New Approach to Practical Leakage-Resilient Public-Key Cryptography

Suvradip Chakraborty[1], Janaka Alawatugoda[2], and C. Pandu Rangan[1]

[1] Computer Science and Engineering Department, Science and Engineering Faculty,
Indian Institute of Technology Madras
Chennai, Tamil Nadu 600036, India
{suvradip,rangan}@cse.iitm.ac.in
[2] Department of Computer Engineering, Faculty of Engineering,
University of Peradeniya
Peradeniya 20400, Sri Lanka
janaka@ce.pdn.ac.lk

Abstract. We present a new approach to construct several leakage-resilient cryptographic primitives, including leakage-resilient public-key encryption (PKE) schemes, authenticated key exchange (AKE) protocols and low-latency key exchange (LLKE) protocols. To this end, we introduce a new primitive called *leakage-resilient non-interactive key exchange* (LR-NIKE) protocol. We introduce a generic security model for LR-NIKE protocols, which can be instantiated in both the bounded and continuous-memory leakage ((B/C)-ML) settings. We then show a secure construction of LR-NIKE protocol in the bounded-memory leakage (BML) setting, that achieves an optimal leakage rate, i.e., $1 - o(1)$. Finally, we show how to construct the aforementioned leakage-resilient primitives from such a LR-NIKE protocol as summarized below. All the primitives also achieve the *same* (optimal) leakage rate as the underlying LR-NIKE protocol.

- We show how to construct a leakage-resilient IND-CCA-2-secure PKE scheme in the BML model generically from a LR-NIKE protocol. Our construction *differs* from the state-of-the-art constructions of leakage-resilient IND-CCA-2-secure PKE schemes, which use hash proof techniques to achieve leakage-resilience. Moreover, our transformation preserves the leakage-rate of the underlying LR-NIKE and admits more efficient construction than previous such PKE constructions.
- We introduce a new leakage model for AKE protocols, in the BML setting. We show how to construct a leakage-resilient AKE protocol starting from LR-NIKE protocol.
- We introduce the first-ever leakage model for LLKE protocols in the BML setting, and the first construction of such a leakage-resilient LLKE from LR-NIKE protocol.

Keywords: leakage-resilient cryptography, bounded-leakage, non-interactive key exchange, key exchange protocols, low-latency key exchange.

1 Introduction and Related Works

Traditional cryptographic primitives are provably analyzed in a black-box model, where the adversary has access to the primitive via restrictive and well-defined interfaces (oracles). However, this does not truly reflect the real-world scenario, where the adversary may obtain lots of unintended *side-channel* information about the cryptosystem from its implementation. This extra leakage of information is not accounted for in its analysis in the aforementioned black-box model of security. Leakage-resilient cryptography emerged as a theoretical foundation to address the issue of side-channel attacks. Here it is assumed that the adversary has access to side-channel information, which is modeled by allowing the adversary to specify arbitrary leakage functions (subject to some restrictions) and obtain leakage from the secret key of the system, as dictated by these functions. Note that, some restrictions must be imposed on the class of allowable leakage functions, as otherwise an adversary can simply read off the entire secret key from memory. Depending upon these restrictions, many theoretical models of leakage have emerged in the recent literature [2, 6, 8, 11, 13, 28]. In this work, we focus mainly on the *bounded-memory leakage* model [2, 6]. In this model, the adversary chooses arbitrary polynomial-time computable leakage functions f and receives returns $f(sk)$, where sk is the secret key. The only restriction is that the sum of output length of all these leakage functions that the adversary can ever obtain is bounded by some parameter λ, which is smaller than the size of sk. Other notable models of leakage include the Only Computation Leaks Information (OCLI) model, continual memory leakage model, auxiliary input leakage model etc. We briefly survey these models in Appendix B.

Ever since the ground-breaking work of Diffie and Hellman (DH) [10], authenticated key exchange (AKE) protocols arose as an important cryptographic primitive. The DH key exchange protocol can actually be viewed as a non-interactive key exchange (NIKE) protocol, where the parties can establish a shared key among themselves without any interaction, provided the public-keys of all the parties are pre-distributed and they agree on some (common) global public parameters. NIKE is very useful in any band-width-critical, power-critical, resource-critical systems such as embedded devices, wireless and sensor networks, where the communication must be at its minimum. Despite its real-world applications, NIKE has mostly been overlooked until recently [16]. Freire et al. [16] proposed formal security models for NIKE and efficient constructions of NIKE in these models. Although the NIKE constructions of [16] is secure in the traditional (non-leakage) setting, the security of them may completely break down in the presence of leakage. In fact, we demonstrate that the pairing-based construction of NIKE shown in [16] is insecure, even if the adversary could obtain only a single bit of leakage from the secret key of a party. Therefore, it is really important to thoroughly study on the leakage resiliency of NIKE. We note that much research has been carried out on analyzing leakage resiliency of interactive key exchange protocols [3–5, 9], but the leakage resiliency of NIKE remains largely unstudied.

As one of the central applications of leakage-resilient NIKE (LR-NIKE), we show how to construct leakage-resilient IND-CCA-2-secure PKE scheme generically from LR-NIKE (in the bounded-memory leakage setting). All the previous constructions of practical leakage-resilient IND-CCA-2 (LR-IND-CCA-2) secure PKE schemes relies solely on hash proof techniques to achieve leakage resiliency. However, the generic approach of constructing leakage-resilient CCA secure PKE scheme *solely* using hash proof system (HPS) systems is inherently limited to leakage rate below 1/2, as pointed out by Dodis et al. [12]. The leakage rate of the state-of the art constructions of LR-IND-CCA-2-secure PKE scheme was later improved in the subsequent works of Qin et al. [30, 31], which achieved leakage rates of $1/2 - o(1)$ and $1 - o(1)$ respectively. They could achieve the leakage rate of $1/2 - o(1)$ by using HPS and one-time lossy filters (OTLF)[3], and the optimal rate of $1 - o(1)$ by cleverly instantiating the underlying primitives, namely HPS and OTLF. However, the complexity assumption they make for their construction is rather non-standard, namely refined subgroup indistinguishability (RSI) assumption over composite order groups. The parameters of their construction are also large due to use of composite-order groups.

We deviate significantly from this HPS-based approach of constructing LR-IND-CCA-2-secure PKE schemes, and show that this connection is not inherent. To this end, we develop a new primitive called *leakage-resilient non-interactive key exchange* (NIKE). Our construction of leakage-resilient NIKE relies solely on leakage-resilient chameleon hash function (which in turn relies only on strong collision-resistant

[3] Note that this circumvents the impossibility result of Dodis et al. [12], since the analysis of [12] considered the fact that the LR-IND-CCA-secure PKE was constructed solely from HPS; whereas in [30, 31] they do not solely use HPS and instead relies on both HPS and OTLF for their construction.

hash function) and only a constant number (to be precise only 3) of pairing operations. We then show a very simple and generic construction of LR-IND-CCA-2-secure PKE schemes achieving the optimal leakage rate of $1 - o(1)$, based solely on the assumption that the leakage-resilient NIKE exists. Our construction significantly *improves* the efficiency of LR-IND-CCA-2 secure PKE schemes compared to their state-of-the-art constructions (which use the HPS-based approach) (please refer to Table 1 for details), at the same time achieving the optimal leakage rate of $1 - o(1)$.

We also show the applicability of leakage-resilient NIKE to construct leakage-resilient authenticated key exchange (AKE) protocols and leakage-resilient low-latency key exchange (LLKE) protocols (in the bounded-memory leakage setting). All the previous constructions of leakage-resilient AKE protocols [4, 5, 9] either implicitly rely on HPS (by using leakage-resilient PKE as their building block) or explicitly by using the properties of HPS. Our generic construction of leakage-resilient AKE gives an alternate way to construct AKE protocols, different from the previous constructions of leakage-resilient AKE protocols, achieving the optimal leakage rate of $1 - o(1)$. Low-latency key exchange (LLKE) are one of the most practical key exchange protocols that permits the transmission of cryptographically protected data, without prior key exchange, while providing perfect forward secrecy (PFS). This concept was discussed in the Google's QUIC [4] protocol. Further, a low-latency mode is currently under discussion for inclusion in TLS 1.3 version. Although, the first formal model of LLKE was studied by Hale et al. [21], leakage resiliency of LLKE remains unstudied until present. Being a candidate for TLS 1.3, it is important to explore the leakage resiliency of LLKE protocols, as side-channel attacks widely exist.

Technical Contributions. The main contributions are abridged as follows:

Leakage-resilient NIKE. As our *first* major contribution, we study the leakage resiliency of NIKE protocols. We present a leakage security model for NIKE protocols, defining the notion of leakage-resilient non-interactive key exchange. Our model is generic and can be instantiated in both the bounded-memory leakage model as well as in continuous-memory leakage model. We then show how to construct secure a NIKE protocol in the bounded-memory leakage model.

Our Model: Our model of leakage-resilient NIKE generalizes the CKS-heavy model of NIKE proposed by Freire et al. [16], in the setting of leakage. Our model is a very strong model allowing the adversary to register arbitrary public keys into the system, corrupt honest parties to obtain their secret keys, issue extract queries to obtain shared keys between two honest parties and also between one honest party and another corrupt party. Besides this, we also allow the adversary to obtain additional leakage from both the parties involved in the Test/challenge query. We also introduce the notion of *validity* of a test query reminiscent of the notion of freshness of a test session for (interactive) key exchange protocols. Finally, in the (valid) test query between two honest parties, the adversary has to distinguish the shared key from a random key. In the bounded leakage instantiation of our model, the adversary can obtain bounded leakages from the secret keys of all parties, including those involved in the test query also. Differently, in the continuous leakage instantiation of our model the secret key is refreshed periodically. The adversary can obtain unbounded and arbitrary leakage from the secret keys of the parties, provided the leakage from the secret key per invocation (between two successive refreshes) is bounded.

Our Construction: The starting point of our construction is the NIKE scheme of Freire et al. [16]. However, we show that the above scheme is insecure in the setting of leakage, even if a *single* bit of the secret key is allowed to leak. The main step where the construction breaks down is related to the exponentiation operation. In other words, if exponentiation is performed normally as in the original protocol, it may be completely insecure in the presence of leakage. A common countermeasure against this uses *masking* technique, where the secret key of the system is secret-shared using a multiplicative secret sharing scheme and the exponentiation is done step-wise using each of these shares. However, these masking schemes do not achieve the optimal $1 - o(1)$ leakage-rate, and also require additional restrictions (assumptions) on the class of allowable leakage functions for arguing security. In particular, all these masking schemes are proven secure in the *Only Computation leaks Information* (OCLI) axiom of Micali and Reyzin [28] or under the *split-state* assumption [19, 22, 25].

[4] https://www.chromium.org/quic

The OCLI axiom postulates that the leakage only happens from the memory parts that are touched during the actual computation(s), and the rest of the memory portions not touched by computation are not prone to leakage. In split-state leakage model, it is assumed that the secret key is split into several disjoint parts and the adversary is allowed to obtain leakage independently from each of these parts. However, both these models do not address leakage from the entire memory, which is the case for bounded memory leakage model.

So, a major challenge in our construction is to come up with a leakage-resilient exponentiation operation achieving leakage rate of $1-o(1)$ in the global memory leakage model. Our first idea was to use the techniques of [1,6,29] to perform leakage-resilient exponentiation. In particular let $\mathbb{G}$ be a group of prime order p and let $g_1, \cdots, .g_n$ be random elements of the group. Then the vector $\mathbf{x} = (x_1, \cdots, x_n) \in (\mathbb{Z}_p^*)^n$ is said to be a discrete log representation of some element $y \in \mathbb{G}$ with respect to $g_1, \cdots, .g_n$, if $y = \prod_{i=1}^{n} g_i^{x_i}$. To incorporate this in our construction of NIKE, the elements $g_1, \cdots, .g_n$ can be included in the public parameters *params*, the public key of a party can be set to y and the secret key can be the discrete log representation $\mathbf{x}$ of y. In the works of [1,6,29], it is also shown that, as long as the leakage on each representation is bounded by $(1-2/n)\cdot|\mathbf{x}|$, the adversary cannot come up with another discrete log representation $\mathbf{x}'$ of y. So this achieves the leakage rate of $1-o(1)$. However, it turns out that it becomes surprisingly difficult to incorporate this change in the construction of Freire et al. [16]. The main difficulty stems from the use of multiple generators and also the special structure of the public key.

To this end, we use the ideas of twisted-pair PRF trick [17], but carefully adopted to deal with leakage. The main idea behind the twisted-pair PRF trick is that, it involves two PRFs F and F' with reversing keys. The output of the twisting function is simply the output of the two PRFs that are combined together in special way. The guarantee is that output of the twisting function looks computationally indistinguishable from a uniform value over the same range. For our construction of leakage-resilient twisted-pair PRF trick, we add strong randomness extractors as pre-processors to this original twisting technique [17]. The guarantee is that the output of our leakage-resilient twisted-pair PRF function is computationally indistinguishable from a uniform value over the same range, even if the adversary knows the key of one PRF in full and obtains bounded leakage from the key of the other PRF. For our construction of NIKE in the bounded leakage model, the strong randomness extractor (when appropriately parameterized) takes care of the (bounded) leakage, and then we can extract randomness from the secret key of the NIKE and use the extracted key as the key in one of the PRF. The output of the leakage-resilient twisting function is then used to do secure exponentiation in the presence of leakage. By appropriately parameterizing the extractor we can obtain a leakage rate of $1-o(1)$. Combined with a bounded leakage-resilient CHF tolerating leakage rate of $1-o(1)$, we can achieve a leakage-resilient NIKE in the bounded-memory leakage model with overall leakage rate of $1-o(1)$.

However, a drawback of our leakage-resilient NIKE construction is that it requires a *leak-free* hardware component to store the seed of the extractor. This is required because the view of the adversary in our construction should be independent of the seed, since otherwise the adversary may leak from the extracted value and the uniformity guarantee of the extractor does not hold in this case. Hence, we cannot include the seed in the public key. On the other hand, to compute the shared key, each party needs to have access to the random seed. Hence, we require that the seed be stored in a leak-free hardware component. Note that, the use of leak-free hardware component assumption was also used in many prior works in leakage-resilient cryptography [15,20,23]. In particular, in [23], it is assumed that the leak-free component can produce random encryptions of fixed messages. In [20], it is assumed that there are linear number of such leak-free components and each component is capable of sampling from a fixed polynomial time computable distribution. In [15], it is assumed that the leak-free component can sample two vectors from the underlying field, such that their inner product is zero. In contrast, in our construction, we only require our leak-free component to store the seed of the extractor, which is typically a *short random* bit-string, just short enough to guarantee that the output of the extractor is statistically close to uniform. Hence the use of the leak-free hardware is done in a *minimal* way in our protocol.

Leakage-resilient CCA-2-secure PKE. As one of the central applications of leakage-resilient NIKE (LR-NIKE), we show how to construct leakage-resilient IND-CCA-2 (LR-IND-CCA-2) secure PKE generically from LR-NIKE in the bounded-memory leakage model. This yields a new approach to construct LR-IND-CCA-2-secure PKE schemes, departing completely from the hash-proof frameworks used in prior works. Our construction is practical and also achieves the optimal leakage rate of $1-o(1)$.

Our Construction: Our generic transformation from LR-NIKE to LR-IND-CCA-2-secure PKE scheme essentially follows and adapts the ideas of Freire et al. [16] in the setting of leakage. The main idea behind the transformation is as follows: the public-secret key pair of the LR-IND-CCA-2-secure PKE scheme is same as the public-secret key pair of the underlying LR-NIKE protocol. While encrypting, another public-secret key pair of the NIKE is sampled independently and the shared key generation algorithm of the NIKE is run among the two key pairs yielding a shared key. This key is used as the encapsulation key of the underlying IND-CCA-2-secure key encapsulation mechanism (KEM) and the ciphertext is set to be the new sampled public key. Decryption is straightforward and the decryptor can recover the same shared key by running the shared key generation algorithm with the original secret key and the new sampled public key. Now from the IND-CCA-2-KEM one can easily get a full fledged IND-CCA-2-secure PKE using standard hybrid encryption techniques. Our transformation preserves the leakage rate, in the sense that if the starting point of our construction is a LR-NIKE with leakage rate of $1 - o(1)$, then the LR-CCA-2-secure PKE constructed from it also enjoys the *same* leakage rate.

In Table 1, we show the comparison of our scheme with the state-of-the-art constructions of LR-IND-CCA secure PKE schemes in terms of both *computational* and *communication* complexity. We obtain these complexity figures by instantiating all of the compared schemes with the state-of-the-art constructions of the required underlying primitives. As we can see, the number of group elements involved in our ciphertext is *much lesser* than the number of group elements involved in the ciphertexts of the other schemes. With regard to the number of exponentiations and multiplication operations also, our scheme is much more efficient compared to others, hence improving the computational complexity of the state-of-the-art LR-IND-CCA secure PKE schemes by a significant margin. However, note that we do require a constant number of pairing operations (to be precise only 3) in the encryption side and also in the decryption side. In the table, $n \in \mathbb{N}$, and is usually the number of generators required for the construction. Also [31] works over composite order groups of the form $\mathbb{G} = \mathbb{G}_{\tau_1} \times \mathbb{G}_{\tau_2}$. Here $\mathcal{T}_c$ denotes the tag space in the encryption scheme of [31], and $\{0,1\}^s$ denote the seed space of a strong randomness extractor.

Scheme	#Exponentiations			#Multiplications			#Pairings.			Size of ciphertext C	Leakage rate.	Complexity Assumptions
	KeyGen	Enc.	Dec.	KeyGen	Enc.	Dec.	KeyGen	Enc.	Dec.			
Qin and Liu [30]	n^2	(n^2+n+2)	(n^2+2n)	(n^2+n)	$2n^2$	$(2n^2+n)$	NA	NA	NA	$\mathbb{G}^2 \times \mathbb{Z}_q^n \times \{0,1\}^m \times \widetilde{\mathbb{G}}^{n^2 \times n^2}$	$\frac{1}{2} - o(1)$	DBDH-2
Qin and Liu [31]	(n^2+n+1)	$(3n+2)$	$(3n+1)$	n	$2n$	$2n$	NA	NA	NA	$\mathbb{G}_{\tau_1} \times \{0,1\}^m \times \mathbb{G}^{n+1} \times \mathcal{T}_c$	$1 - o(1)$	RSI over composite order groups
Ours	$(n+5)$	$(2n+8)$	$(n+3)$	$(n+3)$	$(2n+6)$	$(n+3)$	NA	3	3	$\mathbb{G}^2 \times \mathbb{Z}_q^{n+1} \times \{0,1\}^s$	$1 - o(1)$	DBDH-2

Table 1. Comparison among the LR-IND-CCA PKE schemes

Leakage-resilient AKE. We show how to obtain a generic construction of a leakage-resilient authenticated key exchange (LR-AKE) protocol, starting from a leakage-resilient NIKE protocol. We formulate a new security model for LR-AKE protocols, which we call *Bounded-memory Before-the-Fact Leakage eCK* (BBFL-eCK) model. We then show a generic construction of BBFL-eCK-secure AKE protocol using a LR-NIKE in the bounded-memory leakage setting.

Our Model: Our security model for LR-AKE is a strong security model which addresses (bounded) leakage from the entire memory, which is stronger than the Only computation leaks information axiom [28]. We present an eCK-style [26] security model, suitably adjusted to the leakage setting.

Our Construction: We give the generic construction of leakage-resilient AKE from leakage-resilient NIKE in the bounded-memory leakage model. We adapt the construction of Bergsma et al. [7] to the setting of leakage. In particular Bergsma et al. [7] showed a construction of AKE protocols from a standard NIKE protocol and an existentially unforgeable signature scheme. We replace the standard NIKE with our leakage-resilient NIKE and the existentially unforgeable signature scheme with a signature scheme that is existentially unforgeable

under chosen message and leakage attacks [24]. We then show that the constructed AKE protocol is secure in our BBFL-eCK security model. The leakage rate of our construction is $1 - o(1)$, under appropriate choice of parameters.

Leakage-resilient LLKE. We show an extremely important practical application of leakage-resilient NIKE protocols. We study the leakage resiliency of low-latency key exchange (LLKE) protocols. In this paper we give a suitable leakage security model for LLKE protocols which we call the Bounded-memory leakage LLKE-ma (BL-LLKE-ma) model where "ma" stands for mutual authentication. We then present a generic construction of leakage-resilient LLKE (LR-LLKE) construction based on our LR-NIKE protocol in the bounded-memory leakage setting.

Our Model: The security of (standard) LLKE protocols has been recently analyzed by Hale et al. [21], under mutual authentication of the client well as the server. We give a leakage analogue of their security model. Our model allows the adversary to activate arbitrary protocol sessions between the clients and servers. Besides, the adversary can obtain the temporary or the main keys of a session of both the clients as well as the server, obtain the long-term secret key of clients and servers and also obtain bounded leakage from both the client and the server involved in the Test query. Finally in the test query (satisfying some freshness/validity conditions) the adversary has to guess the requested key from a random key.

Our Construction: Adopting the construction of Hale et al. [21] we show a generic construction of leakage-resilient LLKE protocol from a leakage-resilient NIKE protocol. In particular, we require a LR-NIKE scheme and a UF-CMLA secure signature scheme. Plugging them appropriately in our context we obtain the construction of leakage-resilient LLKE protocol. Moreover, the leakage rate enjoyed by our LLKE protocol is also optimal, i.e., $1 - o(1)$.

2 Preliminaries

Notations. For $a, b \in \mathbb{R}$, we let $[a, b] = \{x \in \mathbb{R} : a \leq x \leq b\}$; for $a \in \mathbb{N}$, we let $[a] = \{1, 2, \cdots, a\}$. If x is a string , we denote $|x|$ as the length of x. When x is chosen randomly from a set $\mathcal{X}$, we write $x \xleftarrow{\$} \mathcal{X}$. When A is an algorithm, we write $y \xleftarrow{\$} A(x)$ to denote a run of A on input x and output y; if A is randomized, then y is a random variable and $A(x; r)$ denotes a run of A on input x and randomness r. We denote the security parameter throughout by κ. An algorithm A is probabilistic polynomial-time (PPT) if A is randomized and for any input $x, r \in \{0,1\}^*$; the computation of $A(x; r)$ terminates in at most $poly(|x|)$ steps. Let $\mathbb{G}$ be a group of prime order p such that $\log_2(p) \geq \kappa$. Let g be a generator of $\mathbb{G}$, then for a (column/row) vector $C = (C_1, \cdots, C_n) \in \mathbb{Z}_p^n$, we denote by g^C the vector $C = (g^{C_1}, \cdots, g^{C_n})$. Furthermore, for a vector $D = (D_1, \cdots, D_n) \in \mathbb{Z}_p^n$, we denote by C^D the group element $X = \prod_{i=1}^n g^{C_i D_i} = g^{\sum_{i=1}^n C_i D_i}$. We say two random variables X and Y are ϵ-close statistically, if the statistical distance between them is at most ϵ, and is denoted by $X \approx_\epsilon Y$. On the other hand, if X and Y are computationally indistinguishable, we write $X \approx_c Y$

We refer to Appendix A.1 for the definitions of min-entropy, average conditional min-entropy, randomness extractors and basic results related to them.

2.1 Underlying Primitives for our Constructions.

For our construction of NIKE in the bounded-memory leakage setting we require a *bounded leakage-resilient chameleon hash function* (BLR-CHF). Leakage-resilient chameleon hash functions (LR-CHF) postulate that it is hard to find collisions, even when the adversary learns bounded leakage from the secret key/trapdoor. We refer the reader to Appendix A.2 for the formal definition of LR-CHF. We also need standard pseudo-random function (PRF) for this construction (please refer to Appendix A.3). A function F is a $(\epsilon_{\mathsf{prf}}, s_{\mathsf{prf}}, q_{\mathsf{prf}})$-secure PRF, if no adversary of size s_{prf} can distinguish F (instantiated with a random key) from a uniformly random function, except with negligible probability ϵ_{prf}.

For our construction of leakage-resilient AKE and leakage-resilient LLKE protocols we also need an *existentially unforgeable signature secure against chosen message and leakage attacks* (UF-CMLA). We refer the reader to Appendix A.4 for the formal definition. In all these definitions, the leakage functions can be arbitrarily and adaptively chosen by the adversary, with the only restriction that the output size of those functions are bounded by some leakage parameter.

3 Assumptions in Bilinear Group

In this paper we consider *Type-2* pairings over appropriate elliptic curve groups. Let $\mathcal{G}_2$ be a *type 2 pairing parameter generation* algorithm. It takes as input the security parameter 1^κ and outputs $gk = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, p, e, \psi)$ such that p is a prime, $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ are description of multiplicative cyclic groups of same order p, g_1, g_2 are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively, $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a non-degenerate efficiently computable bilinear map, and ψ is an efficiently computable isomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$, and that $g_1 = \psi(g_2)$.

The Decisional Bilinear Assumption over Type-2 Pairings (DBDH-2). Let $gk = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, p, e, \psi)$ be the output of the parameter generation algorithm $\mathcal{G}_2$ as above. We consider the following version of the DBDH-2 problem introduced by Galindo [18] and also used in [16]. Formally, we say that the DBDH-2 assumption holds for type-2 pairings if the advantage of the adversary $\mathcal{A}_{\mathsf{DBDH\text{-}2}}$ denoted by $\mathsf{Adv}^{\text{dbdh-2}}_{\mathcal{A}_{\mathsf{DBDH\text{-}2}},\mathcal{G}_2}(\kappa)$ is negligible, where

$$\mathsf{Adv}^{\text{dbdh-2}}_{\mathcal{A}_{\mathsf{DBDH\text{-}2}},\mathcal{G}_2}(\kappa) = |\Pr[\mathcal{A}(g_2, g_2^a, g_2^b, g_1^c, e(g_1,g_2)^{abc}) = 1] - \Pr[\mathcal{A}(g_2, g_2^a, g_2^b, g_1^c, e(g_1,g_2)^{z}) = 1]|.$$

where the probability is taken over the random choices of the algorithm $\mathcal{G}_2$ and the internal coin tosses of the algorithm $\mathcal{A}$.

4 Leakage-resilient Non-interactive Key Exchange

In this section, we present our definition of leakage-resilient non-interactive key exchange (LR-NIKE). We assume that the adversary is *not* allowed to register the same public key more than once. In practice, this can easily be ensured by requiring the Certification Authority (CA) to check for consistency whenever an individual attempts to register a public key in the system. So, w.lo.g, when we write pk_i for the i-th public key, we mean that pk_i is associated with the user with identifier $ID_i \in \mathcal{IDS}$, where $\mathcal{IDS}$ denotes the identity space. In the leakage-free scenario, this setting was also considered in the work of Freire et al. [16], which they called the Simplified(S)-NIKE. Also, we denote by $\mathcal{PK}$, $\mathcal{SK}$ and $\mathcal{SHK}$ the space of public keys, secret keys and shared keys respectively. Formally, a LR-NIKE scheme, $\mathsf{LR\text{-}NIKE}$, consists of a tuple of algorithms ($\mathsf{NIKEcommon\_setup}$, $\mathsf{NIKEgen}$, $\mathsf{NIKEkey}$) with the functionalities specified below:

1. $\mathsf{NIKEcommon\_setup}(1^\kappa, \lambda(\kappa))$: The setup algorithm takes as input the security parameter κ and the leakage bound $\lambda(\kappa)$ that can be tolerated by the NIKE scheme, and outputs a set of global parameters *params* of the system. We sometimes drop κ, and write λ instead of $\lambda(\kappa)$, when κ is clear from context.
2. $\mathsf{NIKEgen}(1^\kappa, params)$: The key generation algorithm is probabilistic and can be executed independently by all the users. It takes as input the security parameter κ and *params* and outputs a public/secret key pair $(pk, sk) \in \mathcal{PK} \times \mathcal{SK}$.
3. $\mathsf{NIKEkey}(pk_i, sk_j)$: The shared key generation algorithm takes the public key of user ID_i, namely pk_i and the secret key of user ID_j, namely sk_j, and outputs a shared key $shk_{ij} \in \mathcal{SHK}$ for the two keys or a failure symbol $\perp$ if $i = j$, i.e., if sk_j is the secret key corresponding to pk_i.

The *correctness* requirement states that for any two pairs (pk_i, sk_i) and (pk_j, sk_j), the shared keys computed by them should be identical.

4.1 The Generic Leakage-resilient Non-interactive Key Exchange ((·)LR-CKS-heavy) Security Model

In this section, we present the formal security model for leakage-resilient non-interactive key exchange (LR-NIKE). Our security model of LR-NIKE can be seen as generalization of the CKS-heavy security model introduced in Freire et al. [16] to appropriately model key leakage attacks. Our generic LR-NIKE ((·)LR-CKS-heavy) security model can be instantiated in two different ways which leads to two different security models, namely- *bounded* LR-CKS-heavy (BLR-CKS-heavy) model and *continuous* LR-CKS-heavy (CLR-CKS-heavy) model.

Our model allows the adversary to register *arbitrary* public keys into the system, provided they are distinct from each other and from the public keys of the honestly registered parties. The adversary can also issue

Extract queries to learn the private keys corresponding to the honestly generated public keys. The adversary can also learn the shared key between two honestly generated parties (via HonestReveal query) as long as both of them are not involved in the challenge/Test query. We also allow the adversary to learn the shared key between an honest party and a corrupt party (via CorruptReveal query). Apart from the above queries, the BLR-CKS-heavy model allows the adversary to obtain a *bounded* amount of leakage of the secret/private keys of the parties. Differently, the CLR-CKS-heavy model model allows the adversary to continuously obtain *arbitrarily* large amount of leakage from the secret keys of the parties, enforcing the restriction that the amount of leakage per observation is bounded. Finally, in the Test query, the adversary has to distinguish the real shared key between two honest parties from a random shared key. To prevent trivial wins, we enforce some natural restrictions on the Test query which we call the *validity* conditions. We also note that once the test query is asked by the adversary, he is not allowed to make further leakage queries on the corresponding parties involved in the test query (modeling *before-the-fact* leakage).

Remark 1. (Extract query vs. Leakage queries). By issuing Extract query, the adversary can learn the secret key of a party entirely. Separately, by issuing leakage queries the adversary gets bounded/continuous amount of leakage from the secret key. It may seem paradoxical to consider both Extract as well as Leakage queries at the same time. However, there are good reasons to consider both.

A non-leakage version of the $(\cdot)$LR-CKS-heavy model allow the adversary to corrupt the honest parties to obtain the corresponding secret keys. However, it disallows the adversary to corrupt any of the parties involved in the Test query. This is a natural restriction, since corrupting any of the parties involved in the test session will also allow the adversary to reconstruct the shared key of the test session and hence win the security game with certainty. But note that, in our $(\cdot)$LR-CKS-heavy model the adversary can also obtain (bounded/continuous) leakage from the secret keys of the parties involved in the test session in *addition* to corrupting other (non-test) honest parties in the system.

Hence, the $(\cdot)$LR-CKS-heavy model allows the adversary to obtain more information than a non-leakage version of $(\cdot)$LR-CKS-heavy model, namely CKS-heavy model [16] and hence is necessarily *stronger* than the CKS-heavy model.

Adversarial Powers. The $(\cdot)$LR-CKS-heavy security model is stated in terms of a security game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$. The adversary $\mathcal{A}$ is modeled as a probabilistic polynomial time (PPT) algorithm. We denote by $\Pi_{U,V}$ the protocol run between principal U, with intended principal V. Initially, the challenger $\mathcal{C}$ runs the NIKEcommon_setup algorithm to output the set of public parameters $params$, and gives $params$ to $\mathcal{A}$. The challenger $\mathcal{C}$ also chooses a random bit b in the beginning of the security game and answers all the legitimate queries of $\mathcal{A}$ until $\mathcal{A}$ outputs a bit b'. $\mathcal{A}$ is allowed to ask the following queries:

1. RegisterHonest queries$(1^\kappa, params)$: This query allows the adversary to register honest parties in the system. The challenger runs the NIKEgen algorithm to generate a key pair (pk_U, sk_U) and records the tuple $(honest, pk_U, sk_U)$. It then returns the public key pk_U to $\mathcal{A}$. We refer to the parties registered via this query as *honest* parties.
2. RegisterCorrupt queries(pk_U): This query allows the adversary to register arbitrary corrupt parties in the system. Here, $\mathcal{A}$ supplies a public key pk_U. The challenger records the tuple $(corrupt, pk_U, \perp)$. We demand that all the public keys involved in this query are distinct from one another and from the honestly generated public keys from above. The parties registered via this query are referred to as *corrupt*.
3. Extract queries(pk_U): In this query, the adversary $\mathcal{A}$ supplies the public key pk_U of a honest party. The challenger looks up the corresponding tuple $(honest, pk_U, sk_U)$ and returns the secret key sk_U to $\mathcal{A}$.
4. Reveal queries(pk_U, pk_V): This query can be categorized into two types– HonestReveal and CorruptReveal queries. Here the adversary supplies a pair of public keys pk_U and pk_V. In the HonestReveal query, both pk_U and pk_V are honestly registered, i.e., both of them correspond to honest parties; whereas in CorruptReveal query, one of the public key is registered as honest while the other is registered as corrupt. The challenger runs the NIKEkey algorithm using the secret key of the honest party (in case of HonestReveal query using the secret key of any one of the parties) and the public key of the other party, and returns the result to $\mathcal{A}$.
5. Test (pk_U, pk_V): Here $\mathcal{A}$ supplies two distinct public keys pk_U and pk_V, that were both registered as *honest*. If $pk_U = pk_V$, the challenger aborts and returns $\perp$. Otherwise, it uses the bit b to answer the query. If $b = 0$, the challenger runs the NIKEkey algorithm using the public key of one party say pk_U, and

the private key of the other party sk_V and returns the result to $\mathcal{A}$. If $b = 1$, the challenger samples a random shared key from $\mathcal{SHK}$, and returns that to $\mathcal{A}$.

6. Leakage queries: The leakage queries can be categorized into two types based upon the particular instantiation of our generic $(\cdot)$LR-CKS-heavy model.

   (a) *Bounded leakage queries*: In the BLR-CKS-heavy security model the total amount of leakage from the secret key of the underlying cryptographic primitives is bounded by leakage parameter $\lambda = \lambda(\kappa)$. Here the adversary $\mathcal{A}$ supplies the description of an arbitrary polynomial time computable functions f_i and a public key pk. The challenger computes $f_i(sk)$, where sk is the secret key corresponding to pk, and returns the output to $\mathcal{A}$. The adversary $\mathcal{A}$ can specify multiple such leakage functions as long as the leakage bound is not violated, i.e, $\sum_i |f_i(sk)| \leq \lambda(\kappa)$. Note that $\mathcal{A}$ can obtain λ bits of information/leakage from the secret key from each of the honest parties, *including* those involved in the Test queries.

   (b) *Continuous leakage queries*: In the CLR-CKS-heavy model the adversary can obtain unbounded leakage of the secret key of honest parties, with the only restriction that the amount of leakage per occurrence is bounded by leakage parameter. If the leakage bound of the secret keys of the parties per occurrence is $\lambda(\kappa)$ and the leakage function f_i outputs leakage bits of the i^{th} secret key, then for leakage resilience of the i^{th} secret key we need that $|f_i(sk_i)| \leq \lambda(\kappa)$.

$\mathcal{A}$'s queries may be made *adaptively* and are *arbitrary* in number. However, to prevent trivial wins the adversary should not be allowed to make certain queries to the parties involved in the Test query. We model this by requiring the Test query to be *valid*. We next give the definition of validity in our $(\cdot)$LR-CKS-heavy model (see Def. 1). Our condition of *validity* of the Test query is split into two parts depending on the particular instantiation of our $(\cdot)$LR-CKS-heavy model.

Definition 1 (λ-BLR-CKS-heavy validity). *We say that the* Test *query $\Pi_{U,V}$ between two parties U and V with public-secret key pairs (pk_U, sk_U) and (pk_V, sk_V) respectively is* valid *in the* BLR-CKS-heavy *model, if the following conditions hold:*

1. *The adversary $\mathcal{A}$ is not allowed to ask* $\mathsf{Extract}(pk_U)$ *or* $\mathsf{Extract}(pk_V)$ *queries.*
2. *The adversary $\mathcal{A}$ should not be allowed to ask* $\mathsf{HonestReveal}(pk_U, pk_V)$ *or* $\mathsf{HonestReveal}(pk_V, pk_U)$ *queries.*
3. *The total output length of all the Bounded Leakage queries queried by $\mathcal{A}$ to each party involved in the* Test *query, i.e., U and V is at most $\lambda(\kappa)$, i.e., $\sum_i |f_i(sk_U)| \leq \lambda(\kappa)$ and $\sum_i |f_i(sk_V)| \leq \lambda(\kappa)$.*
4. *After the* Test *query $\Pi_{U,V}$ is activated, the leakage functions $f_i(sk_U)$ and $f_i(sk_V)$ may not be asked by the adversary.*

Definition 2 (λ-CLR-CKS-heavy validity). *We say that the Test query is* valid *in the* CLR-CKS-heavy *model if: Conditions (1)-(2) of* Def. 1 *hold, and*

3. *The output length of the Continuous Leakage queries made by $\mathcal{A}$ per occurrence to each party involved in the* Test *query is at most $\lambda(\kappa)$, i.e., $|f_i(sk_U)| \leq \lambda(\kappa)$ and $|f_i(sk_V)| \leq \lambda(\kappa)$.*
4. *After the* Test *query $\Pi_{U,V}$ is activated, the leakage functions $f_i(sk_U)$ and $f_i(sk_V)$ may not be asked by the adversary.*

Remark 2. Note that in our validity conditions defined above (Defs. 1 and 2) we *do not* allow the adversary to make leakage queries to either of the parties involved in the Test query *after* the Test query is activated. This is because if the adversary is allowed to leak from the secret key after the Test query he can simply encode the specification of the shared key derivation function and other public keys into the leakage function. This reveals some information about the shared key allowing the adversary to successfully answer the challenge (distinguishing real from random). Such an impossibility result is also shown in the context of public-key encryption [29] and (interactive) key exchange protocols [4].

Security Game and Security Definition.

Definition 3 ((·)LR-CKS-heavy security game). *Security of a NIKE protocol in the generic* ((·)LR-CKS-heavy *model is defined using the following security game, which is played by a PPT adversary* $\mathcal{A}$ *against the protocol challenger* $\mathcal{C}$*.*

- **Stage 1:** The challenger $\mathcal{C}$ runs LR-NIKEcommon_setup algorithm to output the global parameters *params* and return it to $\mathcal{A}$.
- **Stage 2:** $\mathcal{A}$ may ask any number of RegisterHonest, RegisterCorrupt, Extract, HonestReveal, CorruptReveal, and Leakage queries adaptively.
- **Stage 3:** At any point of the game $\mathcal{A}$ may ask a Test query that is λ-(·)*LR-CKS-heavy valid*. The challenger chooses a random bit b to respond to this queries. If $b = 0$, the actual shared key between the respective pairs of parties involved in the corresponding test query is returned to $\mathcal{A}$. If $b = 1$, the challenger samples a random shared key from $\mathcal{SHK}$, records it for later and returns that to $\mathcal{A}$.
- **Stage 4:** $\mathcal{A}$ may continue asking RegisterHonest, RegisterCorrupt, Extract, HonestReveal, CorruptReveal, and Leakage queries adaptively, provided the Test query remains *valid*.
- **Stage 5:** At some point $\mathcal{A}$ outputs the bit $b' \leftarrow \{0,1\}$, which is its guess of the value b. $\mathcal{A}$ wins if $b' = b$.

Let $Succ_{\mathcal{A}}$ denote the event that $\mathcal{A}$ wins the above security game (Definition 3).

Definition 4 ((·)LR-CKS-heavy-security). *Let* q_H*,* q_C*,* q_E*,* q_{HR}*, and* q_{CR} *denote the number of* RegisterHonest, RegisterCorrupt, Extract, HonestReveal*, and* CorruptReveal *queries respectively. A NIKE protocol* Π *is said to be* (·)LR-CKS-heavy*-secure if there is no PPT algorithm* $\mathcal{A}$ *that can win the above* (·)LR-CKS-heavy *security game with non-negligible advantage. The advantage of an adversary* $\mathcal{A}$ *is defined as:*

$$\mathsf{Adv}^{(\cdot)\text{LR-CKS-heavy}}_{\Pi,\mathcal{A}}(\kappa, q_H, q_C, q_E, q_{HR}, q_{CR}) = |2\Pr(Succ_{\mathcal{A}}) - 1|.$$

4.2 Constructions of Leakage-resilient Non-interactive key exchange

In this section, we show our construction of leakage-resilient NIKE in the bounded-memory leakage model. We show that the pairing -based NIKE protocol of Friere et al. [16] (in the standard model) which is secure in the non-leakage setting is in fact insecure in the bounded memory leakage model, even if the adversary obtains a single bit of leakage on the secret key of the parties. This is illustrated in Appendix C.

Protocol BLR-NIKE: Construction of NIKE in the bounded-memory leakage model. Table 2 shows our construction of NIKE in the bounded-memory leakage model. The starting point of our construction is the NIKE protocol of [16]. Let $\mathcal{G}_2$ be a type 2 pairing parameter generation algorithm, i.e., it outputs $gk = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, p, e, \psi)$, $\mathsf{ChamH}_{\mathsf{hk}} : \{0,1\}^* \times \mathcal{R}_{cham} \to \mathbb{Z}_p$ be a (bounded) leakage-resilient chameleon hash function tolerating leakage bound up to $\lambda(\kappa)$ (denoted as λ-LR-CHF) indexed with the evaluation/hashing key hk and $\mathcal{R}_{cham}$ denotes the randomness space of the hash function. Also let $F : \{0,1\}^{\ell(\kappa)} \times \mathbb{Z}_p \to \mathbb{Z}_p$, $F' : \mathbb{Z}_p \times \{0,1\}^{\kappa} \to \mathbb{Z}_p$ be $(\epsilon_{\mathsf{prf}}, s_{\mathsf{prf}}, q_{\mathsf{prf}})$ and $(\epsilon'_{\mathsf{prf}}, s'_{\mathsf{prf}}, q'_{\mathsf{prf}})$ secure PRF families $\mathsf{Ext} : \mathbb{Z}_p \times \{0,1\}^s \to \{0,1\}^{\ell(\kappa)}$ be an average-case (v, ε)-extractor, with $v << \log p$. Namely, it has $\log p$ bits of input, s bit seed, and $\ell(\kappa)$-bit outputs, and for a random seed and input with v bits of min-entropy, the output is ε-away from a uniform $\ell(\kappa)$-bit string.

On Leak-free hardware assumption*:* As already stated in Section 1, we consider an additional assumption that each party involved in our LR-NIKE protocol has access to a *leak-free* secure hardware component. However, the role of this component in our protocol is only to store a short random seed of an extractor, which is enough to guarantee that the output of the extractor is statistically close to uniform. Hence, the usage of the leak-free hardware in our context is *minimal*. The above assumption seems to be necessary for our protocol, since otherwise the adversary could leak from the extracted value itself by knowing the seed. Removing this leak-free assumption is a desirable and important problem that we leave open.

Theorem 1. *Let* $\mathsf{ChamH}_{\mathsf{hk}}$ *be a family of bounded leakage-resilient chameleon hash function* (BLR-CHF)*,* F *and* F' *be* $(\epsilon_{\mathsf{prf}}, s_{\mathsf{prf}}, q_{\mathsf{prf}})$ *and* $(\epsilon'_{\mathsf{prf}}, s'_{\mathsf{prf}}, q'_{\mathsf{prf}})$ *secure PRFs and* Ext *be a* (v, ε)*-strong average case randomness extractor where* p *is the order of the underlying groups* $\mathbb{G}_1$*,* $\mathbb{G}_2$ *and* $\mathbb{G}_T$*. Then the above NIKE protocol* BLR-NIKE *is* BLR-CKS-heavy*-secure assuming the intractability of the* DBDH-2 *assumption with respect to*

Party ID_A		Party ID_B
	$\underline{\mathsf{NIKEcommon\_setup}(1^\kappa)}$	
	$gk \xleftarrow{\$} \mathcal{G}_2(1^\kappa)$	
	where $gk = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, p, e, \psi)$	
	$\alpha, \beta, \gamma, \delta \xleftarrow{\$} \mathbb{G}_1^*$	
	$(\mathsf{hk}, \mathsf{ck}) \leftarrow \mathsf{Cham.KeyGen}(1^\kappa, \lambda)$	
	$params := (gk, \alpha, \beta, \gamma, \delta, \mathsf{hk})$	
	Return $params$	
	$\underline{\mathsf{NIKEgen}(1^\kappa, params)}$	
$x_A, r_A \xleftarrow{\$} \mathbb{Z}_p$; $r'_A \xleftarrow{\$} \mathcal{R}_{cham}$;		$x_B, r_B \xleftarrow{\$} \mathbb{Z}_p$; $r'_B \xleftarrow{\$} \mathcal{R}_{cham}$;
Sample $s_A \xleftarrow{\$} \{0,1\}^s$, and *store s_A in leak-free component*;		Sample $s_B \xleftarrow{\$} \{0,1\}^s$, and *store s_B in leak-free component*;
$\widehat{x_A} \leftarrow \mathsf{Ext}(x_A, s_A)$;		$\widehat{x_B} \leftarrow \mathsf{Ext}(x_B, s_B)$;
$x'_A \leftarrow F_{\widehat{x_A}}(r_A) + F'_{r_A}(1^\kappa)$;		$x'_B \leftarrow F_{\widehat{x_B}}(r_B) + F'_{r_B}(1^\kappa)$;
$Z_A \leftarrow g_2^{x'_A}$;		$Z_B \leftarrow g_2^{x'_B}$;
$t_A \leftarrow \mathsf{ChamH}_{\mathsf{hk}}(Z_A \| ID_A; r'_A)$;		$t_B \leftarrow \mathsf{ChamH}_{\mathsf{hk}}(Z_B \| ID_B; r'_B)$;
$Y_A \leftarrow \alpha\beta^{t_A}\gamma^{{t_A}^2}$; $X_A \leftarrow Y_A^{x'_A}$		$Y_B \leftarrow \alpha\beta^{t_B}\gamma^{{t_B}^2}$; $X_B \leftarrow Y_B^{x'_B}$
$pk_A \leftarrow (X_A, Z_A, r'_A, r_A)$; $sk_A \leftarrow x_A$		$pk_B \leftarrow (X_B, Z_B, r'_B, r_B)$; $sk_B \leftarrow x_B$
$\underline{\mathsf{NIKEkey}(pk_B, sk_A)}$		$\underline{\mathsf{NIKEkey}(pk_A, sk_B)}$
If $pk_A = pk_B$, return $\perp$		If $pk_B = pk_A$, return $\perp$
Parse pk_B as (X_B, Z_B, r'_B, r_B);		Parse pk_A as (X_A, Z_A, r'_A, r_A);
$t_B \leftarrow \mathsf{ChamH}_{\mathsf{hk}}(Z_B \| ID_B; r'_B)$;		$t_A \leftarrow \mathsf{ChamH}_{\mathsf{hk}}(Z_A \| ID_A; r'_A)$
If $e(X_B, g_2) \neq e(\alpha\beta^{t_B}\gamma^{{t_B}^2}, Z_B)$,		If $e(X_A, g_2) \neq e(\alpha\beta^{t_A}\gamma^{{t_A}^2}, Z_A)$,
then $shk_{A,B} \leftarrow \perp$		then $shk_{A,B} \leftarrow \perp$
$x'_A \leftarrow F_{\widehat{x_A}}(r_A) + F'_{r_A}(1^\kappa)$;		$x'_B \leftarrow F_{\widehat{x_B}}(r_B) + F'_{r_B}(1^\kappa)$;
$shk_{AB} \leftarrow e(\delta^{x'_A}, Z_B)$		$shk_{AB} \leftarrow e(\delta^{x'_B}, Z_A)$

Table 2. LR-NIKE Protocol in the Bounded Leakage model(BLR-NIKE)

the parameter generator $\mathcal{G}_2$. In particular, let $\mathcal{A}$ be an adversary against the NIKE protocol BLR-NIKE *in the* BLR-CKS-heavy *security model making q_H number of* RegisterHonest *user queries. Then using it we can construct an adversary $\mathcal{A}_{\mathsf{DBDH\text{-}2}}$ against the* DBDH-2 *problem such that:*

$$\mathsf{Adv}^{\text{BLR-CKS-heavy}}_{\mathsf{BLR\text{-}NIKE},\mathcal{A}}(\kappa) \leq q_H^2 \left(2\varepsilon + \epsilon_{\mathsf{prf}} + \epsilon'_{\mathsf{prf}} + \mathsf{Adv}^{\text{coll}}_{\mathcal{A},\mathsf{ChamH}}(\kappa) + \mathsf{Adv}^{\text{dbdh-2}}_{\mathcal{A}_{\mathsf{DBDH\text{-}2}},\mathcal{G}_2}(\kappa)\right).$$

Proof. The proof of this theorem will proceed via the game hopping technique [33]: define a sequence of games and relate the adversary's advantage of distinguishing each game from the previous game to the advantage of breaking one of the underlying cryptographic primitive. Let $\text{Adv}_{Game_\delta}(\mathcal{A})$ denote the advantage of the adversary $\mathcal{A}$ in Game δ.

Game 0: This is the original security game with adversary $\mathcal{A}_{\mathsf{DBDH\text{-}2}}$. When the Test query is asked, the Game 0 challenger chooses a random bit $b \xleftarrow{\$} \{0,1\}$. If $b = 0$, the real shared key is given to $\mathcal{A}$, otherwise a random value chosen from the shared key space is given.

$$\text{Adv}_{Game_0}(\mathcal{A}) = \mathsf{Adv}^{\text{BLR-CKS-heavy}}_{\mathsf{BLR\text{-}NIKE},\mathcal{A}}(\kappa).$$

Game 1: Initially $\mathcal{A}_{\mathsf{DBDH\text{-}2}}$ chooses two identities $ID_A, ID_B \in [q_H]$, where q_H denotes the number of RegisterHonest queries made by $\mathcal{A}_{\mathsf{NIKE}}$. Effectively $\mathcal{A}_{\mathsf{DBDH\text{-}2}}$ is guessing that ID_A and ID_B to be honestly registered by $\mathcal{A}_{\mathsf{NIKE}}$ will be involved in the Test query later. When $\mathcal{A}_{\mathsf{NIKE}}$ makes its Test query on a pair of

identities $\{ID_I, ID_J\}$, $\mathcal{A}_{\mathsf{DBDH\text{-}2}}$ checks if $\{ID_I, ID_J\} = \{ID_A, ID_B\}$. If so, it continues with the simulation and gives the result to $\mathcal{A}_{\mathsf{NIKE}}$; else it aborts the simulation.

$$\mathrm{Adv}_{Game_1}(\mathcal{A}) \geq \mathrm{Adv}_{Game_0}(\mathcal{A})/q_H^2.$$

Game 2: This game is identical to the previous game, except that the challenger changes the way how the output of the extractor is computed. In particular, instead of computing $\widehat{x_A} \leftarrow \mathsf{Ext}(x_A, s_A)$ and $\widehat{x_B} \leftarrow \mathsf{Ext}(x_B, s_B)$, the challenger chooses a uniformly random $\widehat{x_A}, \widehat{x_B} \leftarrow \{0,1\}^{\ell(\kappa)}$. Game 0 and Game 1 are indistinguishable by the property of the strong average case randomness extractor. Suppose that the adversary obtains at most $\lambda = \lambda(\kappa)$ bits of leakage from the secret keys x_A and x_B of parties A and B respectively. Since Ext can work with inputs that have min-entropy $v << \log p$, even given the bounded leakage of λ bits, we have that $\left(x_A, s_A, \mathsf{Ext}(x_A, s_A)\right) \approx_\varepsilon \left(x_A, s_A, U_{\ell(\kappa)}\right)$ and $\left(x_B, s_B, \mathsf{Ext}(x_B, s_B)\right) \approx_\varepsilon \left(x_B, s_B, U_{\ell(\kappa)}\right)$, where $U_{\ell(\kappa)}$ denotes the uniform distribution over $\{0,1\}^{\ell(\kappa)}$. Recall that, in our construction the seeds s_A and s_B are stored in the *leak-free* hardware component, and hence are outside the view of the adversary. Thus, it is possible to replace the output the extractor with a uniformly random value in this game. This is the *only* place where we require the leak-free assumption in our proof.

$$\left|\mathrm{Adv}_{Game_2}(\mathcal{A}) - \mathrm{Adv}_{Game_1}(\mathcal{A})\right| \leq 2\varepsilon.$$

Game 3: This game is identical to the previous game, except that the challenger changes the way how the PRFs are computed. In particular, instead of computing $x'_A \leftarrow F_{\widehat{x_A}}(r_A) + F'_{r_A}(1^\kappa)$ and $x'_B \leftarrow F_{\widehat{x_B}}(r_B) + F'_{r_B}(1^\kappa)$, the challenger computes $x'_A \leftarrow RF(r_A) + F'_{r_A}(1^\kappa)$ and $x'_B \leftarrow RF(r_B) + F'_{r_B}(1^\kappa)$, where RF is random function with the same range as F. If $\mathcal{A}$ can distinguish the difference between Game 2 and Game 3, then $\mathcal{A}$ can be used as a subroutine to construct a distinguisher $\mathcal{D}$ between the PRF $F : \{0,1\}^{\ell(\kappa)} \times \mathbb{Z}_p \rightarrow \mathbb{Z}_p$ and and a random function RF.

$$|\mathrm{Adv}_{Game_3}(\mathcal{A}) - \mathrm{Adv}_{Game_2}(\mathcal{A})| \leq \epsilon_{\mathsf{prf}} + \epsilon'_{\mathsf{prf}}.$$

Game 4: This game is identical to the previous game, except that the challenger nows samples x'_A and x'_B randomly. In particular, instead of computing $x'_A \leftarrow F_{\widehat{x_A}}(r_A) + F'_{r_A}(1^\kappa)$ and $x'_B \leftarrow F_{\widehat{x_B}}(r_B) + F'_{r_B}(1^\kappa)$, the challenge samples $x'_A, x'_B \xleftarrow{\$} \mathbb{Z}_p$. Note that x'_A and x'_B are identically distributed in both Game 3 and Game 4, and hence both these games are identical from the view of an adversary.

$$\mathsf{Adv}_{Game_4}(\mathcal{A}) = \mathsf{Adv}_{Game_3}(\mathcal{A}).$$

Game 5: In this game the challenger changes the way in which it answers $\mathsf{RegisterCorrupt}$ queries. In particular let, ID_A and ID_B be identities of two honest parties involved in the Test query with public keys $(X_A, Z_A, r'_A, r_A, s_A)$ and $(X_B, Z_B, r'_B, r_B, s_B)$ respectively. Let ID_D be the identity of the party with public key (X_D, Z_D, r'_D, r_D) that is subject to a $\mathsf{RegisterCorrupt}$ query. If $t_D = \mathsf{ChamH}_{\mathsf{hk}}(Z_A || ID_A; r'_A)$ or $t_D = \mathsf{ChamH}_{\mathsf{hk}}(Z_B || ID_B; r'_B)$, the challenger aborts. Note that if the above happens, then the challenger has successfully found a collision of the chameleon hash function. By the difference lemma [32] we have:

$$\left|\mathrm{Adv}_{Game_5}(\mathcal{A}) - \mathrm{Adv}_{Game_4}(\mathcal{A})\right| \leq \mathsf{Adv}^{\mathrm{coll}}_{\mathcal{A},\mathsf{ChamH}}(\kappa).$$

Game 6: In this game the DBDH-2 adversary $\mathcal{A}_{\mathsf{DBDH\text{-}2}}$ receives as input $(g_2, g_2^a, g_2^b,$
$g_1^c, T)$ as input, and its objective is to determine if $T = e(g_1, g_2)^{sbc}$ or a random element from $\mathbb{G}_T$, where g_1 and g_2 are generators of the group $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively and a, b, c are random elements from $\mathbb{Z}_p$. We now describe how $\mathcal{A}_{\mathsf{DBDH\text{-}2}}$ sets up the environment for $\mathcal{A}_{\mathsf{NIKE}}$ and simulates all its queries properly.

$\mathcal{A}_{\mathsf{DBDH\text{-}2}}$ runs $\mathsf{Cham.KeyGen}(1^\kappa, \lambda)$ to obtain a key pair for a chameleon hash function, $(\mathsf{hk}, \mathsf{ck})$. It then chooses two messages $m_1, m_2 \leftarrow \{0,1\}^*$ and $r_1, r_2 \leftarrow \mathcal{R}_{cham}$, where $\mathcal{R}_{cham}$ is the randomness space of the chameleon hash function. $\mathcal{A}_{\mathsf{DBDH\text{-}2}}$ then computes the values $t_A = \mathsf{Cham.Eval}(m_1; r_1)$ and $t_B = \mathsf{Cham.Eval}(m_2; r_2)$.

Let us define a polynomial of degree 2 $p(t) = p_0 + p_1 t + p_2 t^2$ over $\mathbb{Z}_p$ such that t_A and t_B are the roots of $p(t)$, i.e., $p(t_A) = 0$ and $p(t_B) = 0$. Also, let $q(t) = q_0 + q_1 t + q_2 t^2$ be a random polynomial of degree 2 over $\mathbb{Z}_p$. $\mathcal{A}_{\mathsf{DBDH\text{-}2}}$ then sets $\alpha = (g_1^c)^{p_0} \cdot g_1^{q_0}$, $\beta = (g_1^c)^{p_1} \cdot g_1^{q_1}$, $\gamma = (g_1^c)^{p_2} \cdot g_1^{q_2}$ and $\delta = g_1^c$ (g_1^c was obtained as input of the hard problem instance). Note that, since $p_i, q_i \leftarrow \mathbb{Z}_p$ are randomly chosen, the values of α, β and γ are also random. Also note that $\alpha\beta^t\gamma^{t^2} = (g_1)^{p_1(t)} g_1^{q(t)}$. In particular $Y_A = g_1^{q(t_A)}$ and $Y_B = g_1^{q(t_B)}$ (since $p(t_A) = p(t_B) = 0$). $\mathcal{A}_{\mathsf{DBDH\text{-}2}}$ then simulates all the queries of $\mathcal{A}_{\mathsf{NIKE}}$ as follows:

- $\mathsf{RegisterHonest}$: When $\mathcal{A}_{\mathsf{DBDH\text{-}2}}$ receives as input a $\mathsf{RegisterHonest}$ user query from $\mathcal{A}_{\mathsf{NIKE}}$ for a party with identity ID, it fist checks whether $ID \in \{ID_A, ID_B\}$. Depending upon the result it does the following:

– If $ID \notin \{ID_A, ID_B\}$, $\mathcal{A}_{\mathsf{DBDH\text{-}2}}$ runs $\mathsf{NIKE.gen}$ to generate a pair of keys (pk, sk), and makes returns pk to $\mathcal{A}_{\mathsf{NIKE}}$.

– If $ID \in \{ID_A, ID_B\}$, $\mathcal{A}_{\mathsf{DBDH\text{-}2}}$ does the following. Without loss of generality let $ID = ID_A$. Now, $\mathcal{A}_{\mathsf{DBDH\text{-}2}}$ uses the trapdoor ck of the chameleon hash to produce $r'_A \in \mathcal{R}_{cham}$ such that $\mathsf{Cham.Eval}(g_2^a || ID_A; r'_A) =$
$\mathsf{Cham.Eval}(m_1; r_1)$. Note that, by the random trapdoor collision property of the chameleon hash function, r'_A is uniformly distributed over $\mathcal{R}_{cham}$ and also independent of r_1. Similarly when $ID = ID_B$, $\mathcal{A}_{\mathsf{DBDH\text{-}2}}$ uses the trapdoor ck to produce $r'_B \in \mathcal{R}_{cham}$ such that $\mathsf{Cham.Eval}(g_2^b || ID_B; r'_B) = \mathsf{Cham.Eval}(m_2; r_2)$. The value r'_B is also uniformly distributed over $\mathcal{R}_{cham}$ and also independent of r_2. $\mathcal{A}_{\mathsf{DBDH\text{-}2}}$ then sets:

$$pk_A = (\psi(g_2^a)^{q(t_A)}, g_2^a, r'_A, r_A) \text{ and } pk_B = (\psi(g_2^b)^{q(t_B)}, g_2^B, r'_B, r_B).$$

where $r_A, r_B \leftarrow \mathbb{Z}_p$. Note that these are correct public keys since $p(t_A) = p(t_B) = 0$.

- $\mathsf{RegisterCorrupt}$: Here $\mathcal{A}_{\mathsf{DBDH\text{-}2}}$ receives as input a public key pk and an identity string ID from $\mathcal{A}_{\mathsf{NIKE}}$. If $ID \in \{ID_A, ID_B\}$, $\mathcal{A}_{\mathsf{DBDH\text{-}2}}$ aborts as in the original attack game.
- $\mathsf{HonestReveal}$: When $\mathcal{A}_{\mathsf{NIKE}}$ supplies identities of two honest parties, ID and ID' say, $\mathcal{A}_{\mathsf{DBDH\text{-}2}}$ checks if $\{ID, ID'\} = \{ID_A, ID_B\}$. If this happens, $\mathcal{A}_{\mathsf{DBDH\text{-}2}}$ aborts. Else, if $\{ID, ID'\} \cap \{ID_A, ID_B\} \leq 1$, there are three cases:
  - $ID \cap \{ID_A, ID_B\} \neq \phi$ and $ID' \cap \{ID_A, ID_B\} = \phi$. In this case, the challenger $\mathcal{A}_{\mathsf{DBDH\text{-}2}}$ runs $\mathsf{NIKE.key}(pk_{ID}, sk_{ID'})$ to produce the shared key $shk_{ID,ID'}$. Note that $\mathcal{A}_{\mathsf{DBDH\text{-}2}}$ can do this since it knows the secret key $sk_{ID'}$ of the party ID'. $\mathcal{A}_{\mathsf{DBDH\text{-}2}}$ then gives $shk_{ID,ID'}$ to $\mathcal{A}_{\mathsf{NIKE}}$.
  - $ID \cap \{ID_A, ID_B\} = \phi$ and $ID' \cap \{ID_A, ID_B\} \neq \phi$. In this case, the challenger $\mathcal{A}_{\mathsf{DBDH\text{-}2}}$ runs $\mathsf{NIKE.key}(pk_{ID'}, sk_{ID})$ to produce the shared key $shk_{ID,ID'}$. Note that $\mathcal{A}_{\mathsf{DBDH\text{-}2}}$ can do this since it knows the secret key $sk_{ID'}$ of the party ID'. $\mathcal{A}_{\mathsf{DBDH\text{-}2}}$ then gives $shk_{ID,ID'}$ to $\mathcal{A}_{\mathsf{NIKE}}$.
  - $\{ID, ID'\} \cap \{ID_A, ID_B\} = \phi$. In this case, the challenger $\mathcal{A}_{\mathsf{DBDH\text{-}2}}$ runs $\mathsf{NIKE.key}(pk_{ID'}, sk_{ID})$ (it can use $sk_{ID'}$ also) to produce the shared key $shk_{ID,ID'}$. $\mathcal{A}_{\mathsf{DBDH\text{-}2}}$ then gives $shk_{ID,ID'}$ to $\mathcal{A}_{\mathsf{NIKE}}$.
- $\mathsf{CorruptReveal}$: When $\mathcal{A}_{\mathsf{NIKE}}$ supplies two identities ID and ID' where ID was registered as corrupt and ID' was registered as honest, $\mathcal{A}_{\mathsf{DBDH\text{-}2}}$ checks if $ID' \in \{ID_A, ID_B\}$. If $ID' \notin \{ID_A, ID_B\}$, $\mathcal{A}_{\mathsf{DBDH\text{-}2}}$ runs $\mathsf{NIKE.key}(pk_{ID}, sk_{ID'})$ to obtain $shk_{ID,ID'}$ and returns it to $\mathcal{A}_{\mathsf{NIKE}}$. However, if $ID' \in \{ID_A, ID_B\}$, $\mathcal{A}_{\mathsf{DBDH\text{-}2}}$ checks whether the public key $pk_{ID} = (X_{ID}, Z_{ID}, r'_{ID}, r_{ID})$ by checking the pairing. This makes sure that pk_{ID} is of the form $(Y_{ID}^d, g_2^d, r'_D, r_D)$ for some $d \in \mathbb{Z}_p$, where $Y_D = (g_1^c)^{p(t_{ID})} g_1^{q(t_{ID})}$, $r_D \leftarrow \mathbb{Z}_p$ and $r'_D \leftarrow \mathcal{R}_{cham}$. This means that $X_{ID} = (g_1^{cd})^{p(t_{ID})} g_1^{dq(t_{ID})}$. From this the value g_1^{cd} can be computed as:

$$g_1^{cd} = (X_{ID}/\psi(Z_{ID})^{q(t_{ID})})^{1/p(t_{ID}) \mod p}.$$

Note that the value $1/p(t_{ID})$ is well defined since $p(t_{ID}) \neq 0 \mod p$. Also note that $t_{ID} \neq t_A, t_B$, since we have already eliminated the hash collisions.
Assume w.l.o.g. that $ID' = ID_A$. So writing the public key of ID_A as (Y_A, Z_A, r'_A, r_A), the shared key between ID_A and ID is given by $shk_{ID_A,ID} = e(g_1^{cd}, Z_A)$.
- $\mathsf{Leakage}$ queries: The adversary $\mathcal{A}_{\mathsf{NIKE}}$ may specify arbitrary polynomial time computable function(s) f_i to leak from the secret keys x_A and x_B. The challenger $\mathcal{A}_{\mathsf{DBDH\text{-}2}}$ forwards the functions f_i to its leakage oracle and forwards the answers to $\mathcal{A}_{\mathsf{NIKE}}$.
- Test query: Here, $\mathcal{A}_{\mathsf{DBDH\text{-}2}}$ returns T.

This completes the description of simulation by $\mathcal{A}_{\mathsf{DBDH\text{-}2}}$. If $\mathcal{A}_{\mathsf{NIKE}}$ can distinguish between real and random key in Game 4, then it is equivalent to solving the DBDH-2 problem. To see this, note that for user ID_A we have $Z_A = g_2^a$ and $X_A = \psi(Z_A)^{q(t_A)}$, and for user ID_B we have $Z_B = g_2^b$ and $X_B = \psi(Z_B)^{q(t_B)}$. Hence, $shk_{ID_A,ID_B} = e((g_1^c)^b, Z_A) = e((g_1^c)^a, Z_B) = e(g_1, g_2)^{abc}$.

Since the simulation done by $\mathcal{A}_{\mathsf{DBDH\text{-}2}}$ is perfect, we have:

$$\mathrm{Adv}_{Game_6}(\mathcal{A}) = \mathrm{Adv}_{Game_5}(\mathcal{A})$$

Game 7. In this game the challenger $\mathcal{A}_{\mathsf{DBDH\text{-}2}}$ chooses T randomly from the target group $\mathbb{G}_T$. Since T is now completely independent of the challenge bit, we have $\Pr(Succ_{Game6}) = 1/2$. Game 5 and Game 6 are identical unless adversary $\mathcal{A}_{\mathsf{DBDH\text{-}2}}$ can distinguish $e(g_1, g_2)^{abc}$ from a random element. So, we have,

$$|\mathrm{Adv}_{Game_7}(\mathcal{A}) - \mathrm{Adv}_{Game_6}(\mathcal{A})| \leq \mathsf{Adv}^{\text{dbdh-2}}_{\mathcal{A}_{\mathsf{DBDH\text{-}2}}, \mathcal{G}_2}(\kappa).$$

By combining all the above expression from Game 0 - Game 7 we have the following. We use G_i to denote $Game_i$.

$$\begin{aligned}
\mathsf{Adv}^{\text{BLR-CKS-heavy}}_{\mathsf{BLR\text{-}NIKE},\mathcal{A}_{\mathsf{NIKE}}}(\kappa) &= \mathrm{Adv}_{G_0}(\mathcal{A}) \\
&\leq q_H^2 \cdot (\mathrm{Adv}_{G_1}(\mathcal{A})) \\
&\leq q_H^2 \cdot (\mathrm{Adv}_{G_2}(\mathcal{A}) + 2\varepsilon) \\
&\leq q_H^2 \cdot (\mathrm{Adv}_{G_3}(\mathcal{A}) + 2\varepsilon + \epsilon_{\mathsf{prf}} + \epsilon'_{\mathsf{prf}}) \\
&= q_H^2 \cdot (\mathrm{Adv}_{G_4}(\mathcal{A}) + 2\varepsilon + \epsilon_{\mathsf{prf}} + \epsilon'_{\mathsf{prf}}) \\
&\leq q_H^2 \cdot (\mathrm{Adv}_{G_5}(\mathcal{A}) + 2\varepsilon + \epsilon_{\mathsf{prf}} + \epsilon'_{\mathsf{prf}} + \mathsf{Adv}^{\text{coll}}_{\mathcal{A},\mathsf{ChamH}}(\kappa)) \\
&= q_H^2 \cdot (\mathrm{Adv}_{G_6}(\mathcal{A}) + 2\varepsilon + \epsilon_{\mathsf{prf}} + \epsilon'_{\mathsf{prf}} + \mathsf{Adv}^{\text{coll}}_{\mathcal{A},\mathsf{ChamH}}(\kappa)) \\
&\leq q_H^2 \cdot (\mathrm{Adv}_{G_7}(\mathcal{A}) + 2\varepsilon + \epsilon_{\mathsf{prf}} + \epsilon'_{\mathsf{prf}} + \mathsf{Adv}^{\text{coll}}_{\mathcal{A},\mathsf{ChamH}}(\kappa) + \mathsf{Adv}^{\text{dbdh-2}}_{\mathcal{A}_{\text{DBDH-2}},\mathcal{G}_2}(\kappa)) \\
&\leq q_H^2 \cdot (2\varepsilon + \epsilon_{\mathsf{prf}} + \epsilon'_{\mathsf{prf}} + \mathsf{Adv}^{\text{coll}}_{\mathcal{A},\mathsf{ChamH}}(\kappa) + \mathsf{Adv}^{\text{dbdh-2}}_{\mathcal{A}_{\text{DBDH-2}},\mathcal{G}_2}(\kappa))
\end{aligned}$$

Thus we have,

$$\mathsf{Adv}^{\text{BLR-CKS-heavy}}_{\mathsf{BLR\text{-}NIKE},\mathcal{A}_{\mathsf{NIKE}}}(\kappa) \leq q_H^2 \cdot \left(2\varepsilon + \epsilon_{\mathsf{prf}} + \epsilon'_{\mathsf{prf}} + \mathsf{Adv}^{\text{coll}}_{\mathcal{A},\mathsf{ChamH}}(\kappa) + \mathsf{Adv}^{\text{dbdh-2}}_{\mathcal{A}_{\text{DBDH-2}},\mathcal{G}_2}(\kappa)\right) \quad \square$$

Leakage Tolerance of Protocol BLR-NIKE. The order of the groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are p. Note that, although the secret key in our protocol BLR-NIKE may appear to be a single field element, but in the actual instantiation of the protocol the secret key is a tuple of $n+1$ field elements. This is because the secret key of the concrete instantiation of the BLR-CHF of Wang and Tanaka [34] consists of n field elements which also corresponds to the number of generators in the construction of [34]. The leakage tolerance of the BLR-CHF is $\lambda' = ((n-1)\log(p) - \omega(\log\kappa))$ as shown in [34]. The size of the secret key of the BLR-CHF is $L = n\log p$. So, for sufficiently large n, the leakage rate $\eta = \lambda'/L$ of the BLR-CHF approaches $1 - o(1)$. We also consider a very good randomness extractor that can work with inputs that have min-entropy $v << \log p$ and produces outputs whose distance from uniform $\ell(\kappa)$-bit strings is $\varepsilon < 2^{-\ell(\kappa)}$. The size of the secret key x_A (respectively x_B) of our NIKE construction is $\log p$ (apart from the size of the secret key of λ'-LR-CHF, i.e, $n\log p$). So the leakage tolerated from x_A (respectively x_B) is at least $\log p - v \approx \log p$. Hence, the overall leakage tolerated by our construction is $\lambda \approx ((n-1)\log(p) - \omega(\log\kappa)) + \log p \approx n\log(p) - \omega(\log\kappa)$. The overall size of the secret key of our construction is $L' = (n+1)\log p$. So the overall leakage rate of our construction is $\eta' = \lambda/L' = 1 - o(1)$, for sufficiently large n.

5 Constructions of various cryptographic primitives from leakage-resilient NIKE

Now, we show many potential applications of leakage-resilient NIKE.

5.1 Leakage-resilient Adaptive Chosen Ciphertext secure PKE

We now present our construction of LR-IND-CCA-2-secure PKE scheme from a BLR-CKS-heavy-secure LR-NIKE scheme. Actually, we show how to construct a LR-IND-CCA-2-secure key encapsulation mechanism (KEM) given such a NIKE. Before proceeding with the construction, we give the LR-IND-CCA-2 security model for KEMs.

Leakage-resilient Chosen-Ciphertext security for KEM. We say that a KEM $\Gamma = (\mathsf{KEM.Setup}, \mathsf{KEM.Gen}, \mathsf{KEM.Encap}, \mathsf{KEM.Decap})$ satisfies *correctness* if for all $pub \xleftarrow{\$} \mathsf{Setup}(1^\kappa)$, $(pk_{\text{KEM}}, sk_{\text{KEM}}) \xleftarrow{\$} \mathsf{Gen}(1^\kappa, pub)$, and $(C, K) \leftarrow \mathsf{Encap}(pk_{\text{KEM}})$, it holds that $\Pr[\mathsf{Decap}(sk_{\text{KEM}}, C) = K] = 1$ (where the randomness is taken over the internal coin tosses of algorithm KEM.Gen and KEM.Encap).

LR-IND-CCA-2 security. We now turn to defining indistinguishability under adaptive chosen-ciphertext and leakage attacks in the bounded-memory leakage setting (BLR-IND-CCA-2).

Definition 5 (BLR-IND-CCA-2 security). Let $\kappa \in \mathbb{N}$ and $\lambda = \lambda(\kappa)$ be parameters. We say that KEM $\Gamma = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Encap}, \mathsf{Decap})$ is λ-BLR-CCA-2-secure if for all PPT adversaries A there exists a negligible $\nu(\kappa)$ such that

$$\left|\Pr[\mathbf{Exp}^{\text{BLR-IND-CCA-2}}_{\Gamma,\mathsf{A}}(\kappa,\lambda) = 1]\right| \leq \nu(\kappa).$$

where the experiment $\mathbf{Exp}^{\text{BLR-IND-CCA-2}}_{\Gamma,\mathsf{A}}(\kappa,\lambda)$ is defined in Table 3.

Experiment $\mathbf{Exp}^{\text{BLR-IND-CCA-2}}_{\Gamma,\mathsf{A}}(\kappa,\lambda)$	Oracle $\mathsf{Decap}^*(C)$
$pub \xleftarrow{\$} \mathsf{Setup}(1^\kappa)$;	If $C = C^*$, abort
$(pk, sk) \xleftarrow{\$} \mathsf{Gen}(1^\kappa, pub)$, $L \leftarrow \emptyset$, $b \xleftarrow{\$} \{0,1\}$;	Return $\mathsf{Decap}(sk, C)$
$(C^*, K_1^*) \leftarrow \mathsf{Encap}(pk)$; $K_0^* \xleftarrow{\$} \mathcal{K}$;	
$b' \leftarrow \mathsf{A}^{\mathsf{Decap}^*, O^\lambda_{sk}(\cdot)}(pk, C^*, K_b^*)$;	Oracle $O^\lambda_{sk}(f)$
If $b' = b$, then return 1, else return 0	If $L + \lvert f(sk)\rvert \leq \lambda(\kappa)$
	Return $f(sk)$

Table 3. Experiment defining LR-CCA-2 security of KEM

Generic Construction of leakage-resilient KEM. We now show the construction of leakage-resilient CCA-2-secure KEM Γ = (KEM.Setup, KEM.Gen, KEM.Encap, KEM.Decap) from a leakage-resilient NIKE (see Figure 1).

Let LR-NIKE = (NIKEcommon_setup, NIKEgen, NIKEkey) be a BLR-CKS-heavy NIKE with leakage rate $1 - o(1)$.

1. $\mathsf{KEM.Setup}(1^\kappa)$: This algorithm runs the $\mathsf{NIKEcommon\_setup}(1^\kappa)$ algorithm to obtain the system parameters *params*. It then sets the public parameters of the KEM *pub* as *params*.
2. $\mathsf{KEM.Gen}(1^\kappa, pub)$: This algorithm runs the algorithm $\mathsf{NIKEgen}(1^\kappa, pub)$ to obtain a pair of public-private key (pk, sk). It then sets $pk_{\text{KEM}} = pk$ and $sk_{\text{KEM}} = sk$.
3. $\mathsf{KEM.Encap}(pk_{\text{KEM}})$: This algorithm parses pk_{KEM} as pk, samples another key pair $(pk', sk') \leftarrow \mathsf{NIKEgen}(1^\kappa, pub)$. Then it runs $\mathsf{NIKEkey}(pk, sk')$ to produce the shared key shk. It then sets the encapsulated key $K = shk$ and the ciphertext $C = pk'$.
4. $\mathsf{KEM.Decap}(sk_{\text{KEM}}, C)$: This algorithm parses the ciphertext C as pk' and the secret key sk_{KEM} as sk. It then runs $\mathsf{NIKEkey}(pk', sk)$ and obtains the result.

Figure 1. Construction of LR-CCA-2 secure KEM Γ

Theorem 2. *Suppose the leakage-resilient NIKE scheme* LR-NIKE *is* BLR-CKS-heavy*-secure with leakage rate* $1 - o(1)$*. Then the KEM scheme* Γ *is* BLR-IND-CCA-2*-secure KEM. More formally, let* $\mathcal{A}_{\text{KEM}}$ *be an adversary against* Γ *making* q_D *number of decapsulation queries and* q_L *number of leakage queries. Then using* $\mathcal{A}_{\text{KEM}}$*, we can construct another adversary* $\mathcal{A}_{\text{NIKE}}$ *in the* BLR-CKS-heavy *security model who makes two* RegisterHonest *queries,* q_D *number of* RegisterCorrupt *queries,* q_D *number of* CorruptReveal *queries, and* q_L *number of* Leakage *queries and the running time of* $\mathcal{A}_{\text{NIKE}}$ *is roughly same as that* $\mathcal{A}_{\text{KEM}}$*. Moreover the leakage rate of* Γ *is* $1 - o(1)$*.*

Proof. Let $\mathcal{A}_{\text{KEM}}$ be an adversary against the BLR-IND-CCA-2-secure KEM Γ. We now show how to use $\mathcal{A}_{\text{KEM}}$ to construct another adversary $\mathcal{A}_{\text{NIKE}}$ against LR-NIKE, thereby contradicting its BLR-CKS-heavy security. $\mathcal{A}_{\text{NIKE}}$ simulates the environment to $\mathcal{A}_{\text{KEM}}$ in the following way:

- KEM.Setup: On input the public parameters *params*, $\mathcal{A}_{\text{NIKE}}$ sets the public parameters *pub* of the KEM scheme as *params*.
- KEM.Gen: $\mathcal{A}_{\text{NIKE}}$ makes *two* RegisterHonest queries, receiving as input two honestly registered public keys pk_1 and pk_2. It then sets $pk_{\text{KEM}} = pk_1$ and $sk_{\text{KEM}} = \perp$.
- $\mathsf{KEM.Encap}(pk)$: To simulate the challenge phase, $\mathcal{A}_{\text{NIKE}}$ makes a $\mathsf{Test}(pk_1, pk_2)$ query. It receives as reply a shared key K which is either the real key, i.e., $K = \mathsf{NIKE.key}(pk_1, sk_2)$, or a random shared $K \leftarrow \mathcal{SHK}$. It then sets the encapsulated key $K^* = K$ and $C^* = pk_2$.
- Leakage queries: When $\mathcal{A}_{\text{KEM}}$ queries with leakage functions f, the challenger $\mathcal{A}_{\text{NIKE}}$ forwards f to the leakage oracle $O^\lambda_{sk_1}(\cdot)$, and receives as response $f(sk_1)$. It then returns $f(sk_1)$ to $\mathcal{A}_{\text{KEM}}$.

– Decapsulation queries: $\mathcal{A}_{\text{KEM}}$ makes decapsulation queries to $\mathcal{A}_{\text{NIKE}}$ with ciphertexts C. $\mathcal{A}_{\text{NIKE}}$ parses C as pk', and since we have $C \neq C^*$, we have $pk' \neq pk_2$. If $pk' = pk_1$, $\mathcal{A}_{\text{NIKE}}$ outputs $\perp$. This is consistent with the rejection rule of the KEM Γ and also LR-NIKE. Else, $\mathcal{A}_{\text{NIKE}}$ makes a RegisterCorrupt query on pk'. Here we assume that all of $\mathcal{A}_{\text{KEM}}$'s decapsulation queries are distinct without loss of generality and hence all of the RegisterHonest queries are distinct. $\mathcal{A}_{\text{NIKE}}$ then makes a CorruptReveal(pk_1, pk') query to get a shared key $K \in \mathcal{SHK}$ or a symbol $\perp$. It then returns K to $\mathcal{A}_{\text{KEM}}$.

This completes the description of $\mathcal{A}_{\text{NIKE}}$'s simulation. From the description it is clear than the above simulation is perfect. Note that, if K^* is the real shared key, i.e., it is the output of NIKE.key algorithm, then it is properly simulating the Encap algorithm in the $\mathbf{Exp}^{\text{BLR-IND-CCA-2}}_{\Gamma,\mathsf{A}}(\kappa, \lambda)$ security experiment; else if K^* is chosen randomly it also properly simulates the fact that it is chosen randomly in the experiment. Finally, when $\mathcal{A}_{\text{KEM}}$ outputs a bit b' as its guess for b in the experiment, $\mathcal{A}_{\text{NIKE}}$ also outputs the same bit b'. So the advantage of $\mathcal{A}_{\text{NIKE}}$ in breaking the BLR-CKS-heavy security of LR-NIKE is exactly the *same* as the advantage of $\mathcal{A}_{\text{KEM}}$ in breaking the BLR-IND-CCA-2 security of the KEM scheme Γ. Also note that the number of RegisterCorrupt and CorruptReveal queries made by $\mathcal{A}_{\text{NIKE}}$ is same as the number of decapsulation queries asked by $\mathcal{A}_{\text{KEM}}$. This completes the proof of the above theorem. □

5.2 Leakage-Resilient Authenticated Key Exchange

The work of Bergsma et al. [7] shows a generic construction of eCK-secure AKE protocol, using an UF-CMA-secure signature scheme, CKS-light-secure NIKE scheme and a pseudo-random function as underlying primitives.

In this paper, we present a construction of a leakage-resilient NIKE protocol, which is secure in the CKS-heavy model, under bounded-memory leakage, i.e. BLR-CKS-heavy-secure NIKE protocol (Table 2). Since the CKS-heavy-security implies the CKS-light-security, our leakage-resilient NIKE protocol can work as a bounded-memory-leakage-resilient CKS-light-secure NIKE protocol. Further, in the literature we can find UF-CMLA-secure signature schemes [24], which are UF-CMA-secure signature schemes under the bounded-memory leakage model. Thus, we have the necessary primitives to transform our leakage-resilient NIKE to a leakage-resilient AKE, following the NIKE to AKE transformation of Bergsma et al [7], in the bounded-memory leakage model.

There are numerous leakage versions of the eCK model, under OCLI axiom [3–5], and the memory leakage model [9]. Further, they address after-the-fact leakage. With our new BLR-CKS-heavy-secure NIKE protocol, following the Bergsma et al. [7] transformation, we can achieve leakage-resilient AKE in an eCK-style model,

- under the memory leakage (stronger than the OCLI axiom),
- addressing before-the-fact leakage (weaker than the after-the-fact leakage).

Bounded-memory Before-the-fact Leakage eCK Model. We present a suitable security model to analyze the leakage resiliency of AKE protocols, considering the aforementioned points, i.e. in an eCK-style security model [26] addressing before-the-fact, bounded-memory leakage.

Let κ be the security parameter. Let $\mathcal{U} = \{U_1, ..., U_n\}$ be a set of n parties. We use the term *principal* to identify a party involved in a protocol instance. Each party U_i where $i \in [1, N_P]$ has a pair of long-term public and secret-keys, (pk_{U_i}, sk_{U_i}). The term *session* is used to identify a protocol instance at a principal. Each principal may have multiple sessions and they may run concurrently. The oracle $\Pi^s_{U,V}$ represents the s^{th} session at the owner principal U, with intended partner principal V. The principal which sends the first protocol message of a session is the *initiator* of the session, and the principal which responds to the first protocol message is the *responder* of the session.

Partner sessions in BBFL-eCK model. Two oracles $\Pi^s_{U,V}$ and $\Pi^{s'}_{U',V'}$ are said to be partners, if all of the following conditions hold:

1. both $\Pi^s_{U,V}$ and $\Pi^{s'}_{U',V'}$ have computed session keys;
2. messages sent from $\Pi^s_{U,V}$ and messages received by $\Pi^{s'}_{U',V'}$ are identical;
3. messages sent from $\Pi^{s'}_{U',V'}$ and messages received by $\Pi^s_{U,V}$ are identical;
4. $U' = V$ and $V' = U$;
5. Exactly one of U and V is the initiator and the other is the responder.

The protocol is said to be *correct* if two partner oracles compute identical session keys.

Modelling Leakage. We consider the bounded-memory leakage setting for modelling the leakage As before, the adversary is allowed to issue arbitrary efficiently computable leakage functions f_i and obtain the leakage $f_i(sk)$ of the secret key sk, before the session key is established. As mentioned above the constraint is $\sum_{i=1} |f_i(sk)| \leq \lambda$, where λ is the leakage parameter.

Adversarial Powers.

- $\mathsf{Send}(U, V, s, m)$ query: The oracle $\Pi^s_{U,V}$, computes the next protocol message according to the protocol specification and sends it to the adversary. $\mathcal{A}$ can also use this query to activate a new protocol instance with blank m.
- $\mathsf{SessionKeyReveal}(U, V, s)$ query: $\mathcal{A}$ is given the session key of the oracle $\Pi^s_{U,V}$.
- $\mathsf{EphemeralKeyReveal}(U, V, s)$ query: $\mathcal{A}$ is given the ephemeral keys (per-session randomness) of the oracle $\Pi^s_{U,V}$.
- $\mathsf{Corrupt}(U)$ query: $\mathcal{A}$ is given the long-term secrets of the principal U.
- $\mathsf{Test}(U, s)$ query: When $\mathcal{A}$ asks the `Test` query, the challenger first chooses a random bit $b \xleftarrow{\$} \{0, 1\}$ and if $b = 1$ then the actual session key is returned to $\mathcal{A}$, otherwise a random string chosen from the same session key space is returned to $\mathcal{A}$.
- $\mathsf{Leakage}(U, f_i)$ query: The leakage $f_i(sk_U)$ is computed and returns to the adversary if and only if $\sum_{i=1} |f_i(sk_U)| \leq \lambda$.

λ-BBFL-eCK-freshness. Let λ be the leakage parameter. An oracle $\Pi^s_{U,V}$ is said to be λ-*BBFL-eCK*-fresh if and only if the conditions (1)-(3) of Alawatugoda et al. [4, Def.4] hold, and

4. Before $\Pi^s_{U,V}$ is activated, for all $\mathsf{Leakage}(U, f_i)$ queries, $\sum_{i=1} |f_i(sk_U)| \leq \lambda$, and for all $\mathsf{Leakage}(V, f_i)$ queries, $\sum_{i=1} |f_i(sk_V)| \leq \lambda$.
5. After $\Pi^s_{U,V}$ is activated no leakage allowed from U and V.

The *BBFL-eCK* security game. The adversary $\mathcal{A}$ interacts with the challenger by issuing any combination of $\mathsf{Send}()$, $\mathsf{SessionKeyReveal}()$, $\mathsf{EphemeralKeyReveal}()$, $\mathsf{Leakage}()$ and $\mathsf{Corrupt}()$ queries at will. At some point the adversary chooses a λ-BBFL-eCK-fresh oracle and issues a $\mathsf{Test}()$ query. Then, the adversary may continue asking the $\mathsf{Send}()$, $\mathsf{SessionKeyReveal}()$, $\mathsf{EphemeralKeyReveal}()$, $\mathsf{Leakage}()$ and $\mathsf{Corrupt}()$ queries while preserving the freshness of the test session, and finally outputs answer bit b' for the challenge. $\mathcal{A}$ wins if $b' = b$. Let $Succ_{\mathcal{A}}$ is the event that the adversary $\mathcal{A}$ wins the above security game.

Definition 6 (BBFL-eCK-security). *A protocol π is said to be* BBFL-eCK*-secure if there is no PPT adversary $\mathcal{A}$ that can win the* BBFL-eCK *security game with non-negligible advantage. The advantage of an adversary $\mathcal{A}$ is defined as* $\mathsf{Adv}^{\text{BBFL-eCK}}_{\pi,\mathcal{A}}(\kappa) = |2\Pr(Succ_{\mathcal{A}}) - 1|$.

Constructing BBFL-eCK-secure Key Exchange Protocols. In Table 4, we show the generic leakage-resilient variant of the Bergsma et al. [7] AKE protocol. We replace the CKS-light-secure NIKE with a BLR-CKS-heavy-secure NIKE, and the UF-CMA-secure signature scheme with a UF-CMLA-secure signature scheme in the bounded-memory leakage model, to come up with the generic BBFL-eCK-secure AKE protocol. In this protocol, the final shared key is obtained by xor-ing the intermediate keys. Since the adversary learns the leakage only from the long-term secret parameters, it is not necessary to use leakage-resilient PRFs for the construction of LR-AKE, following NIKE to AKE transformation of Bergsma et al.

Let $\mathsf{LR\text{-}NIKE} = (\mathsf{NIKEcommon\_setup}, \mathsf{NIKEgen}, \mathsf{NIKEkey})$ be the underlying BLR-CKS-heavy-secure NIKE protocol, $\mathsf{LR\text{-}SIG} = (\mathsf{SIGkg}, \mathsf{SIGsign}, \mathsf{SIGvfy})$ be the underlying UF-CMLA-secure signature scheme and PRF be a secure pseudo-random function. Since, the generic construction of the AKE protocol remains unchanged with respect to the Bergsma et al. [7], except the replacement of the leakage-resilient advancements of the underlying primitives, in the bounded-memory leakage setting, the security of the resulting AKE still preserves the eCK-style with the advancements of leakage resiliency in the bounded-memory leakage setting. Therefore, the security theorem and the flow of the security proof is similar to Theorem 1 and its proof in [7, Appendix A].

A (Initiator) $\big((sk_A^{\text{nike}}, sk_A^{\text{sig}}), (pk_A^{\text{nike}}, pk_A^{\text{sig}})\big)$		B (Responder) $\big((sk_B^{\text{nike}}, sk_B^{\text{sig}}), (pk_B^{\text{nike}}, pk_B^{\text{sig}})\big)$
$r_A \xleftarrow{\$} \{0.1\}^\kappa$		$r_B \xleftarrow{\$} \{0.1\}^\kappa$
$(sk_A^{\text{tmp}}, pk_A^{\text{tmp}}) \leftarrow \mathsf{NIKEgen}(1^\kappa, r_A)$		$(sk_B^{\text{tmp}}, pk_B^{\text{tmp}}) \leftarrow \mathsf{NIKEgen}(1^\kappa, r_B)$
$\sigma_A \leftarrow \mathsf{SIGsign}(sk_A^{\text{sig}}, pk_A^{\text{tmp}})$	$\xrightarrow{(pk_A^{\text{tmp}}, \sigma_A)}$	$\sigma_B \leftarrow \mathsf{SIGsign}(sk_B^{\text{sig}}, pk_B^{\text{tmp}})$
If: $\mathsf{SIGvfy}(pk_B^{\text{sig}}, pk_B^{\text{tmp}}, \sigma_B) = 1$;	$\xleftarrow{(pk_B^{\text{tmp}}, \sigma_B)}$	If: $\mathsf{SIGvfy}(pk_A^{\text{sig}}, pk_A^{\text{tmp}}, \sigma_A) = 1$;
$T := \mathsf{sort}(pk_A^{\text{tmp}}, pk_B^{\text{tmp}})$		$T := \mathsf{sort}(pk_A^{\text{tmp}}, pk_B^{\text{tmp}})$
$k_{\text{nike,nike}} = \mathsf{PRF}(\mathsf{NIKEkey}(sk_A^{\text{nike}}, pk_B^{\text{nike}}), T)$		$k_{\text{nike,nike}} = \mathsf{PRF}(\mathsf{NIKEkey}(sk_B^{\text{nike}}, pk_A^{\text{nike}}), T)$
$k_{\text{nike,tmp}} = \mathsf{PRF}(\mathsf{NIKEkey}(sk_A^{\text{nike}}, pk_B^{\text{tmp}}), T)$		$k_{\text{nike,tmp}} = \mathsf{PRF}(\mathsf{NIKEkey}(sk_B^{\text{nike}}, pk_A^{\text{tmp}}), T)$
$k_{\text{tmp,nike}} = \mathsf{PRF}(\mathsf{NIKEkey}(sk_A^{\text{tmp}}, pk_B^{\text{nike}}), T)$		$k_{\text{tmp,nike}} = \mathsf{PRF}(\mathsf{NIKEkey}(sk_B^{\text{tmp}}, pk_A^{\text{nike}}), T)$
$k_{\text{tmp,tmp}} = \mathsf{PRF}(\mathsf{NIKEkey}(sk_A^{\text{tmp}}, pk_B^{\text{tmp}}))$		$k_{\text{tmp,tmp}} = \mathsf{PRF}(\mathsf{NIKEkey}(sk_B^{\text{tmp}}, pk_A^{\text{tmp}}))$
$k_{A,B} := k_{\text{nike,nike}} \oplus k_{\text{nike,tmp}} \oplus k_{\text{tmp,nike}} \oplus k_{\text{tmp,tmp}}$		$k_{B,A} := k_{\text{nike,nike}} \oplus k_{\text{nike,tmp}} \oplus k_{\text{tmp,nike}} \oplus k_{\text{tmp,tmp}}$

Table 4. Leakage-resilient AKE Protocol LR-AKE

Theorem 3. *If the underlying NIKE protocol,* LR-NIKE*, is* BLR-CKS-heavy*-secure, the signature scheme,* LR-SIG*, is* UF-CMLA*-secure in the bounded-memory leakage model, and the pseudo-random property holds for the* PRF*, then the* LR-AKE *protocol is* BBFL-eCK*-secure.*
Let d be the number of parties. Each party U_i owns at most ℓ number of protocol sessions. Let $\mathcal{A}$ be an adversary against the above protocol LR-AKE*. We construct attackers $\mathcal{B}_{\mathsf{sig}}, \mathcal{B}^{(1)}_{\mathsf{nike}}, \mathcal{B}^{(0)}_{\mathsf{nike}}$ and $\mathcal{B}_{\mathsf{prf}}$ against the underlying leakage-resilient signature scheme, leakage-resilient NIKE protocol (matching session exists and no matching session exists respectively), and the pseudo-random function s.t.,*

$$\begin{aligned}\mathsf{Adv}^{\text{BBFL-eCK}}_{\text{LR-AKE},\mathcal{A}}(\kappa) \leq 4 \cdot d^2\ell^2 \cdot \Big(\mathsf{Adv}^{\text{BLR-CKS-heavy}}_{\text{LR-NIKE},\mathcal{B}^{(1)}_{\text{nike}}}(\kappa) + \mathsf{Adv}^{\text{prf}}_{\mathsf{PRF},\mathcal{B}_{\mathsf{prf}}}(\kappa)\Big)\\ +4 \cdot \mathsf{Adv}^{\text{BLR-CKS-heavy}}_{\text{LR-NIKE},\mathcal{B}^{(0)}_{\text{nike}}}(\kappa) + 4 \cdot d \cdot \mathsf{Adv}^{\text{UF-CMLA}}_{\text{LR-SIG},\mathcal{B}_{\mathsf{sig}}}(\kappa)\ .\end{aligned}$$

Proof Sketch. To prove the Theorem 3, we need to consider four types of attackers.

- A1-type attacker never asks EphemeralKeyReveal query for the test session. If there exists a partner to the test session it will also never asks EphemeralKeyReveal query for the partner session.
- A2-type attacker never asks EphemeralKeyReveal query for the test session. If there exists a partner to the test session it also never asks Corrupt query for the owner of the partner session.
- A3-type attacker never asks Corrupt query to the owner of the test session. If there exists a partner to the test session it also never asks the EphemeralKeyReveal query to the partner session.
- A4-type attacker never asks Corrupt query to the owner of the test session. If there exists a partner to the test session it also never asks the Corrupt query to the owner of the partner session.

Each legitimate attacker according to the freshness definition falls into at least on of these categories.

In the LR-AKE protocol the session key is computed as $k := k_{\text{nike,nike}} \oplus k_{\text{nike,tmp}} \oplus k_{\text{tmp,nike}} \oplus k_{\text{tmp,tmp}}$. The main intuition behind this construction is that we need to reduce the indistinguishability of the shared key of LR-AKE to the indistinguishability of LR-NIKE. In this simulation we can easily simulate the leakage, by giving the adversary $\mathcal{A}$ the leakage obtained from the underlying leakage-resilient NIKE-challenger and the signature scheme challenger. In the security experiment against the leakage-resilient NIKE, the NIKE-adversary gets two challenge public-keys from the leakage-resilient NIKE-challenger. In the reduction, we need to embed them into the view of the adversary $\mathcal{A}$, in a way that we can embed the leakage-resilient NIKE-challenge key into k while successfully answering all the legitimate Corrupt and EphemeralKeyReveal queries.

A1-type attacker never asks EphemeralKeyReveal queries to the test session and the partner to the test session. Thus, it is possible to embed the public keys from the leakage-resilient NIKE-challenger, as the

ephemeral public keys of the test session. Then, use the challenge key from the leakage-resilient NIKE challenger as $k_{\text{tmp,tmp}}$.

For the case of A2-type attacker, embed the public keys from the leakage-resilient NIKE-challenger, one as the ephemeral public key and the other one as the long-term public-key of the test session. Then, use the challenge key as $k_{\text{tmp,nike}}$. Since this embedding involves a long-term secret of one party of the test session, we need to use an additional PRF, and this long-term secret is used in many protocol executions involving the corresponding party. Similarly, A3 and A4-type attackers can be handled by embedding the leakage-resilient NIKE-challenger's public and challenge keys accordingly.

Thus, the four attackers correspond to all possible combinations of Corrupt and EphemeralKeyReveal queries, that are allowed in our BBFL-eCK security model. □

Leakage Tolerance of the Generic LR-AKE Protocol. This generic protocol can tolerate the leakage according to the leakage tolerance of the underlying leakage-resilient NIKE and the leakage-resilient signature scheme. Our LR−NIKE can tolerate $1 - o(1)$ leakage and the UF-CMLA signature scheme of Katz et al. [24] can tolerate $n - n^{\epsilon}$ leakage, for n bit key and $1 > \epsilon > 0$, which approaches $1 - o(1)$ leakage rate for sufficiently large n and small enough ϵ. Thus, the corresponding instantiation can tolerate an overall leakage rate of $1 - o(1)$.

5.3 Leakage-resilient Low-latency Key Exchange

Low-latency key exchange (LLKE) can be considered as one of the important practical usages of NIKE protocols, which permits the transmission of cryptographically protected data, without prior key exchange, while providing perfect forward secrecy (PFS). Leakage resiliency of LLKE remains unstudied.

Bounded-memory Leakage LLKE-ma Model. We refer the security model under mutual authentication of Hale et al. [21, Section 5] as LLKE-ma model. In this work, we introduce bounded-memory leakage model on top of LLKE-ma model (We use the notation BL-LLKE-ma to identify our model whenever necessary). Let d be the number of clients and ℓ be the number of servers. Each client is represented by a collection of n oracles $\mathsf{C}_{\mathsf{i},1}, \ldots, \mathsf{C}_{\mathsf{i},\mathsf{n}}$ and each server is represented by a collection of k oracles $\mathsf{S}_{\mathsf{j},1}, \ldots, \mathsf{S}_{\mathsf{j},\mathsf{k}}$. Each oracle represents an instance of the protocol. Each principal has a long-term key pair $(sk_{\mathsf{i}}, pk_{\mathsf{i}})$. Let κ be the security parameter and λ be the leakage parameter. Each oracle $\mathsf{C}_{\mathsf{i},\mathsf{s}} \in [d] \times [n]$ (or $\mathsf{S}_{\mathsf{j},\mathsf{t}} \in [\ell] \times [k]$, respectively), maintains:

1. two variables $\mathsf{k}_{\mathsf{i}}^{\mathsf{tmp}}$ and $\mathsf{k}_{\mathsf{i}}^{\mathsf{main}}$ to store the temporary and main keys of a session,
2. a variable $\mathsf{Partner}_{\mathsf{i}}$ contains the identity of the intended communication partner, and
3. variables $\mathcal{M}_{\mathsf{i},\mathsf{s}}^{in}$ and $\mathcal{M}_{\mathsf{i},\mathsf{s}}^{out}$ containing messages sent and received by the oracle.

Adversarial Powers:

- $\mathsf{Send}(\mathsf{C}_{\mathsf{i},\mathsf{s}}/\mathsf{S}_{\mathsf{j},\mathsf{t}}, \mathsf{m})$: The adversary sends the message m to the requested oracle, the oracle processes m according to the protocol specification, and the response is returned to the adversary.
- $\mathsf{Reveal}(\mathsf{C}_{\mathsf{i},\mathsf{s}}/\mathsf{S}_{\mathsf{j},\mathsf{t}}, \mathsf{tmp/main})$: This query returns the key of the given stage if it has been already computed, or $\perp$ otherwise.
- $\mathsf{Corrupt}(\mathsf{i/j})$: This query returns the long-term secret key of the server or the client accordingly. If $\mathsf{Corrupt}(\mathsf{j/i})$ is the τ-th query issued by the adversary, we say a party is τ-corrupted. For the parties that are not corrupted we define $\tau := \infty$.
- $\mathsf{Test}(\mathsf{C}_{\mathsf{i},\mathsf{s}}/\mathsf{S}_{\mathsf{j},\mathsf{t}}, \mathsf{tmp/main})$: This query is used to test a key. If the variable for the requested key is not empty, the challenger chooses $b \xleftarrow{\$} \{0,1\}$, and if $b = 0$ then the requested key is returned, else a random key is returned. Otherwise, $\perp$ is returned.
- $\mathsf{Leakage}(\mathsf{i/j}, f_i)$: The leakage $f_i(sk_{\mathsf{i/j}})$ is computed and returns to the adversary if and only if $\sum_{i=1} |f_i(sk_{\mathsf{i/j}})| \leq \lambda$.

BL-LLKE-*ma* security game: The adversary interacts with the challenger by issuing any combination of $\mathsf{Send}(), \mathsf{Corrupt}(), \mathsf{Reveal}(), \mathsf{Leak}()$ queries. At some point the adversary issues a $\mathsf{Test}()$ query, to an oracle that holds the conditions in Def. 7. Then, the adversary may continue asking the $\mathsf{Send}(), \mathsf{Corrupt}(), \mathsf{Reveal}(), \mathsf{Leak}()$ queries, without violating the conditions of the Def. 7, and finally outputs answer bit b' for the challenge. $\mathcal{A}$ wins if $b' = b$. Let $Succ_{\mathcal{A}}$ is the event that the adversary $\mathcal{A}$ wins the above security game.

Definition 7 (Leakage-Resilient Key-security (under Mutual Authentication)). *A protocol π is said to be* BL-LLKE-ma*-secure if there is no PPT adversary $\mathcal{A}$ that can win the* BL-LLKE-ma *security game with non-negligible advantage, while holding the following conditions:*

- *All the conditions* [21, Def. 8].
- *Before activation of the test session on* $\mathsf{C_i}$, $\forall$ $\mathsf{Leakage}(\mathsf{i}, f_i)$ *queries,* $\sum_{i=1} |f_i(sk_\mathsf{i})| \leq \lambda$, *and before activation of the test session on* $\mathsf{S_j}$ $\forall$ $\mathsf{Leakage}(\mathsf{j}, f_i)$ *queries,* $\sum_{i=1} |f_i(sk_\mathsf{j})| \leq \lambda$.
- *After activation of the Test session on* $\mathsf{C_i}$ *no leakage allowed from* sk_i *(same as to the case of* $\mathsf{S_j}$*).*

The advantage of $\mathcal{A}$ is defined as $\mathsf{Adv}^{\text{BL-LLKE-ma}}_{\pi,\mathcal{A}}(\kappa) = |2\Pr(Succ_\mathcal{A}) - 1|$.

Generic Construction of BL-LLKE-ma-secure LLKE from NIKE. In the work of Hale et al., they have used a CKS-light-secure NIKE scheme NIKE and UF-CMA-secure signature scheme SIG. We simply replace those primitives with their respective leakage-resilient versions.

Let $\mathsf{LR\text{-}NIKE} = (\mathsf{NIKEcommon\_setup}, \mathsf{NIKEgen}, \mathsf{NIKEkey})$ be a BLR-CKS-heavy-secure NIKE scheme, $\mathsf{LR\text{-}SIG} = (\mathsf{SIGkg}, \mathsf{SIGsign}, \mathsf{SIGvfy})$ be an UF-CMLA-secure signature scheme. Then we construct a LLKE protocol $\mathsf{LR\text{-}LLKE}$=$(\mathsf{Gen}, \mathsf{KE}^{\mathsf{client}}_{\mathsf{init}}, \mathsf{KE}^{\mathsf{client}}_{\mathsf{refresh}}, \mathsf{KE}^{\mathsf{server}}_{\mathsf{refresh}})$ same as the description in Hale et al. [21, Section 6.1].

Since, the generic construction of the LLKE protocol remains unchanged with respect to the Hale et al., except the replacement of the leakage-resilient advancements of the underlying primitives (in the bounded-memory leakage model), the security of the resulting AKE still preserves the LLKE-ma-style with the advancements of leakage resiliency in the bounded-memory leakage model. Therefore, the security theorem and the flow of the security proof is similar to the theorem and proof of the Theorem 2 of Hale et al. [21, Appendix 6.2].

Theorem 4. *If the underlying NIKE protocol,* $\mathsf{LR\text{-}NIKE}$*, is* BLR-CKS-heavy*-secure, the signature scheme,* $\mathsf{LR\text{-}SIG}$*, is* UF-CMLA*-secure in the bounded-memory leakage model, then the* $\mathsf{LR\text{-}LLKE}$ *protocol is* BK-LLKE-ma*-secure.*

Let d be the number of clients and ℓ be the number of servers. Each client and each server is represented by a collection of n and k oracles respectively. Let $\mathcal{A}$ be an adversary against the above protocol $\mathsf{LR\text{-}LLKE}$*. We construct attackers $\mathcal{B}_{\mathsf{sig}}$ and $\mathcal{B}_{\mathsf{nike}}$ against the underlying leakage-resilient signature scheme and the leakage-resilient NIKE protocol such that,*

$$\begin{aligned}\mathsf{Adv}^{\text{BK-LLKE-ma}}_{\mathsf{LR\text{-}LLKE},\mathcal{A}}(\kappa) \leq d\ell n \cdot \left(\mathsf{Adv}^{\text{BLR-CKS-heavy}}_{\mathsf{LR\text{-}NIKE},\mathcal{B}_{\mathsf{nike}}}(\kappa) + 2\mathsf{Adv}^{\text{UF-CMLA}}_{\mathsf{LR\text{-}SIG},\mathcal{B}_{\mathsf{sig}}}(\kappa)\right)\\ + d\ell n \cdot \left(k \cdot \mathsf{Adv}^{\text{BLR-CKS-heavy}}_{\mathsf{LR\text{-}NIKE},\mathcal{B}_{\mathsf{nike}}}(\kappa) + 2\mathsf{Adv}^{\text{UF-CMLA}}_{\mathsf{LR\text{-}SIG},\mathcal{B}_{\mathsf{sig}}}(\kappa)\right)\\ + 2kd\ell n \cdot \left(\mathsf{Adv}^{\text{BLR-CKS-heavy}}_{\mathsf{LR\text{-}NIKE},\mathcal{B}_{\mathsf{nike}}}(\kappa) + 2\mathsf{Adv}^{\text{UF-CMLA}}_{\mathsf{LR\text{-}SIG},\mathcal{B}_{\mathsf{sig}}}(\kappa)\right) + 4 \cdot \mathsf{Adv}^{\text{BLR-CKS-heavy}}_{\mathsf{LR\text{-}NIKE},\mathcal{B}_{\mathsf{nike}}}(\kappa)\ .\end{aligned}$$

Proof Sketch. We distinguish between four different attackers:

- A1-type attacker asks Test query to a client oracle and the temporary key.
- A2-type attacker asks Test query to a client oracle and the main key.
- A3-type attacker asks Test query to a server oracle and the temporary key.
- A4-type attacker asks Test query to a client oracle and the main key.

The four different attackers correspond to all possible combinations of queries, that are allowed in our BK-LLKE-ma security model. The four distinct lines of the equation in Theorem 4 corresponds to each of above cases respectively. We can easily simulate the leakage, by giving the adversary $\mathcal{A}$ the leakage obtained from the underlying leakage-resilient NIKE-challenger and the signature scheme challenger. Apart from that, the simulation is same as to the Hale et al [21]. □

Leakage Tolerance of the Generic LR-LLKE Protocol. This generic protocol can tolerate the leakage according to the leakage tolerance of the underlying leakage-resilient NIKE and the leakage-resilient signature scheme. Our $\mathsf{LR\text{-}NIKE}$ can tolerate $1 - o(1)$ leakage and the UF-CMLA signature scheme of Katz et al. [24] can tolerate $n - n^\epsilon$ leakage, for n bit key and $1 > \epsilon > 0$, which approaches $1 - o(1)$ leakage rate for sufficiently large n and small enough ϵ. Thus, the corresponding instantiation can tolerate an overall leakage rate of $1 - o(1)$.

6 Conclusion and Future Works

In this paper, we present a new approach to construct several leakage-resilient cryptographic primitives, such as leakage-resilient PKE schemes, AKE protocols and LLKE protocols, based on leakage-resilient NIKE protocols as the main building block. Our construction of LR-NIKE in the bounded leakage setting achieves an optimal leakage rate, i.e., $1 - o(1)$, and the resulting leakage-resilient constructions from that also preserve the same leakage-rate, upon the appropriate choice of parameters. Our work opens up leakage-resilient LLKE protocols, and we hope there is much work to be done on this. We leave open the following main problems:

- Efficient construction of leakage-resilient NIKE in the $1 - o(1)$-bounded-memory leakage model, *without* leak-free hardware assumption.
- Leakage-resilient NIKE in the $1 - o(1)$-continuous-memory leakage model.
- Leakage-resilient NIKE in after-the-fact leakage model.
- On leakage-resilient LLKE, as this paper opens up that direction, we hope there is much work to be done in future.

References

1. Shweta Agrawal, Yevgeniy Dodis, Vinod Vaikuntanathan, and Daniel Wichs. On continual leakage of discrete log representations. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 401–420. Springer, 2013.
2. Adi Akavia, Shafi Goldwasser, and Vinod Vaikuntanathan. Simultaneous hardcore bits and cryptography against memory attacks. In *Theory of Cryptology Conference*, pages 474–495, 2009.
3. Janaka Alawatugoda, Colin Boyd, and Douglas Stebila. Continuous after-the-fact leakage-resilient key exchange. In *Information Security and Privacy - 19th Australasian Conference, ACISP 2014, Wollongong, NSW, Australia, July 7-9, 2014. Proceedings*, pages 258–273, 2014.
4. Janaka Alawatugoda, Douglas Stebila, and Colin Boyd. Modelling after-the-fact leakage for key exchange. In *Proceedings of the 9th ACM symposium on Information, computer and communications security*, pages 207–216. ACM, 2014.
5. Janaka Alawatugoda, Douglas Stebila, and Colin Boyd. Continuous after-the-fact leakage-resilient eck-secure key exchange. In *Cryptography and Coding - 15th IMA International Conference, IMACC 2015, Oxford, UK, December 15-17, 2015. Proceedings*, pages 277–294, 2015.
6. Joël Alwen, Yevgeniy Dodis, and Daniel Wichs. Leakage-resilient public-key cryptography in the bounded-retrieval model. In *Advances in Cryptology-CRYPTO 2009*, pages 36–54. Springer, 2009.
7. Florian Bergsma, Tibor Jager, and Jörg Schwenk. One-round key exchange with strong security: An efficient and generic construction in the standard model. In *IACR International Workshop on Public Key Cryptography*, pages 477–494. Springer, 2015.
8. Zvika Brakerski, Yael Tauman Kalai, Jonathan Katz, and Vinod Vaikuntanathan. Overcoming the hole in the bucket: Public-key cryptography resilient to continual memory leakage. *IACR Cryptology ePrint Archive*, Report 2010/278, 2010.
9. Rongmao Chen, Yi Mu, Guomin Yang, Willy Susilo, and Fuchun Guo. Strongly leakage-resilient authenticated key exchange. In *Topics in Cryptology - CT-RSA 2016 - The Cryptographers' Track at the RSA Conference 2016, San Francisco, CA, USA, February 29 - March 4, 2016, Proceedings*, pages 19–36, 2016.
10. W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, pages 644 – 654, 1976.
11. Yevgeniy Dodis, Kristiyan Haralambiev, Adriana López-Alt, and Daniel Wichs. Cryptography against continuous memory attacks. In *Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on*, pages 511–520. IEEE, 2010.
12. Yevgeniy Dodis, Kristiyan Haralambiev, Adriana López-Alt, and Daniel Wichs. Efficient public-key cryptography in the presence of key leakage. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 613–631. Springer, 2010.
13. Yevgeniy Dodis, Yael Tauman Kalai, and Shachar Lovett. On cryptography with auxiliary input. In *STOC*, pages 621–630, 2009.
14. Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM journal on computing*, 38(1):97–139, 2008.
15. Stefan Dziembowski and Sebastian Faust. Leakage-resilient circuits without computational assumptions. In *Theory of Cryptography Conference*, pages 230–247. Springer, 2012.

16. Eduarda SV Freire, Dennis Hofheinz, Eike Kiltz, and Kenneth G Paterson. Non-interactive key exchange. In *Public-Key Cryptography–PKC 2013*, pages 254–271. Springer, 2013.
17. Atsushi Fujioka, Koutarou Suzuki, Keita Xagawa, and Kazuki Yoneyama. Strongly secure authenticated key exchange from factoring, codes, and lattices. *Designs, Codes and Cryptography*, 76(3):469–504, 2015.
18. David Galindo. Boneh-franklin identity based encryption revisited. In *International Colloquium on Automata, Languages, and Programming*, pages 791–802. Springer, 2005.
19. David Galindo, Johann Großschädl, Zhe Liu, Praveen Kumar Vadnala, and Srinivas Vivek. Implementation of a leakage-resilient elgamal key encapsulation mechanism. *Journal of Cryptographic Engineering*, 6(3):229–238, 2016.
20. Shafi Goldwasser and Guy N Rothblum. Securing computation against continuous leakage. In *CRYPTO*, volume 6223, pages 59–79. Springer, 2010.
21. Britta Hale, Tibor Jager, Sebastian Lauer, and Jörg Schwenk. Speeding: On low-latency key exchange. *IACR Cryptology ePrint Archive*, 2015:1214, 2015.
22. Shai Halevi and Huijia Lin. After-the-fact leakage in public-key encryption. In *Theory of Cryptography Conference*, pages 107–124. Springer, 2011.
23. Ali Juma and Yevgeniy Vahlis. Protecting cryptographic keys against continual leakage. In *Annual Cryptology Conference*, pages 41–58. Springer, 2010.
24. Jonathan Katz and Vinod Vaikuntanathan. Signature schemes with bounded leakage resilience. In *ASIACRYPT*, pages 703–720, 2009.
25. Eike Kiltz and Krzysztof Pietrzak. Leakage resilient elgamal encryption. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 595–612. Springer, 2010.
26. Brian LaMacchia, Kristin Lauter, and Anton Mityagin. Stronger security of authenticated key exchange. In *International Conference on Provable Security*, pages 1–16. Springer, 2007.
27. Tal Malkin, Isamu Teranishi, Yevgeniy Vahlis, and Moti Yung. Signatures resilient to continual leakage on memory and computation. In *Theory of Cryptology Conference*, pages 89–106, 2011.
28. Silvio Micali and Leonid Reyzin. Physically observable cryptography (extended abstract). In *Theory of Cryptology Conference*, pages 278–296, 2004.
29. Moni Naor and Gil Segev. Public-key cryptosystems resilient to key leakage. In *Advances in Cryptology-CRYPTO 2009*, pages 18–35. Springer, 2009.
30. Baodong Qin and Shengli Liu. Leakage-resilient chosen-ciphertext secure public-key encryption from hash proof system and one-time lossy filter. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 381–400. Springer, 2013.
31. Baodong Qin and Shengli Liu. Leakage-flexible cca-secure public-key encryption: simple construction and free of pairing. In *International Workshop on Public Key Cryptography*, pages 19–36. Springer, 2014.
32. Victor Shoup. Oaep reconsidered. *Journal of Cryptology*, 15(4):223–249, 2002.
33. Victor Shoup. Sequences of games: a tool for taming complexity in security proofs. *IACR Cryptology EPrint Archive*, 2004:332, 2004.
34. Yuyu Wang and Keisuke Tanaka. Generic transformation to strongly existentially unforgeable signature schemes with leakage resiliency. In *International Conference on Provable Security*, pages 117–129. Springer, 2014.

Appendix

A Additional Preliminaries

A.1 Basics of Information Theory.

Definition 8. (Min-Entropy). *The min-entropy of a random variable X, denoted as $\mathrm{H}_\infty(\mathrm{X})$ is defined as $\mathrm{H}_\infty(\mathrm{X}) \stackrel{\text{def}}{=} -\log(\max_x \Pr[X = x])$. This is a standard notion of entropy used in cryptography, since it measures the worst-case predictability of X.*

Definition 9. (Average Conditional Min-Entropy). *The average-conditional min-entropy of a random variable X conditioned on a (possibly) correlated variable Z, denoted as $\widetilde{\mathrm{H}}_\infty(X|Z)$ is defined as*

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

This measures the worst-case predictability of X by an adversary that may observe a correlated variable Z.

The following bound on average min-entropy was proved in Dodis et al. [14].

Lemma 1. [14] *For any random variable X, Y and Z, if Y takes on values in $\{0,1\}^\ell$, then*

$$\widetilde{\mathrm{H}}_\infty(X|Y,Z) \geq \widetilde{\mathrm{H}}_\infty(X|Z) - \ell \quad \text{and} \quad \widetilde{\mathrm{H}}_\infty(X|Y) \geq \widetilde{\mathrm{H}}_\infty(X) - \ell.$$

A.2 Leakage-resilient (LR) Chameleon Hash Functions

In this section we give the definition of LR chameleon hash functions (CHF) in the *bounded* memory leakage model following [34].

LR-CHF in bounded leakage model: Informally, a chameleon hash function (CHF) is a collision-resistant hash function, the only difference being that it is easy to find collision given a trapdoor. Without knowing the trapdoor it is hard to find any collision. Leakage-resilient chameleon hash functions (LR-CHF) postulate that it is hard to find collisions, even when the adversary learns bounded leakage/information about the secret key. Formally, an λ-LR-CHF $\mathsf{ChamH} : \mathcal{D} \times \mathcal{R}_{cham} \to \mathcal{I}$, where $\mathcal{D}$ is the domain, $\mathcal{R}_{cham}$ the randomness space and $\mathcal{I}$ the range, consists of the algorithms ($\mathsf{Cham.KeyGen}$, $\mathsf{Cham.Eval}$, $\mathsf{Cham.TCF}$).

1. $\mathsf{Cham.KeyGen}(1^\kappa, \lambda)$: The key generation algorithm takes as input 1^κ and the leakage bound λ as parameters and output an evaluation key along with a trapdoor $(\mathsf{hk}, \mathsf{ck})$. The public key hk defines a chameleon hash function, denoted $\mathsf{ChamH}_{\mathsf{hk}}(.,.)$.
2. $\mathsf{Cham.Eval}(\mathsf{hk}, m, r)$: The hash function evaluation algorithm that takes as input hk, a message $m \in \mathcal{D}$, and a randomizer $r \in \mathcal{R}_{cham}$ and outputs a hash value $h = \mathsf{ChamH}_{\mathsf{hk}}(m, r)$.
3. $\mathsf{Cham.TCF}(\mathsf{ck}, (m, r), m')$: The trapdoor collision finder algorithm takes as the trapdoor ck, a message-randomizer pair (m, r), an additional message m', and outputs a value $r' \in \mathcal{R}_{cham}$ such that $\mathsf{ChamH}_{\mathsf{hk}}(m, r) = \mathsf{ChamH}_{\mathsf{hk}}(m', r')$.

A λ-LR-CHF must satisfy the following three properties:

- **Reversibility:** The reversibility property is satisfied if $r' = \mathsf{Cham.TCF}(\mathsf{ck}, (m, r), m')$ is equivalent to $r = \mathsf{Cham.TCF}(\mathsf{ck}, (m', r'), m)$.
- **Random Trapdoor Collisions:** The random trapdoor collision property is satisfied if for a trapdoor ck, an arbitrary message pair (m, m'), and a randomizer r, $r' = \mathsf{Cham.TCF}(\mathsf{ck}, (m, r), m')$ has uniform probability distribution on the randomness space $\mathcal{R}_{cham}$.
- **LR-collision resistance:** The LR collision-resistance property is satisfied if for any PPT adversary $\mathcal{A}$, we have that the following advantage in negligible:

$$\mathsf{Adv}^{\mathrm{coll}}_{\mathcal{A},\mathsf{ChamH}}(\kappa) = |\Pr[(\mathsf{hk}, \mathsf{ck}) \leftarrow \mathsf{Cham.KeyGen}(1^\kappa, \ell); (m, r), (m', r') \leftarrow \mathcal{A}^{O^{\kappa,\lambda}_{\mathsf{ck}}}(\mathsf{hk}) : (m, r) \neq (m', r') \wedge \mathsf{ChamH}_{\mathsf{hk}}(m, r) = \mathsf{ChamH}_{\mathsf{hk}}(m', r')]|$$

where $O^{\kappa,\lambda}_{\mathsf{ck}}$ is the leakage oracle to which $\mathcal{A}$ can adaptively query to learn at most λ bits of information about the trapdoor ck.

A.3 Pseudo-random Functions

$F : \Sigma^k \times \Sigma^m \to \Sigma^n$ is a $(\epsilon_{\mathsf{prf}}, s_{\mathsf{prf}}, q_{\mathsf{prf}})$ secure pseudo-random function (PRF) if no adversary of size s_{prf} can distinguish F (instantiated with a random key) from a uniformly random function, i.e., for any $\mathcal{A}$ of size s_{prf} making q_{prf} oracle queries we have:

$$|\Pr_{K \xleftarrow{\$} \Sigma^K}[\mathcal{A}^{F(K,.)} \to 1] - \Pr_{R_{m,n}}[\mathcal{A}^{R_{m,n}(\dot{})} \to 1]| \leq \epsilon_{\mathsf{prf}},$$

where $R(m, n)$ is the set of all functions from $\Sigma^m \to \Sigma^n$.

A.4 UF-CMLA-Secure Signature Schemes

We review the definition of UF-CMLA security according to Katz et al. [24]. The leakage function f_i is an adversary chosen efficiently computable adaptive leakage function, which leaks $f_i(sk)$ from a secret key sk.

Definition 10 (Unforgeability Against Chosen Message Leakage Attacks (UF-CMLA)). *Let κ be the security parameter and λ be the leakage parameter. Let $\mathsf{LR\text{-}SIG} = (\mathsf{SIGkg}, \mathsf{SIGsign}, \mathsf{SIGvfy})$ be a signature scheme, we define $\mathsf{Adv}^{\text{UF-CMLA}}_{\mathsf{LR\text{-}SIG},\mathcal{B}_{\mathsf{sig}}}(\kappa)$ as the advantage of any PPT adversary $\mathcal{B}_{\mathsf{sig}}$, winning the following game:*

1. $(sk^{\text{sig}}, pk^{\text{sig}}) \xleftarrow{\$} \mathsf{SIGkg}(1^{\kappa})$

2. $(m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}(\cdot,\cdot)}(pk^{\text{sig}})$

3. *If* $\mathsf{SIGvfy}(pk^{\text{sig}}, m^*, \sigma^*) =$ *"true" and* m^* *is not been previously signed, then* $\mathcal{B}_{\mathsf{sig}}$ *wins.*

Oracle $\mathcal{O}(m, f_i)$

- $\sigma \xleftarrow{\$} \mathsf{SIGsign}(sk^{\text{sig}}, m)$
- $\gamma_i \leftarrow f_i(sk^{\text{sig}})$
- *If* $\sum_{i=1} |\gamma_i| \leq \lambda$
  - $\gamma \leftarrow \gamma_i$
  - $\gamma \leftarrow \perp$
- *Return* (σ, γ)

We say the signature scheme LR-SIG *is* UF-CMLA*-secure, if* $\mathsf{Adv}^{\text{UF-CMLA}}_{\text{LR-SIG},\mathcal{B}_{\mathsf{sig}}}(\kappa)$ *is negligible.*

Katz et al. [24] constructed an UFCMLA-secure signature scheme in bounded leakage model in which $n = 1$. It contains signing and verification operations based on NIZK proofs, where signature can be generated with cost of 2 **Exp**onentiations, and verified with cost of 4 Exponentiations (with a simple NIZK proof).

B Leakage-resilient Cryptography and Leakage Models

During the past two decades side-channel attacks have arisen as a popular method of attacking cryptographic systems. In order to abstractly model the side-channel attacks and analyze the security of cryptographic primitives against them, cryptographers have proposed the notions of *leakage-resilient* cryptography, introducing various leakage models. [2, 6, 27, 28], introducing various leakage models.

In the work of Micali and Reyzin [28], a general framework was introduced to model the leakage, that occurs during computation with secret parameters, which is widely known as *Only computation leaks information (OCLI) axiom*. They mentioned that the leakage only occurs from the secret memory portions which are actively involved in computations. The leakage amount is bounded per computation, though the adversary is allowed to obtain the leakage from many computations. Therefore, the overall leakage amount is unbounded. Since, this assumption enforces that the leakage is only occurred due to computations, this does not cover the attacks that happen due to the leakage from the memory such as malware attacks, cold-boot attacks etc.

Inspired by the cold-boot attacks, Akavia et al. [2] constructed a general framework to model bounded leakage attacks, which is widely known as *bounded-memory leakage* model. The adversary chooses an arbitrary polynomial time leakage function, f and sends it to the leakage oracle. The leakage oracle returns $f(sk)$ to the adversary, where sk is the secret key. The only restriction here is that the sum of output length of all the leakage functions that an adversary can obtain is bounded by some parameter λ, which is smaller than the size of sk. This leakage model does not address the continuous leakage from the memory, which can often happen due to attacks such as malware attacks.

Previous works of Zvika et al. [8] and Dodis et al. [11] presented a *continual-memory leakage* model, in which it is assumed that the leakage happens from the entire secret memory. The other characteristics of this model is same as the OCLI model. This leakage model is stronger than the OCLI model, because here the adversary can obtain the leakage from the entire memory regardless of computations.

Differently, Dodis et al. [13] introduced a leakage model, where the adversary is allowed to obtain the leakage as any computationally uninvertible function of the secret key as auxiliary input. That model eliminates the concept of leakage parameter, but enforcing the hardness parameter instead.

C Vulnerability of the NIKE protocol of [16] in the bounded-leakage setting.

In this section we show that the NIKE protocol of Freire et al. [16] from pairings in the standard model is completely insecure, even if the adversary is given only a *single* bit of leakage on the secret key. The attack exploits the fact that the adversary can ask any arbitrary leakage function as long as the output of the function is length shrinking in its input size. In particular, the secret key of a party in the NIKE protocol in Freire et al. [16] is a field element, i.e., $x \in \mathbb{Z}_p$, and one of the components of the public key is $Z = g^x$. The shared key between two parties ID_i and ID_j has the structure $e(S^{x_i}, Z_j)$, where S is a public element, $Z_j = g^{x_j}$ and x_i and x_j are the secret keys of parties ID_i and ID_j respectively.

Now, given the public-key, the adversary can encode the function that leaks the hardcore bit of the discrete logarithm of Z. In other words, he can specify the leakage function in such a way such that it leaks exactly the most significant bit (MSB) of x. Note that the MSB of x is actually the hardcore bit of the discrete logarithm function. So, with a single bit of leakage the adversary can recover x completely and hence he can distinguish the shared secret key from a random key with probability 1 and win the indistinguishability game. In fact here with only a single bit of leakage the adversary can perform *key recovery attack*, which is stronger than the attack on the indistinguishability game.