk,k,c'}$ is $(\ell + 1)$-leaky. But this follows from the same argument as Claim M.3.

□

M.3.3 "Challenge-Response" Interactive Signatures

We can also define a weaker notion of security for signatures, called challenge-response unforgeability, where the adversary only wins the game if she is able to forge a signature of a message of the type (z, m^*) where z is chosen by the challenger from some appropriately random distribution, *after all adversarial leakage queries have been made.*[11] In particular, the adversary's leakage cannot depend on z. We notice that, for an analogue of Theorem M.5 is simpler to prove for entropically-unforgeable signature. In particular, since the oracle-query $(a^*, (z, m^*))$ for the forgery is made only *after* all leakage queries have been made, $\mathcal{B}$ does not make any new leakage queries after rewinding during the extraction stage. Therefore we can allow ℓ bits of leakage, as compared to $\ell/2$ in the previous scheme. Moreover, we do not need concurrent security from the CLR-OWR any longer.

Theorem M.6. *Assume that $\mathcal{C} = (\texttt{KeyGen}, \texttt{ReRand}, \texttt{Ver}^{\mathcal{C}})$ is an ℓ-CLR-OWR with* noisy *leakage security. Assume that $\Sigma = (\mathcal{P}, \mathcal{V})$ is a Σ-protocol for the relation $R_{\mathcal{C}}$. Then the Fiat-Shamir scheme $\mathcal{S} = (\texttt{KeyGen}, \texttt{Sign}, \texttt{SigVer}, \texttt{ReRand})$, as defined in Figure 15 is an ℓ-CLR-secure* entropically unforgeable *signature scheme with leakage-of-computation security.*

M.3.4 AKA

Here, we just notice that in our construction of AKA from CLR-Sigs, we only require *entropic unforgeability*, since, to mount an attack, the adversary has to forge a signature for a message that partially depends on the (random) protocol messages of the other party. This same observation was made in [3].

M.4 A Σ-Protocol for the Secret-Key of a CLR-OWR

Let $\mathcal{C} = (\texttt{KeyGen}, \texttt{ReRand}, \texttt{Ver}^{\mathcal{C}})$ be our concrete instantiation of a CLR-OWR based on the schemes $\mathcal{E}_1, \mathcal{E}_2, \Pi$ under the K-linear assumption. Let $R_c = \{(pk, sk) \mid \texttt{Ver}^{\mathcal{C}}(pk, sk) = 1\}$ be an NP relation.

Recall that $pk = (\text{CRS}, pk_1, pk_2, c_1)$ and $sk = (c_2, \pi)$ where $\texttt{Ver}^{\mathcal{C}}(pk, sk) = 1$ if and only if

$$\texttt{Ver}^{\Pi}_{\text{CRS}}((pk_1, pk_2, c_1, c_2), \pi) = 1$$

[11]The work of [3] gave a somewhat more general definition called *entropic unforgeability.*

where $\mathtt{Ver}^{\Pi}$ is the verification procedure of the NIZK Π for the plaintext-equality relation R_{eq}.[12]

Also recall that $\mathtt{Ver}^{\Pi}_{\text{CRS}}((pk_1, pk_2, c_1, c_2), \pi) = 1$ iff

$$\mathbf{B} \bullet \mathbf{\Delta} = \left(\vec{\mathbf{c}} \mid \mathbf{P} \right) \bullet \mathbf{\Upsilon}$$

where $\mathbf{\Upsilon}$ depends on the CRS, $\mathbf{B}$ depends only on (pk_1, pk_2), $\vec{\mathbf{c}}$ depends on c_1, c_2, and $(\mathbf{\Delta}, \mathbf{P}) = \pi$. However, we can also write $\vec{\mathbf{c}} = \vec{\mathbf{c}}_1 \boxplus \vec{\mathbf{c}}_2$ where $\vec{\mathbf{c}}_1$ depends only on c_1 and $\vec{\mathbf{c}}_2$ depends only on c_2 (see Appendix E.3 for why that is).

Therefore, we can write $pk = (\text{CRS}, pk_1, pk_2, c_1) = (\mathbf{\Upsilon}, \mathbf{B}, \vec{\mathbf{c}}_1)$, and secret-keys $sk = (c_2, \pi) = (\vec{\mathbf{c}}_2, \mathbf{\Delta}, \mathbf{P})$ such that:

$$\begin{array}{lll} (pk, sk) \in R_{\mathcal{C}} & \text{iff} & \mathbf{B} \bullet \mathbf{\Delta} = \left(\vec{\mathbf{c}}_1 \boxplus \vec{\mathbf{c}}_2 \mid \mathbf{P} \right) \bullet \mathbf{\Upsilon} \\ & \text{iff} & \mathbf{B} \bullet \mathbf{\Delta} = \left(\left(\vec{\mathbf{c}}_2 \mid \mathbf{P} \right) \boxplus \left(\vec{\mathbf{c}}_1 \mid \{1_{\mathbb{G}}\} \right) \right) \bullet \mathbf{\Upsilon} \\ & \text{iff} & \mathbf{B} \bullet \mathbf{\Delta} \boxminus \left(\left(\vec{\mathbf{c}}_2 \mid \mathbf{P} \right) \bullet \mathbf{\Upsilon} \right) = \left(\vec{\mathbf{c}}_1 \mid 1_{\mathbb{G}} \right) \bullet \mathbf{\Upsilon} \end{array}$$

Therefore each $pk = (\mathbf{\Upsilon}, \mathbf{B}, \vec{\mathbf{c}}_1)$ defines a *group homomorphism*

$$\rho_{pk} \quad : \quad (\vec{\mathbf{c}}_2, \mathbf{\Delta}, \mathbf{P}) \quad \mapsto \quad \mathbf{B} \bullet \mathbf{\Delta} \boxminus \left(\left(\vec{\mathbf{c}}_2 \mid \mathbf{P} \right) \bullet \mathbf{\Upsilon} \right)$$

and we want a Σ-protocol to prove the knowledge of a pre-image of the element $\left(\vec{\mathbf{c}}_1 \mid 1_{\mathbb{G}} \right) \bullet \mathbf{\Upsilon}$ under the above homomorphism. We now use a general strategy for building efficient Σ-protocols for the pre-image of a group homomorphism (see [47] for an elegant general description of this strategy). Our protocol is shown in Figure 16.

Let $pk = (\text{CRS}, pk_1, pk_2, c_1) = (\mathbf{\Upsilon}, \mathbf{B}, \vec{\mathbf{c}}_1)$, $\quad sk = (c_2, \pi) = (\vec{\mathbf{c}}_2, \mathbf{\Delta}, \mathbf{P})$.
Define $\mathbf{T} = \left(\vec{\mathbf{c}}_1 \mid 1_{\mathbb{G}} \right) \bullet \mathbf{\Upsilon}$.

First Message $\mathcal{P}(pk, sk)$: Choose a random $\mathsf{aux} = (\vec{\mathbf{c}}'_2, \mathbf{\Delta}', \mathbf{P}')$ in the same domain as sk. Output $\mathbf{A} = \rho_{pk}(\mathsf{aux})$.

Challenge $\mathcal{V}(pk)$: Choose a random challenge $c \leftarrow \mathbb{Z}_p$.

Response $\mathcal{P}(pk, sk, \mathsf{aux})$: Output $\mathbf{Z} = (\vec{\mathbf{c}}^*_2, \mathbf{\Delta}^*, \mathbf{P}^*)$ where

$$\begin{array}{rcl} \vec{\mathbf{c}}^*_2 & := & \vec{\mathbf{c}}'_2 \boxplus (\vec{\mathbf{c}}_2 \cdot c) \\ \mathbf{\Delta}^* & := & \mathbf{\Delta}' \boxplus (\mathbf{\Delta} \cdot c) \\ \mathbf{P}^* & := & \mathbf{P}' \boxplus (\mathbf{P} \cdot c) \end{array}$$

Verify Conversation $\mathcal{V}(pk, (\mathbf{A}, c, \mathbf{Z}))$: To verify the conversation $(\mathbf{A}, c, \mathbf{Z})$, the verifier checks $\rho_{pk}(\mathbf{Z}) \overset{?}{=} \mathbf{A} \boxplus (\mathbf{T} \cdot c)$.

Figure 16: A Σ-protocol for $R_{\mathcal{C}}$.

Theorem M.7. *The protocol in Figure 16 is a Σ-protocol (with perfect completeness, special soundness and perfect HVZK) for the relation $R_{\mathcal{C}}$.*

Proof. We show the thee properties one-by-one.

Perfect Completeness: Assume that the conversation $(\mathbf{A}, c, \mathbf{Z})$ comes from an honest prover where $\mathsf{aux} = (\vec{\mathbf{c}}'_2, \mathbf{\Delta}', \mathbf{P}')$ and $\mathbf{Z} = (\vec{\mathbf{c}}^*_2, \mathbf{\Delta}^*, \mathbf{P}^*)$. Then

$$\begin{array}{rcl} \rho_{pk}(\mathbf{Z}) & = & \mathbf{B} \bullet \mathbf{\Delta}^* \boxminus \left(\left(\vec{\mathbf{c}}^*_2 \mid \mathbf{P}^* \right) \bullet \mathbf{\Upsilon} \right) \\ & = & \mathbf{B} \bullet \mathbf{\Delta}' \boxminus \left(\left(\vec{\mathbf{c}}'_2 \mid \mathbf{P}' \right) \bullet \mathbf{\Upsilon} \right) \quad \boxplus \quad \mathbf{B} \bullet (\mathbf{\Delta} \cdot c) \boxminus \left(\left((\vec{\mathbf{c}}_2 \cdot c) \mid (\mathbf{P} \cdot c) \right) \bullet \mathbf{\Upsilon} \right) \\ & = & \rho_{pk}(\mathsf{aux}) \boxplus (\rho_{pk}(sk) \cdot c) \\ & = & \mathbf{A} \boxplus (\mathbf{T} \cdot c) \end{array}$$

[12] It's important, at this point, to remember the difference between R_{eq} and $R_{\mathcal{C}}$. For R_{eq}, the statements are (pk_1, pk_2, c_1, c_2) and we want to prove that c_1, c_2 encrypt the same plaintext. For $R_{\mathcal{C}}$ the statement is $pk = (\text{CRS}, pk_1, pk_2, c_1)$, the witness is (c_2, π), and we want to prove that π is a valid GS NIZK for the relation R_{eq} with statement (pk_1, pk_2, c_1, c_2).

Special Soundness: Assume that $(\mathbf{A}, c, \mathbf{Z}), (\mathbf{A}, \tilde{c}, \tilde{\mathbf{Z}})$ are two accepting conversations where $c \neq \tilde{c}$. We write
$\mathbf{Z} = (\vec{\mathbf{z}}_1, \mathbf{Z}_2, \mathbf{Z}_3), \tilde{\mathbf{Z}} = (\tilde{\vec{\mathbf{z}}}_1, \tilde{\mathbf{Z}}_2, \tilde{\mathbf{Z}}_3)$. Then

$$\left\{ \begin{array}{l} \rho_{pk}(\mathbf{Z}) = \mathbf{A} \boxplus (\mathbf{T} \cdot c) \\ \rho_{pk}(\tilde{\mathbf{Z}}) = \mathbf{A} \boxplus (\mathbf{T} \cdot \tilde{c}) \end{array} \right\} \begin{array}{l} \Rightarrow \quad \rho_{pk}(\mathbf{Z}) \boxminus \rho_{pk}(\tilde{\mathbf{Z}}) = \mathbf{T} \cdot (c - \tilde{c}) \\ \Rightarrow \quad \rho_{pk}(\vec{\mathbf{z}}_1 \boxminus \tilde{\vec{\mathbf{z}}}_1, \mathbf{Z}_2 \boxminus \tilde{\mathbf{Z}}_2, \mathbf{Z}_3 \boxminus \tilde{\mathbf{Z}}_3) = \mathbf{T} \cdot (c - \tilde{c}) \\ \Rightarrow \quad \rho_{pk}(sk^*) = \mathbf{T} \end{array}$$

Where

$$sk^* = \left(\quad (\vec{\mathbf{z}}_1 \boxminus \tilde{\vec{\mathbf{z}}}_1) \cdot \frac{1}{(c - \tilde{c})} \quad , \quad (\mathbf{Z}_2 \boxminus \tilde{\mathbf{Z}}_2) \cdot \frac{1}{(c - \tilde{c})} \quad , \quad (\mathbf{Z}_3 \boxminus \tilde{\mathbf{Z}}_3) \cdot \frac{1}{(c - \tilde{c})} \quad \right)$$

Therefore the extractor can efficiently compute the above sk^* such that $(pk, sk^*) \in R_{\mathcal{C}}$.

Perfect HVZK: The simulator samples $c \leftarrow Z_p$ at random. It then chooses $\mathbf{Z} = (\vec{\mathbf{c}}_2^*, \mathbf{\Delta}^*, \mathbf{P}^*)$ uniformly at random from the same domain as sk. Lastly it computes $\mathbf{A} := \rho_{pk}(\mathbf{Z}) \boxminus (\mathbf{T} \cdot c)$ and outputs $(\mathbf{A}, c, \mathbf{Z})$.

Perfect simulation follows since honest and simulated conversations are both uniformly random subject only to the constraint: $\rho_{pk}(\mathbf{Z}) = \mathbf{A} \boxplus (\mathbf{T} \cdot c)$.

□

M.5 Summary of Results of Leakage-of-Randomness

The generic constructions of ID schemes (Theorem M.2) and signatures (Theorem M.5), along with the construction of a Σ-protocol for the relation $R_{\mathcal{C}}$ when instantiated with our efficient CLR-OWR (based on K-linear) proves the existence and relative leakage of the schemes. The AKA protocol then follows black-box from the construction of entropic-signatures Theorem M.6 and their superior relative leakage. This proves Theorem 7.1.

N Proofs Omitted from the Main Body

N.1 Theorem 3.2 (LIRR $\Rightarrow$ CLR-OWR)

Proof of Theorem 3.2. The correctness properties of CLR-OWR follow from the correctness/re-randomization of the LIRR. We use a series-of-games argument to argue ℓ-CLR security:

Game 0: This is the original ℓ-CLR Game from Definition 2.3.

Game 1: In this game, the challenger initially samples $(pk, \mathtt{sam}_G, \mathtt{sam}_B, dk) \leftarrow \mathtt{Setup}(1^\lambda)$ $sk_1 \leftarrow \mathtt{SampG}(\mathtt{sam}_G)$ and gives pk to $\mathcal{A}$. The game them proceeds as before with many leakage rounds, *except that* the secret key used in each leakage-round i is chosen as $sk_i \leftarrow \mathtt{SampG}(\mathtt{sam}_G)$, independently of all previous rounds.

Games 0 and 1 are indistinguishable by the *re-randomization property* (applied q times, where q is the total number of leakage-rounds).

Game 2: Game 2 is the same as Game 1, except that we modify the winning-condition to say that the adversary only wins if, at the end, it outputs sk^* such that $\mathtt{isGood}(pk, sk^*, dk) = 1$.

The winning probability of $\mathcal{A}$ in Games 2 is at least that of Game 1 minus the probability that sk^* satisfies $\mathtt{Ver}(pk, sk^*) = 1 \wedge \mathtt{isGood}(pk, sk^*, dk) = 0$. However, since the entire interaction between the challenger and the adversary in Games 1,2 can be simulated using $(pk, \mathtt{sam}_G)$, we can use the *hardness of good keys* property to argue that the probability of the above happening is negligible. Therefore the probability of $\mathcal{A}$ winning in Games 2 is at least that of Game 1, up to negligible factors.

Games 2.i - 3: Let q be the total number of leakage rounds for which $\mathcal{A}$ runs. We define the Games 2.i for $i = 0, 1 \ldots, q$ as follows. The challenger initially samples $(pk, \mathsf{sam}_G, \mathsf{sam}_B, dk) \leftarrow \mathtt{KeyGen}(1^\lambda)$ and gives pk to $\mathcal{A}$. The game them proceeds as before with many leakage rounds, *except that* the secret keys in rounds $j \leq i$ are chosen by $sk_j \leftarrow \mathtt{SampB}(\mathsf{sam}_B)$, and in the rounds $j > i$, they are chosen as $sk_j \leftarrow \mathtt{SampG}(\mathsf{sam}_G)$. Note that Game 2.0 is the same as Game 2, and we define Game 3 to be the same as Game 2.q.

We use the *ℓ-Leakage Indistinguishability* property to argue that, for $i = 1, \ldots, q$, the winning probability of $\mathcal{A}$ is the same in Game 2.$(i-1)$ as in Game 2.i, up to negligible factors. This is because a reduction $\mathcal{B}$, attacking the LR subset indistinguishability game, can simulate $\mathcal{A}$s view in leakage-rounds $j < i$ using sam_B and rounds $j > i$ using sam_G. For round i, the reduction simulates leakage by calling its own leakage-oracle, on the challenge secret-key. At the end, $\mathcal{B}$ outputs the value sk^* output by $\mathcal{A}$. If $\mathcal{B}$'s challenger uses a good key then that corresponds to the view of $\mathcal{A}$ in game 2.$(i-1)$ and a bad key corresponds to game 2.i. Therefore, letting b be the bit used by $\mathcal{B}$s challenger:

$$\begin{aligned}\left|\Pr[\mathcal{B} \text{ wins }] - \frac{1}{2}\right| &= \left|\Pr[\mathtt{isGood}(pk, sk^*, dk) = b] - \frac{1}{2}\right| \\ &= \frac{1}{2}\left|\ \Pr[\mathtt{isGood}(pk, sk^*, dk) = 1 \mid b = 1] - \Pr[\mathtt{isGood}(pk, sk^*, dk) = 1 \mid b = 0]\right| \\ &= \frac{1}{2}\left|\ \Pr[\mathcal{A} \text{ wins in Game } 2.(i-1)] - \Pr[\mathcal{A} \text{ wins in Game } 2.i]\right|\end{aligned}$$

Can't Win Game 3: We now argue that probability of $\mathcal{A}$ winning game 3 is negligible, by *the hardness of good keys*. Notice that $\mathcal{A}$'s view in Game 3 can be simulated entirely just given pk, sam_B. Therefore, there is a PPT algorithm which, given pk, sam_B as inputs, can run Game 3 with $\mathcal{A}$ and output sk^* such that $\mathtt{isGood}(pk, sk^*, dk) = 1$ whenever $\mathcal{A}$ wins. So the probability of $\mathcal{A}$ winning in Game 3 is negligible.

By the hybrid argument, the probability of $\mathcal{A}$ winning in Game 0 is at most that of $\mathcal{A}$ winning in Game 3, up to negligible factors. Therefore, since the latter is negligible, the former must be negligible as well, which concludes the proof of the theorem. □

N.2 Lemma 4.1 (Hardness Properties of LIRR)

Proof of Lemma 4.1. The correctness properties follow from the correctness of NIZK and encryption. The *hardness of bad keys* property follows by the *soundness* of the argument-system Π. In particular, if $\mathtt{Ver}(pk, sk^*) = 1$ and $\mathtt{isGood}(pk, sk^*, dk) = 0$, then $sk^* = (c_2, \pi)$ where π is a valid proof (i.e. $\mathtt{Ver}((c_1, c_2), \pi) = 1$) for a false statement $(c_1, c_2) \notin L_{eq}$. Therefore, an adversary $\mathcal{A}$ that wins the "hardness of bad keys" game, and produces such sk^* with non-negligible probability, can be used to break the soundness of the NIZK argument system Π. Notice that we can easily simulate pk, sam_G for such an adversary without knowing the trapdoor TK for the CRS of the NIZK.

The *hardness of good keys* property follows by the one-wayness of $\mathcal{E}_1$. In particular, if $sk^* = (c_2^*, \pi^*)$ is such that $\mathtt{isGood}(pk, sk^*, dk) = 1$, then $\mathtt{Dec}^1_{sk_1}(c_1) = \mathtt{Dec}^2_{sk_2}(c_2^*) = m$. Therefore, we can use an adversary that wins the "hardness of good keys" game, and produces such sk^* with non-negligible probability, to break the one-wayness of $\mathcal{E}_1$. We simply simulate (pk, sam_B) for the adversary by choosing the key-pair (pk_2, sk_2) of $\mathcal{E}_2$ and the (CRS, TK) of the NIZK system ourselves, and using the challenge public-key pk_1 and challenge-ciphertext c_1 to generate $pk = (pk_1, pk_2, \text{CRS}, c_1)$, $\mathsf{sam}_B = \text{TK}$. Then, given the adversary's forged key $sk^* = (c_2^*, \pi^*)$, we simply recover $m = \mathtt{Dec}^2_{sk_2}(c_2^*)$ to break the one-wayness of the challenge-ciphertext c_1. □

N.3 Lemma 4.5 (Homomorphic Relation)

Proof of Lemma 4.5. Let $y = (c_1, c_2), y' = (c'_1, c'_2), x = (m, r_1, r_2), x' = (m', r'_1, r'_2)$ such that $(y, x), (y', x') \in R_{eq}^{(pk_1,pk_2)}$. Then $y{\cdot}y' = (c_1{\cdot}c'_1, c_2{\cdot}c'_2)$ and $x{+}x' = (m{\cdot}m', r_1{+}r'_1, r_2{+}r'_2)$ such that $c_1{\cdot}c'_1 = \mathtt{Enc}^1_{pk_1}(m{\cdot}m'; r_1{+}r'_1)$, $c_2 \cdot c'_2 = \mathtt{Enc}^2_{pk_2}(m \cdot m'; r_2 + r'_2)$. Therefore $(y \cdot y', x + x') \in R_{eq}^{(pk_1,pk_2)}$. □

N.4 Lemma 4.6 (Rerandomization)

Proof of Lemma 4.6. Fix any $pk = (pk_1, pk_2, \textsc{crs}, c_1), \mathsf{sam}_G$ output by $\mathtt{KeyGen}$ and $sk_G = (c_2, \pi)$ output by $\mathtt{SampG}(\mathsf{sam}_G)$. Then there exist some m, r_1, r_2, r_3 for which

$$\mathsf{sam}_G = (m, r_1) \quad , \quad c_1 = \mathtt{Enc}^1_{pk_1}(m; r_1) \quad , \quad c_2 = \mathtt{Enc}^2_{pk_2}(m; r_2) \quad , \quad \pi = \mathtt{Prov}((c_1, c_2), (m, r_1, r_2); r_3).$$

If we sample $sk^* \leftarrow \mathtt{ReRand}(sk_G)$, we have $sk^* = (c^*, \pi^*)$ and:

$$\begin{aligned} c^* &= c_2 \cdot \mathtt{Enc}_{pk_2}(1_{\mathcal{M}}; r'_2) = \mathtt{Enc}_{pk_2}(m; r_2 + r'_2) \\ \pi^* &= \pi \cdot \mathtt{Prov}((1_{\mathcal{C}_1}, c^*/c_2), (1_{\mathcal{M}}, 0_{\mathcal{R}_1}, r'_2); r'_3) = \mathtt{Prov}((c_1, c^*), (m, r_1, r_2 + r'_2); r_3 + r'_3) \end{aligned}$$

for some fresh random values r'_2, r'_3. Therefore, sampling $sk^* \leftarrow \mathtt{ReRand}(sk_G)$ is *exactly the same* as sampling $sk^* \leftarrow \mathtt{SampG}(\mathsf{sam}_G)$, even after fixing any choice of pk, sam_G, sk_G. This shows that the re-randomization property holds with *perfect* distributional equality.

□

N.5 Lemma 4.8 (LoC-NM ⇒ LIRR)

Proof of Lemma 4.8. Assume $\mathcal{A}$ is a PPT adversary which has a non-negligible advantage in the leakage-indistinguishability (LI) game of Definition 3.1.

Consider the following modification to the LI game. In the original LI game, if the challenger's bit is $b = 1$, then the challenger samples the challenge secret-key $\tilde{sk} = (\tilde{c}, \tilde{\pi}) \leftarrow \mathtt{SampG}(\mathsf{sam}_G)$. Therefore, $\tilde{c}$ is a random encryption of m, and π is an honestly generated proof for the statement $(c_1, c_2) \in L_{eq}^{(pk_1,pk_2)}$ under the witness (m, r_1, r_2). In the modified-LI game, if the challenger's bit is $b = 1$, then the challenger samples $\tilde{c}$ as before, but uses a simulated proof $\tilde{\pi} \leftarrow \mathtt{Sim}^{\Pi}((c_1, c_2), \textsc{tk})$.

The adversary's probability of winning in the modified LI game is the same as that of the original game, up to negligible factors, by the the (composable) NIZK property of the proof system Π (note that we need composable NIZK, since the adversary see the trapdoor $\textsc{tk}$ for the $\textsc{crs}$ of Π).

We now show how to use any adversary $\mathcal{A}$ that attacks the modified LI game to construct a reduction $\mathcal{B}$ that attacks the LoC-NM game.

1. The reduction $\mathcal{B}$ initially gets a challenge public-key, which it sets as pk_2. In addition $\mathcal{B}$ samples and the values (pk_1, sk_1), $(\textsc{crs}, \textsc{tk})$, $\mathsf{sam}_G = (m, r)$, $\mathsf{sam}_B = \textsc{tk}$ and $c_1 = \mathtt{Enc}_{pk_1}(m; r)$ just as the honest key-generation algorithm of the LIRR construction. It gives $pk = (\textsc{crs}, pk_1, pk_2, c_1), \mathsf{sam}_G, \mathsf{sam}_B$ to $\mathcal{A}$.

2. The reduction $\mathcal{B}$ chooses challenge messages $m_0 = 1_{\mathcal{M}}$, $m_1 = m$ and gives these to its challenger.

3. The adv. $\mathcal{A}$ expects to make up to ℓ calls to a leakage-oracle (simulated by $\mathcal{B}$) on a secret key $\tilde{sk} = (\tilde{c}, \tilde{\pi})$. The reduction $\mathcal{B}$ initially chooses some randomness r for a NIZK simulator. Then, for each function h chosen by $\mathcal{A}$ (recall h expects a value $\tilde{sk} = (\tilde{c}, \tilde{\pi})$ as input), the adv. $\mathcal{B}$ creates a function h' (which expects only $\tilde{c}$ as input, and contains $\textsc{tk}, r$ hard-coded) such that $h'(\tilde{c})$ computes $\tilde{\pi} = \mathtt{Sim}((c_1, \tilde{c}), tk; r)$ and then outputs $h(\tilde{c}, \tilde{\pi})$. The reduction $\mathcal{B}$ then passes h' to its own leakage-oracle on the challenge ciphertext $\tilde{c}$. [13]

[13] Essentially, we are saying that $\mathcal{B}$ can correctly simulate ℓ bits of leakage on $\tilde{sk} = (\tilde{c}, \tilde{\pi})$ given ℓ bits of leakage on $\tilde{c}$ alone, by having the leakage-functions simulate $\tilde{\pi}$.

4. When $\mathcal{A}$ outputs $sk^* = (c^*, \pi^*)$, the reduction $\mathcal{B}$ gives c^* to its challenger and, if it gets back the message m, it outputs $\tilde{b} = 1$. Else it outputs $\tilde{b} = 0$.

We now argue that the reduction $\mathcal{B}$ wins the ℓ-LoC game with the exact same probability that $\mathcal{A}$ wins in the modified-LI game, which concludes the proof. This is because, no matter what bit b is chosen by the challenger in the ℓ-LoC-NM game, the simulated view of $\mathcal{A}$ above is *exactly* that of the modified LI game with challenge-bit b. Moreover, $\mathcal{B}$ outputs $\tilde{b} = b$ whenever $\mathcal{A}$ outputs sk^* such that $\texttt{isGood}(pk, sk^*, dk) = b$. □

N.6 Theorem 7.2 (Leakage of Refreshing)

N.6.1 Proof Intuition

The main idea behind allowing logarithmic leakage-of-refreshing is that we can simulate the $\mu(\lambda)$-bits of leakage on the internal state (sk_i, r_i) of the refresh operation $sk_{i+1} = \texttt{ReRand}(sk_i; r_i)$, by leaking an additional $\mu(\lambda)$-bits on the secret-key sk_{i+1} alone, without knowing r_i. In particular, our leakage function will get sk_{i+1} and try *all possible choices* of the $\mu(\lambda)$-bit leakage that the adversary $\mathcal{A}$ could have seen during the refresh. For each of the $2^{\mu(\lambda)}$ possible choices, our leakage-function it will *gage $\mathcal{A}$'s success probability* given this choice, by running $\mathcal{A}$ on many random continuations of the leakage game using the key sk_{i+1} to simulate the challenger going forward. At the end, the leakage-function will output the optimal $\mu(\lambda)$-bit value that maximizes $\mathcal{A}$'s probability. So leakage on the refreshing (sk_i, r_i) is simulated using sk_{i+1} alone. Note that the *actual* behavior of the challenger in future rounds depends *only* on sk_{i+1} (and randomness), so the estimate that the leakage-function gets on success probability of $\mathcal{A}$ is (likely) accurate. Therefore, given this "simulated" leakage, $\mathcal{A}$'s success probability should not be much smaller then when given the "correct" leakage calculated using (sk_i, r_i). The only bottleneck of the approach is that the simulated-leakage-function runs in $2^{\mu(\lambda)}$ time. Therefore, using this approach, we are stuck with $\mu(\lambda) = O(\log(\lambda))$ or with making stronger hardness assumptions.

N.6.2 Formal Proof

Proof of Theorem 7.2. Let $(\texttt{KeyGen}, \texttt{Ver}, \texttt{ReRand})$ satisfy the syntax of a CLR-OWR. Assume that there is an adversary $\mathcal{A}$ that breaks the $(\ell(\lambda), \mu(\lambda))$-CLR security of the scheme, for some $\mu(\lambda) = O(\log(\lambda))$. Then there are some polynomials $q(\cdot), p(\cdot)$ such that $\mathcal{A}$ runs in $q(\lambda)$ leakage rounds and that its success probability is at least $1/p(\lambda)$ for infinitely many $\lambda \in \mathbb{N}$. Let $\epsilon(\cdot)$ be any inverse-polynomial. We define an adversary $\mathcal{B}$ that wins the $\ell(\lambda)$-CLR security game (without leakage-of-refreshing) against the scheme with probability $1/p(\lambda) - \epsilon(\lambda) - \mathsf{negl}(\lambda)$.

Description of $\mathcal{B}$. It is easiest to think of $\mathcal{A}$ as submitting a single deterministic leakage-query f_i on the secret key sk_i (in-between refreshing), and a single deterministic leakage-query h_i on the state sk_i, r_i (during the refreshing) in each round. On the other hand, $\mathcal{B}$ will submit two randomized queries on each secret sk_i in each round.[14] The adversary $\mathcal{B}$ works as follows. It runs an internal copy of $\mathcal{A}$ and forwards the public-key pk from its challenger to $\mathcal{A}$. In between refreshing, $\mathcal{B}$ forwards the leakage query $f_i : \{0,1\}^* \to \{0,1\}^{\ell_i}$ from $\mathcal{A}$ to its challenger and the response $f_i(sk_i)$ back to $\mathcal{A}$. During the refreshing, when $\mathcal{A}$ submits a leakage query $h_i : \{0,1\}^* \to \{0,1\}^{\mu_i}$ on the refreshing state (sk_i, r_i), the reduction $\mathcal{B}$ moves on to the next leakage round and *simulates* the refreshing leakage by leaking μ_i bits on the secret key sk_{i+1}. In particular, $\mathcal{B}$ constructs a query $h'_i : \{0,1\}^* \to \{0,1\}^{\mu_i}$ on the key sk_{i+1}, as described below, and feeds the output $h'_i(sk_{i+1})$ to $\mathcal{A}$. At the end, $\mathcal{B}$ outputs whatever secret-key sk^* is output by $\mathcal{A}$.

[14]Recall that this is w.l.o.g. as any adversary that submits multiple adaptive randomized queries can always be converted into an adversary making a single deterministic query in each round. In particular, the adversary can choose the randomness for the query itself beforehand, and can combine several adaptive queries into one.

Description of the function $h'_i : \{0,1\}^* \to \{0,1\}^{\mu_i}$.

The function h'_i contains a hard-coded copy of $\mathcal{A}$, in its current state after issuing the query h_i.

Main Computation: Try all 2^{μ_i} possible values of $\psi \in \{0,1\}^{\mu_i}$. For each ψ, find an *estimate* $\tilde{\rho}_\psi \in [0,1]$ for the success probability of $\mathcal{A}$ with the leakage ψ, as described below. Output the choice of ψ that maximizes $\tilde{\rho}_\psi$.

Estimating $\tilde{\rho}_\psi$ using sk_{i+1}: Perform the following random experiment $k = 8q^2(\lambda)(\lambda + \mu_i)/\epsilon^2(\lambda)$ times:

Choose fresh random coins $r_{i+1}, \ldots, r_{q(\lambda)-1}$ and compute the keys

$$sk_{i+2} = \texttt{ReRand}(sk_{i+1}; r_{i+1}), \ldots, sk_{q(\lambda)} = \texttt{ReRand}(sk_{q(\lambda)-1}, r_{q(\lambda)-1}).$$

Then run the rest of the leakage game with $\mathcal{A}$ by feeding it ψ in response to h_i, and use the values $sk_{i+1}, r_{i+1}, sk_{i+2}, \ldots, r_{q(\lambda)-1}, sk_{q(\lambda)}$ to correctly respond to all of $\mathcal{A}$'s future queries f_j, h_j for $j > i$.

Set $\tilde{\rho}_\psi$ to be the fraction of the k experiments in which $\mathcal{A}$ "wins" at the end.

Analysis of $\mathcal{B}$. It is easy to see that $\mathcal{B}$ runs a legal strategy since the number of bits leaked in each round i is $\mu_{i-1} + \ell_i \le \ell(\lambda)$. Furthermore, the run-time of each leakage functions is at most $2^{\mu(\lambda)}\mathsf{poly}(\lambda) = \mathsf{poly}(\lambda)$. So we are left to analyze the success probability of $\mathcal{B}$. To do so, it is useful to consider the following hybrid experiments.

Define **Experiment** 0 to be the $(\ell(\lambda), \mu(\lambda))$-CLR OWR game between the challenger and the adversary $\mathcal{A}$. Define experiments $1, \ldots, q(\lambda)$ analogously, but in **Experiment** i, the first i leakage-of-refreshing queries $h_0, \ldots, h_{i-1}$ asked by $\mathcal{A}$ are answered with $h'_j(sk_{j+1})$ instead of $h_j(sk_j, r_j)$. Let W_i be the event that $\mathcal{A}$ wins in **Experiment** i. Then $\mathcal{B}$'s success probability is exactly $\Pr[W_{q(\lambda)}]$. Let $\mathsf{Pre}_{i,j}$ denote the random variable for the "preamble" of experiment i up to the "point" j, consisting of:

- The view of $\mathcal{A}$ right after making the leakage-of-refreshing query h_j, but before seing the response. This includes the random coins of $\mathcal{A}$, the public-key pk seen by $\mathcal{A}$, and all of the responses to all of the prior leakage queries.
- The value of the secret key sk_j used in round j and the randomness r_j used to compute $sk_{j+1} = \texttt{ReRand}(sk_j; r_j)$.

Conditioned on the preambles $\mathsf{Pre}_{i,i}$ and $\mathsf{Pre}_{i+1,i}$ of experiments i and $i+1$ taking on some concrete value σ, the only difference between experiments i and $i+1$ is how the leakage-of-refreshing query h_i is answered. In the former case, it is answered via $h_i(sk_i, r_i)$, and in the latter case it is answered via $h'_i(sk_{i+1})$, where sk_i, r_i, sk_{i+1} are fixed by σ. The following claim intuitively says that, since h'_i chooses its response by trying all options and testing, its answer should not be much worse than that of h_i.

Claim N.1. *For any σ in the support of $\mathsf{Pre}_{i+1,i}$ and any $i \in \{0, \ldots, q(\lambda)-1\}$, we have*

$$\Pr[W_{i+1} \mid \mathsf{Pre}_{i+1,i} = \sigma] \ge \Pr[W_i \mid \mathsf{Pre}_{i,i} = \sigma] - \epsilon(\lambda)/q(\lambda) - \mathsf{negl}(\lambda).$$

Proof of Claim N.1. In the proof we fix σ and always condition all probabilities on the preamble of the experiment being σ; we will use the variables $W_{i,\sigma}$ and $W_{i+1,\sigma}$ to make this conditioning explicit. Recall that σ fixes sk_i, r_i, sk_{i+1}. Let $\psi^* = h_i(sk_i, r_i)$ and let $\rho^* = \Pr[W_{i,\sigma}]$. Define the random variable A to be the value of $h'_i(sk_{i+1})$ in experiment $i+1$, over the internal randomness of h'_i. Define $\rho_\psi = \Pr[W_{i+1,\sigma} \mid A = \psi]$. Then, it is easy to see that $\rho_{\psi^*} = \rho^*$ since, conditioned on the response to h_i being ψ, the continuation of the two experiments $i+1$ and i are exactly the same. Let us define the set $\mathit{Good} = \{\psi \;:\; \rho_\psi \ge \rho^* - \epsilon(\lambda)/q(\lambda)\}$. Then:

$$\Pr[W_{i+1,\sigma}] \ge \sum_{\psi \in \mathit{Good}} \Pr[W_{i+1,\sigma} \mid A = \psi] \Pr[A = \psi] \ge (\rho^* - \epsilon(\lambda)/q(\lambda)) \Pr[A \in \mathit{Good}] \qquad (13)$$

We now show that $\Pr[A \in \mathit{Good}]$ is high. To do so, recall that the function $h'_i(sk_{i+1})$ chooses the response ψ based on the estimated success probabilities $\tilde{\rho}_\psi$. Since $\tilde{\rho}_\psi$ is computed by averaging k experiments each of which succeeds with probability ρ_ψ, we can use the Chrnoff bound to get:

$$\Pr[|\tilde{\rho}_\psi - \rho_\psi| \geq \epsilon(\lambda)/2q(\lambda)] \leq 2e^{-k\epsilon^2(\lambda)/8q^2(\lambda)} \leq 2e^{-\lambda}2^{-\mu_i} \tag{14}$$

Using the Union Bound over all $\psi \in \{0,1\}^{\mu_i}$, we see that *every estimate* $\tilde{\rho}_\psi$ computed internally by h'_i satisfies $|\tilde{\rho}_\psi - \rho_\psi| < \epsilon(\lambda)/2q(\lambda)$ with probability at least $(1 - 2e^{-\lambda})$ over the internal coins of h'_i . If that occurs then:

- For the "correct" $\psi^* = h_i(sk_i, r_i)$, we have $\tilde{\rho}_{\psi^*} > \rho_{\psi^*} - \epsilon(\lambda)/2q(\lambda) = \rho^* - \epsilon(\lambda)/2q(\lambda)$.
- For the "actual" ψ' output by h'_i, we have $\tilde{\rho}_{\psi'} \geq \tilde{\rho}_{\psi^*}$ and $\rho_{\psi'} > \tilde{\rho}_{\psi'} - < \epsilon(\lambda)/2q(\lambda)$.
- Putting these together, we get: $\tilde{\rho}_{\psi'} > \rho^* - \epsilon(\lambda)/q(\lambda)$ and hence the output is $\psi' \in \mathit{Good}$.

So $\Pr[A \in \mathit{Good}] \geq (1 - 2e^{-\lambda}) = (1 - \mathsf{negl}(\lambda))$ and hence, plugging this into (13) and expanding out, we get

$$\Pr[W_{i+1,\sigma}] \geq \Pr[W_{i,\sigma}] - \epsilon(\lambda)/q(\lambda) - \mathsf{negl}(\lambda)$$

□

Using the above claim, we get that for each $i \in \{0, \dots, q(\lambda)\}$:

$$\begin{aligned}
\Pr[W_{i+1}] &= \sum_{\sigma \in \{0,1\}^*} \Pr[W_{i+1} \mid \mathsf{Pre}_{i+1,i} = \sigma] \Pr[\mathsf{Pre}_{i+1,i} = \sigma] \\
&\geq \sum_{\sigma \in \{0,1\}^*} (\Pr[W_i \mid \mathsf{Pre}_{i,i} = \sigma] - \epsilon(\lambda)/q(\lambda) - \mathsf{negl}(\lambda)) \Pr[\mathsf{Pre}_{i,i} = \sigma] \\
&\geq \Pr[W_i] - \epsilon(\lambda)/q(\lambda) - \mathsf{negl}(\lambda)
\end{aligned}$$

where the second line follows from the claim and the fact that $\mathsf{Pre}_{i+1,i}$ and $\mathsf{Pre}_{i,i}$ have the exact same distribution (the preambles are the same in both experiments). Therefore

$$\Pr[\mathcal{B} \text{ wins }] = \Pr[W_{q(\lambda)}] \geq \Pr[W_0] - \epsilon(\lambda) - \mathsf{negl}(\lambda) = \Pr[\mathcal{A} \text{ wins }] - \epsilon(\lambda) - \mathsf{negl}(\lambda)$$

which proves the theorem. □

It is easy to extend the above proof to encryption, signatures and other schemes with CLR security. Also, it is easy to see that we can allow for a super-logarithmic leakage-of-refreshing bound $\mu(n)$ if the leakage-functions of the original scheme can run in time $2^{\mu(n)}\mathsf{poly}(n)$.