Research Article

Suvradip Chakraborty, Janaka Alawatugoda* and Chandrasekaran Pandu Rangan

New approach to practical leakage-resilient public-key cryptography

https://doi.org/10.1515/jmc-2019-0014
Received April 25, 2019; revised November 14, 2019; accepted November 28, 2019

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 an appropriate security model for LR-NIKE protocols in the bounded memory leakage (BML) settings. We then show a secure construction of the LR-NIKE protocol in the BML setting that achieves an optimal leakage rate, i.e., $1 - o(1)$. Our construction of LR-NIKE requires a minimal use of a leak-free hardware component. We argue that the use of such a leak-free hardware component seems to be unavoidable in any construction of an LR-NIKE protocol, even in the BML setting. Finally, we show how to construct the aforementioned leakage-resilient primitives from such an LR-NIKE protocol as summarized below. All these primitives also achieve the *same* (optimal) leakage rate as the underlying LR-NIKE protocol. We show how to construct a leakage-resilient (LR) IND-CCA-2-secure PKE scheme in the BML model generically from a bounded LR-NIKE (BLR-NIKE) protocol. Our construction of LR-IND-CCA-2 secure PKE *differs* significantly from the state-of-the-art constructions of these primitives, which mainly use hash proof techniques to achieve leakage resilience. Moreover, our transformation preserves the leakage-rate of the underlying BLR-NIKE protocol. We introduce a new leakage model for AKE protocols, in the BML setting, and present a leakage-resilient AKE protocol construction from the 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 the LR-NIKE protocol.

Keywords: Leakage-resilient cryptography, public-key cryptography, non-interactive key exchange, authenticated key exchange, low-latency key exchange

MSC 2010: 94A60, 14G50, 11T71, 68P25, 68M12

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

***Corresponding author: Janaka Alawatugoda,** Department of Computer Science and Engineering, University of Peradeniya, Peradeniya, Sri Lanka, e-mail: alawatugoda@eng.pdn.ac.lk. http://orcid.org/0000-0001-9431-5836
Suvradip Chakraborty, Chandrasekaran Pandu Rangan, Department of Computer Science and Engineering, Indian Institute of Technology Madras, Chennai, India, e-mail: suvradip@cse.iitm.ac.in, rangan@cse.iitm.ac.in

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, 9, 14, 16, 30].

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 $f(sk)$, 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) [13], 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 bandwidth-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 [19]. Freire et al. [19] proposed formal security models for NIKE and efficient constructions of NIKE in these models. Although the NIKE constructions of [19] are 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 [19] 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, 11], but the leakage resiliency of NIKE remains largely unstudied. We note that Morita et al. [31] studied the security of NIKE protocols in the face of related key attacks (RKA) and show various implications and separation results (in the RKA setting) among the security notions of NIKE put forward by Freire et al. [19]. Morita et al. [31] have considered related-key attacks on NIKE and showed that there is a separation among the security notions. We can see that the equivalences between the different NIKE models of Freire et al. [19] easily carry forward to the leakage setting as well. Therefore, such separation does not hold in the leakage setting. We refer the interested readers to Morita et al. [31] and Freire et al. [19] for the detailed exposition.

As one of the central applications of leakage-resilient NIKE (LR-NIKE), we show how to construct a leakage-resilient IND-CCA-2-secure PKE scheme generically from LR-NIKE (in the bounded-memory leakage setting). All the previous constructions of leakage-resilient IND-CCA-2 (LR-IND-CCA-2) secure PKE schemes rely solely on hash proof techniques to achieve leakage resiliency. However, the generic approach of constructing a leakage-resilient CCA secure PKE scheme *solely* using hash proof systems (HPS) is inherently limited to a leakage rate below $\frac{1}{2}$ as pointed out by Dodis et al. [15]. The leakage rate of the state-of-the-art constructions of a LR-IND-CCA-2-secure PKE scheme was later improved in the subsequent works of Qin et al. [35, 36], which achieved leakage rates of $\frac{1}{2} - o(1)$ and $1 - o(1)$, respectively. They could achieve a leakage rate of $\frac{1}{2} - o(1)$ by using HPS and one-time lossy filters (OTLF)[1] 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 a refined subgroup indistinguishability (RSI) assumption over composite order groups. The parameters of their construction are also large due to the use of composite-order groups.

We deviate 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 a leakage-resilient

1 Note that this circumvents the impossibility result of Dodis et al. [15] since the analysis of [15] considered the fact that the LR-IND-CCA-secure PKE was constructed solely from HPS; whereas, in [35, 36], they do not solely use HPS and instead rely on both HPS and OTLF for their construction.

chameleon hash function (which in turn relies only on a strong collision-resistant 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.

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, 11] 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) is 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 Google's QUIC[2] protocol. Further, a low-latency mode is currently under discussion for inclusion in TLS version 1.3. Although the first formal model of LLKE was studied by Hale et al. [23], 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.

Our 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. We point out a subtle point in defining leakage-resilient NIKE protocols as discussed below and show that care must taken while defining it. Finally, we show how to construct a secure NIKE protocol.

Our model: Our model of leakage-resilient NIKE adopts and generalizes the CKS-heavy model of NIKE proposed by Freire et al. [19] in the setting of leakage. Firstly, we notice that defining the security of NIKE protocols in the setting of leakage requires more care than defining the security model for NIKE protocols in the non-leakage setting. Note that the shared key of a party in a two-party NIKE protocol is simply a deterministic function of its own secret key and the public key of the other party. However, in the setting of leakage, there is a simple attack on any LR-NIKE protocol as follows: the adversary can simply encode the (description) of the shared-key derivation function as the leakage function, with the public key of the other party hard-coded in the function. Hence, using this, it can directly leak from the shared key. To prevent this trivial attack, we must impose some meaningful restrictions. One plausible solution to circumvent the above attack could be to restrict the class of allowable leakage functions. In particular, one may assume that the leakage functions are not allowed to access the public parameters of the system (and hence the public keys of the parties) while leaking from the secret key of one party. However, this seems to be an unnatural restriction on the leakage functions since the public parameters (and in particular the public keys of all parties) are known to all the parties, and hence to the adversary also. To this end, we propose an alternative model for LR-NIKE protocols that avoids the above impossibility result. In particular, we assume that all parties participating in a NIKE protocol are equipped with a *leak-free* hardware component which can be used to shield a small part of their public keys. The leakage functions can access the public keys of all the parties, except the compo-

2 https://www.chromium.org/quic

nents stored in the leak-free hardware. Note that the leak-free hardware should be used in a minimal way in any LR-NIKE protocol[3]. Now we informally define our security model for LR-NIKE protocols.

Our model for LR-NIKE assumes that all parties participating in the protocol are equipped with a leak-free hardware component. This hardware is used to store a small part of their public keys, hence effectively shielding these parts from the view of the adversary. The specification of which components are stored in the hardware is protocol-specific. However, we stress that the usage of the leak-free hardware should be *minimal*. Our security model for LR-NIKE protocols 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 (bounded) 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.

Our construction: The starting point of our construction is the NIKE scheme of Freire et al. [19]. 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 [30] or under the *split-state* assumption [24, 27, 40]. 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 the 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 the bounded memory leakage model.

So, a major challenge in our construction is to come up with a leakage-resilient exponentiation operation achieving a leakage rate of $1 - o(1)$ in the global memory leakage model. Our first idea is to use the techniques of [1, 6, 33] to perform leakage-resilient exponentiation. In particular, let $\mathbb{G}$ be a group of prime order p, and let $g_1, \ldots, g_n$ be random elements of the group. Then the vector $\mathbf{x} = (x_1, \ldots, x_n) \in (\mathbb{Z}_p^*)^n$ is said to be a discrete log representation of some element $y \in \mathbb{G}$ with respect to $g_1, \ldots, g_n$, if $y = \prod_{i=1}^{n} g_i^{x_i}$. To incorporate this in our construction of NIKE, the elements $g_1, \ldots, g_n$ 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 [1, 6, 33], it is also shown that, as long as the leakage on each representation is bounded by $(1 - \frac{2}{n})|\mathbf{x}|$, 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. [19]. 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 idea of the twisted-pair PRF trick [20], but carefully adapted 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 the output of the twisting function is computationally indistinguishable from a uniform value over the same range. For our construction of a leakage-resilient twisted-pair PRF trick, we add strong randomness extractors as pre-processors to this original twisting technique [20]. The guarantee is that the output of our leakage-resilient twisted-pair PRF function is computationally indistinguishable from a uniform

3 Jumping ahead, in our LR-NIKE protocol, the leak-free hardware is only used to store a short seed used for randomness extraction.

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 PRFs. 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 obtain the optimum leakage rate of $1 - o(1)$. Combined with a bounded leakage-resilient CHF tolerating a 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, the main drawback of our leakage-resilient NIKE construction is that it requires a *leak-free* hardware assumption. However, as argued before, such an assumption is not merely an artifact of our protocol, but is likely to be required for constructing any LR-NIKE protocol. Since we allow the leakage function to access all the public parameters of the system (and hence the public keys of parties), the adversary can directly leak from the shared key of the parties by encoding the shared key derivation function as the leakage function. Our protocol requires a *leak-free* 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 is stored in a leak-free hardware component. We also note that, since extractors are information theoretic gadgets, reusability of the seed is not permitted. Hence each party needs to store a short random seed corresponding to every other party in the leak-free hardware for establishing session keys with them. Minimizing this leak-free hardware assumption is an interesting and challenging problem we leave open. A leak-free hardware component assumption was also used in many prior works in leakage-resilient cryptography [18, 22, 25], although in the context of continual memory leakage. In particular, in [25], it is assumed that the leak-free component can produce random encryptions of fixed messages. In [22], it is assumed that there is a linear number of such leak-free components and each component is capable of sampling from a fixed polynomial-time computable distribution. In [18], 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 do not require the leak-free hardware to perform any expensive computation. We only require it to store several seeds of the extractor, which are typically short random strings.

Comparison with Chen et al. [11, 12]: It is instructive to compare our construction of BLR-NIKE with the leakage-resilient AKE protocols of Chen et al. [11, 12]. Chen et al. [11, 12] proposed constructions of AKE protocols secure against "after-the-fact" bounded-memory [11] (resp. auxiliary input [12]) leakage attacks. Our main idea of the BLR-NIKE protocols shares some technical similarities with [11, 12]. Namely, the main idea of both these works is an "extract-then-PRF" technique, where a randomness extractor is applied on the long-term secret key (in case of [11, 12] also on the ephemeral secret key) and then two PRFs with reversing keys are applied to the extracted values (twisted PRF trick). Our NIKE protocol uses the framework of Freire et al. [19] and uses the "extract-then-PRF" technique to tackle key leakage attacks. However, in this work, we consider before-the-fact leakage attacks, in contrast to [11, 12] which considered after-the-fact leakage, however, at the cost of restricting the leakage model further to avoid the impossibility result related to after-the-fact leakage.

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 $1 - o(1)$.

	# exponentiations			# multiplications			# pairings		
Scheme	**KeyGen**	**Enc.**	**Dec.**	**KeyGen**	**Enc.**	**Dec.**	**KeyGen**	**Enc.**	**Dec.**
Qin and Liu [35]	n^2	(n^2+n+2)	(n^2+2n)	(n^2+n)	$2n^2$	$(2n^2+n)$	NA	NA	NA
Qin and Liu [36]	(n^2+n+1)	$(3n+2)$	$(3n+1)$	n	$2n$	$2n$	NA	NA	NA
This work	$(n+5)$	$(2n+8)$	$(n+3)$	$(n+3)$	$(2n+6)$	$(n+3)$	NA	3	3

Scheme	**Size of ciphertext C**	**Leakage rate**	**Complexity assumptions**	**Additional assumptions**
Qin and Liu [35]	$\mathbb{G}^2 \times \mathbb{Z}_q^n \times \{0,1\}^m \times \widetilde{\mathbb{G}}^{n^2 \times n^2}$	$\frac{1}{2} - o(1)$	DDH	No
Qin and Liu [36]	$\mathbb{G}_{\tau_1} \times \{0,1\}^m \times \mathbb{G}^{n+1} \times \mathcal{T}_c$	$1 - o(1)$	RSI over composite order groups	No
This work	$\mathbb{G}^2 \times \mathbb{Z}_q^{n+1} \times \{0,1\}^s$	$1 - o(1)$	DBDH	*Leak-free component*

Table 1: Comparison among the LR-IND-CCA PKE schemes. Here DDH stands for the decisional Diffie–Hellman problem. RSI denotes the refined subgroup indistinguishability problem, and DBDH refers to the decisional bilinear Diffie–Hellman problem.

Our construction: Our generic transformation from an LR-NIKE to an LR-IND-CCA-2-secure PKE scheme essentially follows and adapts the ideas of Freire et al. [19] 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 the 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 IND-CCA-2-KEM, one can easily get 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 LR-NIKE with a 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 less* 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 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. 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. According to Benhamouda et al. [7], pairing is roughly three times slower than computing an exponentiation. Therefore, each encryption and decryption cost is roughly nine times of an exponentiation. Since we can achieve the optimum leakage rate (albeit using a leak-free hardware assumption), this additional computation cost is reasonable. In the table, $n \in \mathbb{N}$ is usually the number of generators required for the construction. Also, [36] 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 [36], and $\{0,1\}^s$ denotes the seed space of a strong randomness extractor. Lastly, we want to stress that, although our scheme is more efficient than that of Qin et al. [35, 36] in terms of computational cost and communication complexity and also achieves an optimal leakage rate, our scheme is *not* superior to them because of the use of a leak-free hardware component. The leak-free hardware is clearly a *strong assumption*.

Scheme	Leakage model B/C	BFL/AFL	Complexity assumptions	Leakage rate
Moriyama and Okamoto [32]	B	BFL	DDH, HPS, CR, π-PRF, Ext	$1 - o(1)$
Alawatugoda, Boyd and Stebila [3]	C	AFL	CCLA-2 secure PKE, PRF	*sub-optimal**
Alawatugoda, Stebila and Boyd [4]	B/C	AFL	DDH, ODH, ε-PG-IND and CPLA-2 secure PKE, UFCMLA-secure signature, secure KDF, PRF	min{CPLA-2 PKE, UFCMLA-sig}
Chen, Mu, Yang, Susilo and Guo [11]	B	AFL	DDH, HPS, CR, π-PRF, Ext	min{HPS, Ext}
This work	B	BFL	BLR-CKS-heavy NIKE, UFCMLA-secure signature	$1 - o(1)$

Table 2: Comparison among the LR-AKE schemes. Here B/C stands for either bounded or continuous memory leakage; BFL/AFL denotes the resilience of the AKE protocols to before-the-fact/after-the-fact leakage attacks. The shorthands DDH, HPS, CR, π-PRF and Ext stand for the decisional Diffie–Hellman problem, hash proof systems, collision-resistant hash function, pairwise-independent PRF families and strong extractors, respectively. CPLA-2 (resp. CCLA-2) secure PKE denotes an adaptively chosen plaintext (resp. ciphertext) after-the-fact leakage secure public-key cryptosystem. ε-PG-IND refers to the pair-generation indistinguishable PKE scheme; ODH and KDF refer to the oracle Diffie–Hellman and secure key derivation functions, respectively. BLR-CKS-heavy stands for bounded leakage-resilient "Cash–Kiltz–Shoup"-heavy model (BLR analogue of the CKS-heavy model).

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 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 [30]. We present an eCK-style [28] 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. [8] to the setting of leakage. In particular, Bergsma et al. [8] 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 [26]. 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 an appropriate choice of parameters. We refer the reader to Table 2 for a more detailed comparison of leakage-resilient AKE protocols.

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. [23] under mutual authentication of the client as 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. [23], we show a generic construction of leakage-resilient LLKE protocol from a leakage-resilient NIKE protocol. In particular, we require an LR-NIKE scheme and a UF-CMLA secure signature scheme. Plugging them appropriately in our context, we obtain the construction of a leakage-resilient LLKE protocol. Moreover, the leakage rate enjoyed by our LLKE protocol is also optimal, i.e., $1 - o(1)$.

Alternative thought

It is possible to compare our LR-NIKE protocol (in the bounded-memory leakage model) with an alternative simpler construction, which is essentially an adaptation of the idea which we made explicit only for the LR-CCA-secure PKE scheme.

Essentially, the idea is as follows: sample a random string r, and use a (seeded) randomness extractor to extract another the key generation function. The seed s of the extractor is stored in the leak-free hardware component. The intuition behind the security is that, since the leakage from r is bounded and the seed s is kept outside the view of the adversary, the extracted value r' should look random to the adversary (the parameters need to be appropriately set to argue this). While we have made this explicit only for PKE, the above simple idea also works for achieving leakage resilience of NIKE protocols starting from any NIKE protocol in the CKS-heavy model. In fact, our construction of NIKE is exactly along this line: we start with the NIKE protocol of Freire et al. [19] and show the above NIKE protocol can be made (bounded) leakage-resilient by using the above trick. The reason we choose to give a concrete NIKE protocol is that the protocol of Freire et al. [19] is the only existing NIKE protocol secure in the CKS-heavy model (the base model we consider in our paper as well) in the standard model. Hence we start with the protocol of Freire et al. [19] and explicitly show the above idea to bootstrap its security in the bounded leakage setting.

For other primitives like AKE and LLKE, a similar idea works, and we can construct leakage-resilient versions of these primitives in a stand-alone manner using the above idea (given a leak-free hardware assumption). However, the focus of the paper is to present LR-NIKE as a unified paradigm for constructing other leakage-resilient primitives like PKE, AKE and LLKE. In particular, a construction of continuous leakage-resilient NIKE will directly translate into a construction of continuous leakage-resilient PKE, AKE, LLKE via our transformations (which would otherwise not be possible using the above simpler transformations). Also, any improvement in the construction of NIKE will directly impact the efficiency of the corresponding PKE, AKE and LLKE schemes.

2 Preliminaries

2.1 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 $[a] = \{1, 2, \ldots, a\}$. 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 $C = (C_1, \ldots, C_n) \in \mathbb{Z}_p^n$, we denote by g^C the vector $C = (g^{C_1}, \ldots, g^{C_n})$. Furthermore, for a vector $D = (D_1, \ldots, D_n) \in \mathbb{Z}_p^n$, 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 this is denoted by $X \approx_\varepsilon Y$. 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.

2.2 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 an $(\varepsilon_{\text{prf}}, s_{\text{prf}}, q_{\text{prf}})$-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 a 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 a description of multiplicative cyclic groups of the 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, ψ is an efficiently computable isomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$ and $g_1 = \psi(g_2)$.

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 [21] and also used in [19]. Formally, we say that the DBDH-2 assumption holds for type-2 pairings if the advantage of the adversary $\mathcal{A}_{\text{DBDH-2}}$ denoted by $\text{Adv}^{\text{dbdh-2}}_{\mathcal{A}_{\text{DBDH-2}}, \mathcal{G}_2}(\kappa)$ is negligible, where

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

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 the syntax of leakage-resilient non-interactive key exchange (LR-NIKE) protocols. We denote by $\mathcal{PK}$, $\mathcal{SK}$ and $\mathcal{SHK}$ the space of public keys, secret keys and shared keys of LR-NIKE, respectively. When we write pk_i for the i-th public key, we mean that pk_i is associated with the user with identifier $\text{ID}_i \in \mathcal{IDS}$, where $\mathcal{IDS}$ denotes the identity space[4]. Formally, an LR-NIKE scheme, LR-NIKE, consists of a tuple of algorithms (NIKEcommon_setup, NIKEgen, NIKEkey) with the functionalities specified below.

4 Note that, we are *not* in the identity-based setting.

(i) $\mathsf{NIKEcommon\_setup}(1^\kappa, \lambda(\kappa))$: 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.
(ii) $\mathsf{NIKEgen}(1^\kappa, params)$: 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 $(pk, sk) \in \mathcal{PK} \times \mathcal{SK}$.
(iii) $\mathsf{NIKEkey}(pk_i, sk_j)$: 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 Bounded leakage-resilient non-interactive key exchange (BLR-CKS-heavy) security model

In this section, we present the formal security model for leakage-resilient non-interactive key exchange (LR-NIKE). Before defining our security model for LR-NIKE protocols, we present an impossibility result related to LR-NIKE protocols in Section 4.1.1. In particular, we show that if the leakage function is an arbitrary polynomial-time computable function having access to the public parameters, we cannot hope to construct a secure LR-NIKE protocol, even in the bounded memory leakage model. In Section 4.1.2, we show how to circumvent this impossibility result by enforcing some restrictions on the class of allowable leakage functions in our BLR-CKS-heavy security model for NIKE.

4.1.1 Impossibility of LR-NIKE protocols

In this section, we present an impossibility result of constructing the LR-NIKE protocol, even in the bounded leakage model. Then we suitably adapt our security model to circumvent this impossibility result. Let us assume that the NIKE protocol is run between two users, say, Alice and Bob. The key pairs of Alice and Bob are (pk_A, sk_A) and (pk_B, sk_B), respectively. In the leakage setting, the adversary may ask bounded leakage from the secret keys of both Alice and Bob. Note that the shared key between Alice and Bob is a deterministic function of their own secret keys and the public key of the other party, namely, for Alice (for Bob), the shared key shk_{AB} is derived as $\mathsf{NIKEkey}(pk_B, sk_A)$ (as $\mathsf{NIKEkey}(pk_A, sk_B)$). Now the adversary can set the leakage function for Alice as $L(\cdot) = L'_{pk_B}(\cdot) = \mathsf{NIKEkey}(pk_B, \cdot)$, i.e. the adversary can specify the leakage function as the shared key derivation function $\mathsf{NIKEkey}$ with the public key of the other party hard-coded in it. This allows the adversary to directly leak from the shared key shk_{AB} established between Alice and Bob. If the adversary can leak sufficiently many bits of shk_{AB}, then, with very high probability, the adversary can distinguish the shk_{AB} from a random key.

The above attack demonstrates that if we allow the leakage function to be arbitrary polynomial-time computable functions with access to all the public parameters (and hence to the public keys of all the parties) of the system, it is *impossible* to have a secure instantiation of an LR-NIKE protocol. Hence we need to enforce some meaningful restrictions on the leakage functions. In particular, one may assume that the leakage functions are not allowed to access the public parameters of the system. This can be enforced by having the adversary specify the set of leakage functions before receiving the public parameters, the so-called "*non-adaptive*" leakage model. However, this is necessarily a much more restrictive model than the *adaptive* leakage model, where the leakage may depend on the parameters on the system. To circumvent the impossibility result in the adaptive leakage model, we incorporate some assumptions in our security model for LR-NIKE. In particular, we assume that every party participating in a NIKE protocol is equipped with a *leak-free* hardware component which can be used to store a very small part of their public key. The leakage function can access the public keys of all the parties, except the components stored in the leak-free hardware. This

essentially provides a way to shield some part of the public key from the view of the adversary. The information that is stored in the leak-free hardware is implementation-specific. However, we stress that the leak-free hardware should be used in a minimal way in any LR-NIKE protocol[5]. Also, note that if we store the entire public key in the leak-free hardware, then our model is essentially equivalent to the non-adaptive leakage model. Hence our security model for LR-NIKE can be seen as a convex combination of the non-adaptive and adaptive leakage model, depending on the information that is stored in the leak-free hardware.

Lastly, we note that the above impossibility results do not carry forward to the setting of (interactive) AKE protocols. This is because the shared key between two parties in an AKE protocol does not only depend on the long-term keys of the parties, but also depends on the ephemeral secret and public keys. Hence, in this case, the leakage functions may be allowed to access the public keys of all the parties.

4.1.2 BLR-CKS-heavy security model for LR-NIKE

We model every party (including the adversary) as a probabilistic polynomial-time Turing machine (PPTM). In addition, we assume that all the legitimate parties involved in the protocol have access to an oracle tape, each per party. When required, the parties can enter into a query phase and can look up the response on its corresponding oracle tape. After obtaining the response, the parties can continue with the execution of the protocol.[6] Moreover, we assume that the adversary cannot read the contents of the oracle tape. In reality, the oracle tape models the leak-free hardware device available to each party. Equipped with this, we next present our BLR-CKS-heavy model for NIKE.

Our security model of LR-NIKE can be seen as generalization of the CKS-heavy security model introduced by Freire et al. [19] to appropriately model key leakage attacks. 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 attempt to register a public key in the system. So, in the leakage-free scenario, this setting was also considered in the work of Freire et al. [19], which they called the Simplified(S)-NIKE. 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. 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 4.1 (Extract query vs. Leakage queries). By issuing the Extract query, the adversary can learn the secret key of a party entirely. Separately, by issuing leakage queries the adversary gets a bounded 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 BLR-CKS-heavy model allows 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

5 In our construction, we use the leak-free hardware only to store a short seed used for randomness extraction.

6 In our construction, the oracle tape generates a short random string and stores it in as a response. When required, the parties can look up the contents of its oracle tape and use the string as a seed for randomness extraction in the LR-NIKE protocol.

the security game with certainty. But note that, in our BLR-CKS-heavy model, the adversary can also obtain bounded 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 BLR-CKS-heavy model allows the adversary to obtain more information than a non-leakage version of BLR-CKS-heavy model, namely the CKS-heavy model [19], and hence is necessarily *stronger* than the CKS-heavy model.

4.1.3 Adversarial powers

Our BLR-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 PPTM 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'. The adversary $\mathcal{A}$ is allowed to ask the following queries.

(i) RegisterHonest(1^κ, *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.

(ii) RegisterCorrupt(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*.

(iii) Extract(pk_U): In this query, the adversary $\mathcal{A}$ supplies the public key pk_U of an honest party. The challenger looks up the corresponding tuple $(honest, pk_U, sk_U)$ and returns the secret key sk_U to $\mathcal{A}$.

(iv) Reveal(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 the CorruptReveal query, one of the public keys 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 the 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 $\mathcal{A}$.

(v) Leakage: In the BLR-CKS-heavy security model, the total amount of leakage from the secret key of the underlying cryptographic primitives is bounded by the leakage parameter $\lambda = \lambda(\kappa)$. Here the adversary $\mathcal{A}$ supplies the description of an arbitrary polynomial-time computable function $f_i \in \mathcal{F}$ 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 class $\mathcal{F} = \{f_i\}_i$ of leakage functions is defined as $f_i : \{0,1\}^* \to \{0,1\}^{\lambda_i(\kappa)}$, where $\lambda_i(\kappa) < \lambda(\kappa)$. Secondly, the functions f_i cannot take as input the values $f(pk)$, where the value $f(pk)$ is stored in a leak-free hardware component, and f is a function of the public key pk.[7] The adversary $\mathcal{A}$ can specify multiple such leakage functions as long as the leakage bound is not violated, i.e., $\sum_i |f_i(sk)| \leq \lambda(\kappa)$, and $f_i \in \mathcal{F}$. 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.

(vi) 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 $\mathcal{A}$.

7 In our LR-NIKE, the public-key is $pk = (pk_1, pk_2, pk_3, pk_4, pk_5$, and $F(pk) = pk_5$.

$\mathcal{A}$'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 BLR-CKS-heavy model (see Definition 4.2).

Definition 4.2 (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.

(i) The adversary $\mathcal{A}$ is not allowed to ask Extract(pk_U) or Extract(pk_V) queries.
(ii) The adversary $\mathcal{A}$ is not allowed to ask HonestReveal(pk_U, pk_V) or HonestReveal(pk_V, pk_U) queries.
(iii) The total output length of all the leakage queries 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 $\sum_i |f_i(sk_V)| \leq \lambda(\kappa)$, and we require that $f_i \in \mathcal{F}$, where $\mathcal{F}$ is as defined above.

4.1.4 Security game and security definition

Definition 4.3 (BLR-CKS-heavy security game). The security of a NIKE protocol in the generic *(BLR-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 the NIKEcommon_setup algorithm to output the global parameters *params* and returns them 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-BLR-CKS-heavy valid*. The challenger chooses a random bit b to respond to this query. 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. Then $\mathcal{A}$ wins if $b' = b$.

Let $\text{Succ}_{\mathcal{A}}$ denote the event that $\mathcal{A}$ wins the above security game (Definition 4.3).

Definition 4.4 (BLR-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 *BLR-CKS-heavy* secure if there is no PPT algorithm $\mathcal{A}$ that can win the above *BLR-CKS-heavy* security game with non-negligible advantage. The advantage of an adversary $\mathcal{A}$ is defined as

$$\text{Adv}^{\text{BLR-CKS-heavy}}_{\Pi,\mathcal{A}}(\kappa, q_{\text{H}}, q_{\text{C}}, q_{\text{E}}, q_{\text{HR}}, q_{\text{CR}}) = |2\Pr(\text{Succ}_{\mathcal{A}}) - 1|.$$

Remark 4.5. We remark that our BLR-CKS-heavy security model for NIKE can be generalized in a straightforward manner to incorporate continuous memory leakage (CML) attacks. However, we do not give our security model for NIKE in the CML setting since we mainly focus on the construction of BLR-NIKE in this work. We note that a recent work [10] already solved the open problem of constructing LR-NIKE in the CML setting. However, they assume additional restrictions on the leakage model, namely a *split-state* model (where the secret key is split into multiple parts and it is assumed that the adversary can leak from both these parts, but in an independent manner) and also does not achieve the optimal leakage rate (i.e. $1 - o(1)$). We leave the construction of LR-NIKE in the CML setting in the non-split state model achieving optimal leakage rate as an exciting open problem.

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

Table 3: LR-NIKE protocol in the bounded leakage model (BLR-NIKE).

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 Freire et al. [19] (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.

4.2.1 Protocol BLR-NIKE: Construction of NIKE in the bounded-memory leakage model

Table 3 shows our construction of NIKE in the bounded-memory leakage model. The starting point of our construction is the NIKE protocol of [19]. Let $\mathcal{G}_2$ be a type-2 pairing parameter generation algorithm, i.e., it outputs $gk = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, p, e, \psi)$. Let $\mathsf{ChamH}_{hk}\colon \{0,1\}^* \times \mathcal{R}_{\mathsf{cham}} \to \mathbb{Z}_p$ 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 $\mathcal{R}_{\mathsf{cham}}$ denotes the randomness space of the hash function. Also, let $F\colon \{0,1\}^{\ell(\kappa)} \times \mathbb{Z}_p \to \mathbb{Z}_p$, $F'\colon \mathbb{Z}_p \times \{0,1\}^\kappa \to \mathbb{Z}_p$ be $(\varepsilon_{\mathrm{prf}}, s_{\mathrm{prf}}, q_{\mathrm{prf}})$ and $(\varepsilon'_{\mathrm{prf}}, s'_{\mathrm{prf}}, q'_{\mathrm{prf}})$ secure PRF families, and let $\mathsf{Ext}\colon \mathbb{Z}_p \times \{0,1\}^s \to \{0,1\}^{\ell(\kappa)}$ be an average-case (ν, ε)-extractor, with $\nu \ll \log p$ and $s = \omega(\log \kappa)$. Namely, it has $\log p$ bits of input, $\omega(\log \kappa)$-bit seed s and $\ell(\kappa)$-bit outputs, and for a random seed and input with ν bits of min-entropy, the output is ε-away from a uniform $\ell(\kappa)$-bit string. We can set the parameters appropriately to achieve this.

Setting the parameters of the extractor: For our construction, the seed s of the extractor Ext is stored in the leak-free hardware component. If the length of the seed s is $O(\log \kappa)$, the adversary can enumerate over the entire seed space itself, and the adversary can simply ask for leakage functions on the private key with enumeration of all possible seeds. This necessitates the length of s to be at least $\omega(\log \kappa)$. The classical result of [34] shows that, for every n, k, ε, there exist (k, ε)-extractors that use a seed of length

$d = \log(n-k) + 2\log(\frac{1}{\varepsilon}) + O(1)$ and output $m = k + d - 2\log(\frac{1}{\varepsilon})$ bits. In our construction, $n = \log p$ (since the input of Ext is from $\mathbb{Z}_p$), $k = n - \lambda = \log p - \lambda$ (since a leakage of λ bits can reduce the entropy of the source by at most λ bits). Hence the seed is of the length $s = \log(\lambda) + 2\log(\frac{1}{\varepsilon}) + O(1)$. By appropriately setting the value of p, λ and ε, one can show that $s \geq \omega(\log \kappa)$.

On the leak-free hardware assumption: As already stated in Section 4.1.2, we consider an additional assumption that each party involved in our LR-NIKE protocol has access to a *leak-free* secure hardware component. In our LR-NIKE construction, each party needs to store a short random seed (which is part of the public key of that party) corresponding to every other party with whom a shared key will be established. 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. However, as discussed in Section 4.1.1, the leak-free hardware assumption seems to be necessary for the construction for any secure LR-NIKE protocol, unless one sacrifices the leakage model further (non-adaptive leakage).

Theorem 4.6. *Let* ChamH_{hk} *be a family of bounded leakage-resilient chameleon hash function* (BLR-CHF). *Let* F *and* F' *be* $(\varepsilon_{\text{prf}}, s_{\text{prf}}, q_{\text{prf}})$ *and* $(\varepsilon'_{\text{prf}}, s'_{\text{prf}}, q'_{\text{prf}})$ *secure PRFs. Let* Ext *be a* (ν, ε)*-strong average case randomness extractor with seed length at least* $\omega(\log \kappa)$*, and let* p *be 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 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} *the number of* RegisterHonest *user queries. Then, using it, we can construct an adversary* $\mathcal{A}_{\text{DBDH-2}}$ *against the* DBDH-2 *problem such that*

$$\text{Adv}^{\text{BLR-CKS-heavy}}_{\text{BLR-NIKE},\mathcal{A}}(\kappa) \leq q_{\text{H}}^2\big(2\varepsilon + \varepsilon_{\text{prf}} + \varepsilon'_{\text{prf}} + \text{Adv}^{\text{coll}}_{\mathcal{A},\mathsf{ChamH}}(\kappa) + \text{Adv}^{\text{dbdh-2}}_{\mathcal{A}_{\text{DBDH-2}},\mathcal{G}_2}(\kappa)\big).$$

Proof. The proof of this theorem will proceed via the game hopping technique [38]: 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 primitives. Let $\text{Adv}_{\supset\delta}(\mathcal{A})$ denote the advantage of the adversary $\mathcal{A}$ in Game δ.

Game 0. This is the original security game with adversary $\mathcal{A}_{\text{DBDH-2}}$. 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.

$$\text{Adv}_{\text{Game}_0}(\mathcal{A}) = \text{Adv}^{\text{BLR-CKS-heavy}}_{\text{BLR-NIKE},\mathcal{A}}(\kappa).$$

Game 1. Initially, $\mathcal{A}_{\text{DBDH-2}}$ chooses two identities $\text{ID}_A, \text{ID}_B \in [q_{\text{H}}]$, where q_{H} denotes the number of RegisterHonest queries made by $\mathcal{A}_{\text{NIKE}}$. Effectively, $\mathcal{A}_{\text{DBDH-2}}$ is guessing that ID_A and ID_B to be honestly registered by $\mathcal{A}_{\text{NIKE}}$ will be involved in the Test query later. When $\mathcal{A}_{\text{NIKE}}$ makes its Test query on a pair of identities $\{\text{ID}_I, \text{ID}_J\}$, $\mathcal{A}_{\text{DBDH-2}}$ checks if $\{\text{ID}_I, \text{ID}_J\} = \{\text{ID}_A, \text{ID}_B\}$. If so, it continues with the simulation and gives the result to $\mathcal{A}_{\text{NIKE}}$; else it aborts the simulation.

$$\text{Adv}_{\text{Game}_1}(\mathcal{A}) \geq \text{Adv}_{\text{Game}_0}(\mathcal{A})/q_{\text{H}}^2.$$

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 $\widehat{x_A} \leftarrow \text{Ext}(x_A, s_A)$ and $\widehat{x_B} \leftarrow \text{Ext}(x_B, s_B)$, 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 $\nu \ll \log p$, even given the bounded leakage of λ bits, we have $(x_A, s_A, \text{Ext}(x_A, s_A)) \approx_\varepsilon (x_A, s_A, U_{\ell(\kappa)})$ and $(x_B, s_B, \text{Ext}(x_B, s_B)) \approx_\varepsilon (x_B, s_B, U_{\ell(\kappa)})$, 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 (i.e., s_A and s_B are stored in the oracle tape of party ID_A and ID_B, respectively), 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.

$$|\text{Adv}_{\text{Game}_2}(\mathcal{A}) - \text{Adv}_{\text{Game}_1}(\mathcal{A})| \leq 2\varepsilon.$$

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 $x'_A \leftarrow \mathrm{RF}(r_A) + F'_{r_A}(1^\kappa)$ and $x'_B \leftarrow \mathrm{RF}(r_B) + F'_{r_B}(1^\kappa)$, 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 $F\colon \{0,1\}^{\ell(\kappa)} \times \mathbb{Z}_p \to \mathbb{Z}_p$ and a random function RF.

$$|\mathrm{Adv}_{\mathrm{Game}_3}(\mathcal{A}) - \mathrm{Adv}_{\mathrm{Game}_2}(\mathcal{A})| \le \varepsilon_{\mathrm{prf}} + \varepsilon'_{\mathrm{prf}}.$$

Game 4. This game is identical to the previous game, except that the challenger now 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 challenger 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.

$$\mathrm{Adv}_{\mathrm{Game}_4}(\mathcal{A}) = \mathrm{Adv}_{\mathrm{Game}_3}(\mathcal{A}).$$

Game 5. In this game, the challenger changes the way in which it answers 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 RegisterCorrupt query. If

$$t_D = \mathsf{ChamH}_{hk}(Z_A \| \mathrm{ID}_A; r'_A) \quad \text{or} \quad t_D = \mathsf{ChamH}_{hk}(Z_B \| \mathrm{ID}_B; r'_B),$$

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 [37], we have

$$|\mathrm{Adv}_{\mathrm{Game}_5}(\mathcal{A}) - \mathrm{Adv}_{\mathrm{Game}_4}(\mathcal{A})| \le \mathrm{Adv}^{\mathrm{coll}}_{\mathcal{A},\mathsf{ChamH}}(\kappa).$$

Game 6. In this game, the DBDH-2 adversary $\mathcal{A}_{\text{DBDH-2}}$ receives as input $(g_2, g_2^a, g_2^b, g_1^c, T)$, 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 $\mathcal{A}_{\text{DBDH-2}}$ sets up the environment for $\mathcal{A}_{\mathsf{NIKE}}$ and simulates all its queries properly.

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

$$t_A = \mathsf{Cham.Eval}(m_1; r_1) \quad \text{and} \quad t_B = \mathsf{Cham.Eval}(m_2; r_2).$$

Let us define a polynomial $p(t) = p_0 + p_1 t + p_2 t^2$ of degree 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 $\mathbb{Z}_p$. Then $\mathcal{A}_{\text{DBDH-2}}$ sets $\alpha = (g_1^c)^{p_0} g_1^{q_0}$, $\beta = (g_1^c)^{p_1} g_1^{q_1}$, $\gamma = (g_1^c)^{p_2} g_1^{q_2}$ 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 $p(t_A) = p(t_B) = 0$). Then $\mathcal{A}_{\text{DBDH-2}}$ simulates all the queries of $\mathcal{A}_{\mathsf{NIKE}}$ as follows.

- RegisterHonest: When $\mathcal{A}_{\text{DBDH-2}}$ receives as input a RegisterHonest user query from $\mathcal{A}_{\mathsf{NIKE}}$ for a party with identity ID, it fist checks whether $\mathrm{ID} \in \{\mathrm{ID}_A, \mathrm{ID}_B\}$. Depending upon the result, it does the following:
  - If $\mathrm{ID} \notin \{\mathrm{ID}_A, \mathrm{ID}_B\}$, $\mathcal{A}_{\text{DBDH-2}}$ runs NIKE.gen to generate a pair of keys (pk, sk) and returns pk to $\mathcal{A}_{\mathsf{NIKE}}$.
  - If $\mathrm{ID} \in \{\mathrm{ID}_A, \mathrm{ID}_B\}$, $\mathcal{A}_{\text{DBDH-2}}$ does the following. Without loss of generality, let $\mathrm{ID} = \mathrm{ID}_A$. Now $\mathcal{A}_{\text{DBDH-2}}$ uses the trapdoor ck of the chameleon hash to produce $r'_A \in \mathcal{R}_{\mathrm{cham}}$ such that

    $$\mathsf{Cham.Eval}(g_2^a \| \mathrm{ID}_A; r'_A) = \mathsf{Cham.Eval}(m_1; r_1).$$

    Note that, by the random trapdoor collision property of the chameleon hash function, r'_A is uniformly distributed over $\mathcal{R}_{\mathrm{cham}}$ and also independent of r_1. Similarly, when $\mathrm{ID} = \mathrm{ID}_B$, $\mathcal{A}_{\text{DBDH-2}}$ uses the trapdoor ck to produce $r'_B \in \mathcal{R}_{\mathrm{cham}}$ such that $\mathsf{Cham.Eval}(g_2^b \| \mathrm{ID}_B; r'_B) = \mathsf{Cham.Eval}(m_2; r_2)$. The value r'_B is also uniformly distributed over $\mathcal{R}_{\mathrm{cham}}$ and also independent of r_2. $\mathcal{A}_{\text{DBDH-2}}$ then sets

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

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

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

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

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

This completes the description of simulation by $\mathcal{A}_{\text{DBDH-2}}$. If $\mathcal{A}_{\text{NIKE}}$ can distinguish between real and random key in Game 4, then this 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 $shk_{\text{ID}_A,\text{ID}_B} = e((g_1^c)^b, Z_A) = e((g_1^c)^a, Z_B) = e(g_1, g_2)^{abc}$.

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

$$\text{Adv}_{\text{Game}_6}(\mathcal{A}) = \text{Adv}_{\text{Game}_5}(\mathcal{A}).$$

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

$$|\text{Adv}_{\text{Game}_7}(\mathcal{A}) - \text{Adv}_{\text{Game}_6}(\mathcal{A})| \leq \text{Adv}^{\text{dbdh-2}}_{\mathcal{A}_{\text{DBDH-2}},\mathcal{G}_2}(\kappa).$$

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

$$\begin{aligned}
\text{Adv}^{\text{BLR-CKS-heavy}}_{\text{BLR-NIKE},\mathcal{A}_{\text{NIKE}}}(\kappa) &= \text{Adv}_{G_0}(\mathcal{A}) \\
&\leq q_{\text{H}}^2(\text{Adv}_{G_1}(\mathcal{A})) \\
&\leq q_{\text{H}}^2(\text{Adv}_{G_2}(\mathcal{A}) + 2\varepsilon) \\
&\leq q_{\text{H}}^2(\text{Adv}_{G_3}(\mathcal{A}) + 2\varepsilon + \varepsilon_{\text{prf}} + \varepsilon'_{\text{prf}}) \\
&= q_{\text{H}}^2(\text{Adv}_{G_4}(\mathcal{A}) + 2\varepsilon + \varepsilon_{\text{prf}} + \varepsilon'_{\text{prf}}) \\
&\leq q_{\text{H}}^2(\text{Adv}_{G_5}(\mathcal{A}) + 2\varepsilon + \varepsilon_{\text{prf}} + \varepsilon'_{\text{prf}} + \text{Adv}^{\text{coll}}_{\mathcal{A},\mathsf{ChamH}}(\kappa))
\end{aligned}$$

$$
\begin{aligned}
&= q_{\mathrm{H}}^2(\mathrm{Adv}_{G_6}(\mathcal{A}) + 2\varepsilon + \varepsilon_{\mathrm{prf}} + \varepsilon'_{\mathrm{prf}} + \mathrm{Adv}^{\mathrm{coll}}_{\mathcal{A},\mathsf{ChamH}}(\kappa)) \\
&\le q_{\mathrm{H}}^2(\mathrm{Adv}_{G_7}(\mathcal{A}) + 2\varepsilon + \varepsilon_{\mathrm{prf}} + \varepsilon'_{\mathrm{prf}} + \mathrm{Adv}^{\mathrm{coll}}_{\mathcal{A},\mathsf{ChamH}}(\kappa) + \mathrm{Adv}^{\text{dbdh-2}}_{\mathcal{A}_{\text{DBDH-2}},\mathcal{G}_2}(\kappa)) \\
&\le q_{\mathrm{H}}^2(2\varepsilon + \varepsilon_{\mathrm{prf}} + \varepsilon'_{\mathrm{prf}} + \mathrm{Adv}^{\mathrm{coll}}_{\mathcal{A},\mathsf{ChamH}}(\kappa) + \mathrm{Adv}^{\text{dbdh-2}}_{\mathcal{A}_{\text{DBDH-2}},\mathcal{G}_2}(\kappa)).
\end{aligned}
$$

Thus we have

$$
\mathrm{Adv}^{\text{BLR-CKS-heavy}}_{\text{BLR-NIKE},\mathcal{A}_{\text{NIKE}}}(\kappa) \le q_{\mathrm{H}}^2(2\varepsilon + \varepsilon_{\mathrm{prf}} + \varepsilon'_{\mathrm{prf}} + \mathrm{Adv}^{\mathrm{coll}}_{\mathcal{A},\mathsf{ChamH}}(\kappa) + \mathrm{Adv}^{\text{dbdh-2}}_{\mathcal{A}_{\text{DBDH-2}},\mathcal{G}_2}(\kappa)). \qquad \square
$$

4.2.2 Leakage tolerance of protocol BLR-NIKE

The order of the groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ is p. Note that, although the secret key in our protocol BLR-NIKE may appear to be a single field element, 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 BLR-CHF of Wang and Tanaka [39] consists of n field elements which also corresponds to the number of generators in the construction of [39]. The leakage tolerance of BLR-CHF is $\lambda' = ((n-1)\log(p) - \omega(\log\kappa))$ as shown in [39]. The size of the secret key of the BLR-CHF is $L = n\log p$. So, for sufficiently large n, the leakage rate $\eta = \frac{\lambda'}{L}$ of BLR-CHF approaches $1 - o(1)$. We also consider a very good randomness extractor that can work with inputs that have min-entropy $\nu \ll \log p$ 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 $\log p - \nu \approx \log p$. 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 $\eta' = \frac{\lambda}{L'} = 1 - o(1)$ 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 an LR-IND-CCA-2-secure PKE scheme from a BLR-CKS-heavy-secure LR-NIKE scheme. Actually, we show how to construct an 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.

5.1.1 Leakage-resilient chosen-ciphertext security for KEM

We say a KEM

$$
\Gamma = (\mathsf{KEM.Setup}, \mathsf{KEM.Gen}, \mathsf{KEM.Encap}, \mathsf{KEM.Decap})
$$

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

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

Oracle $\text{Decap}^*(C)$
if $C = C^*$, abort;
return $\text{Decap}(sk, C)$

Oracle $O^\lambda_{sk}(f)$
if $L + \lvert f(sk)\rvert \le \lambda(\kappa)$,
return $f(sk)$

Table 4: Experiment defining LR-CCA-2 security of KEM.

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

(i) KEM.Setup(1^κ): This algorithm runs the NIKEcommon_setup(1^κ) algorithm to obtain the system parameters *params*. It then sets the public parameters of the KEM *pub* as *params*.

(ii) KEM.Gen(1^κ, *pub*): This algorithm runs the algorithm NIKEgen(1^κ, *pub*) to obtain a pair of public-private keys (pk, sk). It then sets $pk_{\text{KEM}} = pk$ and $sk_{\text{KEM}} = sk$.

(iii) KEM.Encap(pk_{KEM}): This algorithm parses pk_{KEM} as pk and samples another key pair $(pk', sk') \leftarrow \text{NIKEgen}(1^\kappa, pub)$. Then it runs NIKEkey($pk$, sk') to produce the shared key *shk*. It then sets the encapsulated key $K = shk$ and the ciphertext $C = pk'$.

(iv) KEM.Decap(sk_{KEM}, C): This algorithm parses the ciphertext C as pk' and the secret key sk_{KEM} as sk. It then runs NIKEkey(pk', sk) and obtains the result.

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

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 5.1 (BLR-IND-CCA-2 security). Let $\kappa \in \mathbb{N}$ and $\lambda = \lambda(\kappa)$ be parameters. We say a KEM

$$\Gamma = (\text{Setup}, \text{Gen}, \text{Encap}, \text{Decap})$$

is λ-BLR-CCA-2-secure if, for all PPT adversaries A, there exists a negligible $\nu(\kappa)$ such that

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

where the experiment $\mathbf{Exp}^{\text{BLR-IND-CCA-2}}_{\Gamma,\mathsf{A}}(\kappa,\lambda)$ is defined in Table 4.

5.1.2 Generic construction of leakage-resilient KEM

We now show the construction of a leakage-resilient CCA-2-secure KEM

$$\Gamma = (\text{KEM.Setup}, \text{KEM.Gen}, \text{KEM.Encap}, \text{KEM.Decap})$$

from leakage-resilient NIKE (Figure 1).

Theorem 5.2. *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} *decapsulation queries and* q_{L} *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} RegisterCorrupt *queries,* q_{D} CorruptReveal *queries and* q_{L} Leakage *queries, and the running time of* $\mathcal{A}_{\text{NIKE}}$ *is roughly the same as that of* $\mathcal{A}_{\text{KEM}}$. *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 of 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 $sk_{\text{KEM}} = \perp$.
- KEM.Encap(pk): To simulate the challenge phase, $\mathcal{A}_{\text{NIKE}}$ makes a Test(pk_1, pk_2) query. It receives as reply a shared key K which is either the real key, i.e., $K = \text{NIKE.key}(pk_1, sk_2)$, 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 $C \neq C^*$, 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 that the above simulation is perfect. Note that if K^* is the real shared key, i.e., it is the output of the NIKE.key algorithm, then it is properly simulating the Encap algorithm in the $\mathbf{Exp}^{\text{BLR-IND-CCA-2}}_{\Gamma,\text{A}}(\kappa, \lambda)$ 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 the same as the number of decapsulation queries asked by $\mathcal{A}_{\text{KEM}}$. This completes the proof of the above theorem. □

Remark 5.3. Here we show an alternative simple construction of a BLR-IND-CCA-2-secure KEM from a standard IND-CCA-2 secure KEM given access to a leak-free hardware component that can store the seed of a randomness extractor (as in our case). Namely, let π = (Gen, Encap, Decap) be a (standard) IND-CCA-2-secure KEM. We construct a BLR-IND-CCA-2-secure KEM Γ = (KEM.Setup, KEM.Gen, KEM.Encap, KEM.Decap) from π as follows. KEM.Setup(1^κ): Sample a random string r', and compute $r = \text{Ext}(r', s)$, where s is the random seed of the extractor Ext. Store the seed s in the leak-free hardware component. Then run $pk = \text{Gen}(1^\kappa, r)$ to obtain the public key pk. Set the secret key $sk = r'$. The above scheme Γ achieves BLR-IND-CCA-2 security as long as the string r' is sufficiently long and the seed s is kept out of the view of the adversary. The encapsulation and the decapsulation functions remain unchanged from the underlying KEM scheme π. Although this simple solution works, we stress the objective of our work is to show the applications of LR-NIKE as a central unifying to construct many leakage-resilient primitives. We thank an anonymous reviewer of the journal *Design, Codes and Cryptography* for suggesting the above construction.

5.2 Leakage-resilient authenticated key exchange

The work of Bergsma et al. [8] shows a generic construction of an 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 3). 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 [26], 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. [8] in the bounded-memory leakage model.

There are numerous leakage versions of the eCK model, under the OCLI axiom [3–5] and the memory leakage model [11]. Further, they address after-the-fact leakage. With our new BLR-CKS-heavy-secure NIKE protocol, following the Bergsma et al. [8] transformation, we can achieve leakage-resilient AKE in an eCK-style model

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

5.2.1 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 [28], addressing before-the-fact, bounded-memory leakage.

Let κ be the security parameter. Let $\mathcal{U} = \{U_1, \ldots, U_n\}$ 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 the 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:

(i) both $\Pi^s_{U,V}$ and $\Pi^{s'}_{U',V'}$ have computed session keys;
(ii) messages sent from $\Pi^s_{U,V}$ and messages received by $\Pi^{s'}_{U',V'}$ are identical;
(iii) messages sent from $\Pi^{s'}_{U',V'}$ and messages received by $\Pi^s_{U,V}$ are identical;
(iv) $U' = V$ and $V' = U$;
(v) 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.

Modeling leakage: We consider the bounded-memory leakage setting for modeling 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 $\Pi^s_{U,V}$.
- $\mathsf{Corrupt}(U)$ 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 conditions (1)–(3) of Alawatugoda et al. [4, Definition 4] hold, and

(iv) 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, $\sum_{i=1}|f_i(sk_V)| \leq \lambda$;
(v) after $\Pi^s_{U,V}$ is activated, no leakage is allowed from U and V.

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

Table 5: Leakage-resilient AKE protocol LR-AKE.

BBFL-eCK security game: The adversary $\mathcal{A}$ interacts with the challenger by issuing any combination of Send(), SessionKeyReveal(), EphemeralKeyReveal(), Leakage() and Corrupt() queries at will. At some point, the adversary chooses a λ-BBFL-eCK-fresh oracle and issues a Test() query. Then the adversary may continue asking the Send(), SessionKeyReveal(), EphemeralKeyReveal(), Leakage() and 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 $\text{Succ}_{\mathcal{A}}$ denote the event that the adversary $\mathcal{A}$ wins the above security game.

Definition 5.4 (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 $\text{Adv}^{\text{BBFL-eCK}}_{\pi,\mathcal{A}}(\kappa) = |2\Pr(\text{Succ}_{\mathcal{A}}) - 1|$.

5.2.2 Constructing BBFL-eCK-secure key exchange protocols

In Table 5, we show the generic leakage-resilient variant of the Bergsma et al. [8] 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 LR-NIKE = (NIKEcommon_setup, NIKEgen, NIKEkey) be the underlying BLR-CKS-heavy-secure NIKE protocol. Let LR-SIG = (SIGkg, SIGsign, SIGvfy) be the underlying UF-CMLA-secure signature scheme, and let PRF be a secure pseudo-random function. Since the generic construction of the AKE protocol remains unchanged with respect to Bergsma et al. [8], 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 [8, Appendix A, Theorem 1] and its proof.

Theorem 5.5. *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 protocol sessions. Let $\mathcal{A}$ be an adversary against the above protocol LR-AKE*. We construct attackers $\mathcal{B}_{\text{sig}}$, $\mathcal{B}^{(1)}_{\text{nike}}$, $\mathcal{B}^{(0)}_{\text{nike}}$ and $\mathcal{B}_{\text{prf}}$ against the underlying leakage-resilient signature scheme, the leakage-resilient NIKE protocol (matching session exists and no matching session*

exists, respectively) and the pseudo-random function such that

$$\mathrm{Adv}^{\text{BBFL-eCK}}_{\text{LR-AKE},\mathcal{A}}(\kappa) \le 4d^2\ell^2\big(\mathrm{Adv}^{\text{BLR-CKS-heavy}}_{\text{LR-NIKE},\mathcal{B}^{(1)}_{\text{nike}}}(\kappa) + \mathrm{Adv}^{\text{prf}}_{\text{PRF},\mathcal{B}_{\text{prf}}}(\kappa)\big) + 4\,\mathrm{Adv}^{\text{BLR-CKS-heavy}}_{\text{LR-NIKE},\mathcal{B}^{(0)}_{\text{nike}}}(\kappa) + 4d\,\mathrm{Adv}^{\text{UF-CMLA}}_{\text{LR-SIG},\mathcal{B}_{\text{sig}}}(\kappa).$$

Proof sketch. To prove Theorem 5.5, we need to consider four types of attackers.

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

An 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 an 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. □

Remark 5.6. Our construction of an BBFL-eCK secure AKE protocol is obtained by replacing the building blocks (i.e., NIKE and one-time signature schemes) in Bergsma et al.'s framework [8] with a (bounded) leakage-resilient NIKE and a (bounded) leakage-resilient signature scheme. However, such a straightforward replacement of the underlying primitives with their corresponding leakage-resilient counterparts may encounter a subtle technical problem: the adversary can use the leakage oracle to break the authentication mechanism in the "test" session (which corresponds to breaking the UF-CMLA security of the signature scheme), or it can encode the (description of the) session key derivation function in the leakage function to leak from the session key. However, it is to be noted that, in our BBFL-eCK security model, the adversary is not allowed to query the leakage oracle during or after the test session. This is because we consider the setting of "before-the-fact" leakage in our current work. Hence the above impossibility result can be avoided in our setting.

5.2.3 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. [26] can tolerate $n - n^{\varepsilon}$ leakage, for an n-bit key and $1 > \varepsilon > 0$, 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.

5.3.1 Bounded-memory leakage LLKE-ma model

We refer to the security model under mutual authentication of Hale et al. [23, Section 5] as LLKE-ma model. In this work, we introduce a bounded-memory leakage model on top of the 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 the number of servers. Each client is represented by a collection of n oracles $\mathsf{C}_{i,1}, \ldots, \mathsf{C}_{i,n}$, and each server is represented by a collection of k oracles $\mathsf{S}_{j,1}, \ldots, \mathsf{S}_{j,k}$. Each oracle represents an instance of the protocol. Each principal has a long-term key pair (sk_i, pk_i). Let κ be the security parameter and λ the leakage parameter. Each oracle $\mathsf{C}_{i,s} \in [d] \times [n]$ (or $\mathsf{S}_{j,t} \in [\ell] \times [k]$, respectively) maintains

(i) two variables k_i^{tmp} and k_i^{main} to store the temporary and main keys of a session,
(ii) a variable $\mathsf{Partner}_i$ containing the identity of the intended communication partner,
(iii) variables $\mathcal{M}_{i,s}^{\mathrm{in}}$ and $\mathcal{M}_{i,s}^{\mathrm{out}}$ containing messages sent and received by the oracle.

Adversarial powers:

- $\mathsf{Send}(\mathsf{C}_{i,s}/\mathsf{S}_{j,t}, m)$: 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.
- $\mathsf{Reveal}(\mathsf{C}_{i,s}/S_{j,t}, \mathrm{tmp/main})$: This query returns the key of the given stage if it has been already computed, or $\perp$ otherwise.
- $\mathsf{Corrupt}(i/j)$: This query returns the long-term secret key of the server or the client accordingly. If $\mathsf{Corrupt}(j/i)$ 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 $\tau := \infty$.
- $\mathsf{Test}(C_{i,s}/S_{j,t}, \mathrm{tmp/main})$: 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.
- $\mathsf{Leakage}(i/j, f_i)$: The leakage $f_i(sk_{i/j})$ is computed and returns to the adversary iff $\sum_{i=1} |f_i(sk_{i/j})| \leq \lambda$.

BL-LLKE-*ma* security game: The adversary interacts with the challenger by issuing any combination of Send(), Corrupt(), Reveal() and Leak() queries. At some point, the adversary issues a Test() query to an oracle that holds the conditions in Definition 5.7. Then the adversary may continue asking the Send(), Corrupt(), Reveal() and Leak() queries, without violating the conditions of Definition 5.7, and finally outputs answer bit b' for the challenge. $\mathcal{A}$ wins if $b' = b$. Let $\mathrm{Succ}_{\mathcal{A}}$ denote the event that the adversary $\mathcal{A}$ wins the above security game.

Definition 5.7 (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 in [23, Definition 8].
- Before activation of the test session on C_i, for all $\mathsf{Leakage}(i, f_i)$ queries, $\sum_{i=1}|f_i(sk_i)| \le \lambda$, and before activation of the test session on S_j, for all $\mathsf{Leakage}(j, f_i)$ queries, $\sum_{i=1}|f_i(sk_j)| \le \lambda$.
- After activation of the Test session on C_i, no leakage is allowed from sk_i (same as to the case of S_j).

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

5.3.2 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 a 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, and let $\mathsf{LR\text{-}SIG} = (\mathsf{SIGkg}, \mathsf{SIGsign}, \mathsf{SIGvfy})$ be a UF-CMLA-secure signature scheme. Then we construct a LLKE protocol $\mathsf{LR\text{-}LLKE} = (\mathsf{Gen}, \mathsf{KE}^{\text{client}}_{\text{init}}, \mathsf{KE}^{\text{client}}_{\text{refresh}}, \mathsf{KE}^{\text{server}}_{\text{refresh}})$ same as the description in Hale et al. [23, Section 6.1].

Since the generic construction of the LLKE protocol remains unchanged with respect to Hale et al., except for 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 are similar to [23, Appendix 6.2, Theorem 2] and its proof.

Theorem 5.8. *If the underlying NIKE protocol* $\mathsf{LR\text{-}NIKE}$ *is* BLR-CKS-heavy-*secure and 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 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 $\mathcal{B}_{\text{sig}}$ and $\mathcal{B}_{\text{nike}}$ against the underlying leakage-resilient signature scheme and the leakage-resilient NIKE protocol such that*

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

Proof sketch. We distinguish between four different attackers:

- an A1-type attacker asks the Test query to a client oracle and the temporary key;
- an A2-type attacker asks the Test query to a client oracle and the main key;
- an A3-type attacker asks the Test query to a server oracle and the temporary key;
- an A4-type attacker asks the 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 5.8 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 the same as that of Hale et al. [23]. □

5.3.3 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. [26] can tolerate $n - n^{\varepsilon}$ leakage, for an n-bit key and $1 > \varepsilon > 0$, 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

Our work provides a new direction for constructing several leakage-resilient cryptographic primitives, such as leakage-resilient PKE schemes, AKE protocols and LLKE protocols, using 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 also 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:

- construction of leakage-resilient NIKE in the $(1 - o(1))$-bounded-memory leakage model, without the leak-free hardware assumption,
- leakage-resilient NIKE in the $(1 - o(1))$-continuous memory leakage model in the non-split state model.

A Additional preliminaries

A.1 Basics of information theory

Definition A.1 (Min-entropy). The min-entropy of a random variable X, denoted as $\mathrm{H}_\infty(\mathrm{X})$, is defined as $\mathrm{H}_\infty(\mathrm{X}) \stackrel{def}{=} -\log(\max_x \Pr[X = x])$. This is a standard notion of entropy used in cryptography since it measures the worst-case predictability of X.

Definition A.2 (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

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

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 by Dodis et al. [17].

Lemma A.3 ([17]). *For any random variables X, Y and Z, if Y takes on values in $\{0,1\}^\ell$, then*

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

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 [39].

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 $\mathsf{ChamH}\colon \mathcal{D} \times \mathcal{R}_{\mathrm{cham}} \to \mathcal{I}$, where $\mathcal{D}$ is the domain, $\mathcal{R}_{\mathrm{cham}}$ the randomness space and $\mathcal{I}$ the range, consists of the algorithms ($\mathsf{Cham.KeyGen}$, $\mathsf{Cham.Eval}$, $\mathsf{Cham.TCF}$).

(i) $\mathsf{Cham.KeyGen}(1^\kappa, \lambda)$: The key generation algorithm takes as input 1^κ and the leakage bound λ as parameters and outputs an evaluation key along with a trapdoor (hk, ck). The public key hk defines a chameleon hash function, denoted $\mathsf{ChamH}_{hk}(\cdot,\cdot)$.

(ii) $\mathsf{Cham.Eval}(hk, m, r)$: The hash function evaluation algorithm that takes as input hk, a message $m \in \mathcal{D}$ and a randomizer $r \in \mathcal{R}_{\mathrm{cham}}$, outputs a hash value $h = \mathsf{ChamH}_{hk}(m, r)$.

(iii) $\mathsf{Cham.TCF}(ck, (m, r), m')$: The trapdoor collision finder algorithm takes as the trapdoor ck, a message-randomizer pair (m, r), an additional message m', and outputs a value $r' \in \mathcal{R}_{\text{cham}}$ such that

$$\mathsf{ChamH}_{hk}(m, r) = \mathsf{ChamH}_{hk}(m', r').$$

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

- *Reversibility*: The reversibility property is satisfied if $r' = \mathsf{Cham.TCF}(ck, (m, r), m')$ is equivalent to

$$r = \mathsf{Cham.TCF}(ck, (m', r'), m).$$

- *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, we have that $r' = \mathsf{Cham.TCF}(ck, (m, r), m')$ has uniform probability distribution on the randomness space $\mathcal{R}_{\text{cham}}$.
- *LR-collision resistance*: The LR collision-resistance property is satisfied if, for any PPT adversary $\mathcal{A}$, the following advantage in negligible:

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

where $O^{\kappa,\lambda}_{ck}$ 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

$F\colon \Sigma^k \times \Sigma^m \to \Sigma^n$ is a $(\varepsilon_{\text{prf}}, s_{\text{prf}}, q_{\text{prf}})$ 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

$$\Big|\Pr_{K \xleftarrow{\$} \Sigma^K}[\mathcal{A}^{F(K,\cdot)} \to 1] - \Pr_{R_{m,n}}[\mathcal{A}^{R_{m,n}(\cdot)} \to 1]\Big| \leq \varepsilon_{\text{prf}},$$

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. [26]. 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 A.4 (Unforgeability against chosen message leakage attacks (UF-CMLA)). Let κ be the security parameter and λ the leakage parameter. Let $\mathsf{LR\text{-}SIG} = (\mathsf{SIGkg}, \mathsf{SIGsign}, \mathsf{SIGvfy})$ be a signature scheme. We define the advantage $\mathrm{Adv}^{\text{UF-CMLA}}_{\mathsf{LR\text{-}SIG},\mathcal{B}_{\text{sig}}}(\kappa)$ of any PPT adversary $\mathcal{B}_{\text{sig}}$ winning the following game:

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

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

(iii) If $\mathsf{SIGvfy}(pk^{\text{sig}}, m^*, \sigma^*)$ = "true" and m^* is not been previously signed, then $\mathcal{B}_{\text{sig}}$ 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 $\mathsf{LR\text{-}SIG}$ is *UF-CMLA*-secure if $\mathrm{Adv}^{\text{UF-CMLA}}_{\mathsf{LR\text{-}SIG},\mathcal{B}_{\text{sig}}}(\kappa)$ is negligible.

Katz et al. [26] constructed a UFCMLA-secure signature scheme in bounded leakage model in which $n = 1$. It contains signing and verification operations based on NIZK proofs, where the signature can be generated with a cost of two exponentiations and verified with a cost of four 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 notion of *leakage-resilient* cryptography, introducing various leakage models [2, 6, 29, 30].

In the work of Micali and Reyzin [30], a general framework was introduced to model the leakage that occurs during computation with secret parameters, which is widely known as the *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 only occurs 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 lengths 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. [9] and Dodis et al. [14] 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 are the 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. [16] 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 enforces the hardness parameter instead.

C Vulnerability of the NIKE protocol of [19] in the bounded-leakage setting

In this section, we show that the NIKE protocol of Freire et al. [19] 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. [19] 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 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 a *key recovery attack*, which is stronger than the attack on the indistinguishability game.

Funding: Janaka would like to acknowledge the university research grant URG/2018/19/E of the University of Peradeniya, Sri Lanka. The authors would like to acknowledge the support of the Indian Institute of Technology Madras and the University of Peradeniya for this collaborative work.

References

[1] S. Agrawal, Y. Dodis, V. Vaikuntanathan and D. Wichs, On continual leakage of discrete log representations, in: *Advances in Cryptology—ASIACRYPT 2013. Part II*, Lecture Notes in Comput. Sci. 8270, Springer, Heidelberg (2013), 401–420.

[2] A. Akavia, S. Goldwasser and V. Vaikuntanathan, Simultaneous hardcore bits and cryptography against memory attacks, in: *Theory of Cryptography*, Lecture Notes in Comput. Sci. 5444, Springer, Berlin (2009), 474–495.

[3] J. Alawatugoda, C. Boyd and D. Stebila, Continuous after-the-fact leakage-resilient key exchange, in: *Information Security and Privacy—ACISP 2014*, Lecture Notes in Comput. Sci. 8544, Springer, Cham (2014), 258–273.

[4] J. Alawatugoda, D. Stebila and C. Boyd, Modelling after-the-fact leakage for key exchange, in: *Proceedings of the 9th ACM Symposium on Information, Computer and Communications Security—ASIACCS 2014*, ACM, New York (2014), 207–216.

[5] J. Alawatugoda, D. Stebila and C. Boyd, Continuous after-the-fact leakage resilient eCK-secure key exchange, in: *Cryptography and Coding*, Lecture Notes in Comput. Sci. 9496, Springer, Cham (2015), 277–294.

[6] J. Alwen, Y. Dodis and D. Wichs, Leakage-resilient public-key cryptography in the bounded-retrieval model, in: *Advances in Cryptology—CRYPTO 2009*, Lecture Notes in Comput. Sci. 5677, Springer, Berlin (2009), 36–54.

[7] F. Benhamouda, G. Couteau, D. Pointcheval and H. Wee, Implicit zero-knowledge arguments and applications to the malicious setting, in: *Advances in Cryptology—CRYPTO 2015. Part II*, Lecture Notes in Comput. Sci. 9216, Springer, Heidelberg (2015), 107–129.

[8] F. Bergsma, T. Jager and J. Schwenk, One-round key exchange with strong security: an efficient and generic construction in the standard model, in: *Public-Key Cryptography—PKC 2015*, Lecture Notes in Comput. Sci. 9020, Springer, Heidelberg (2015), 477–494.

[9] Z. Brakerski, Y. T. Kalai, J. Katz and V. Vaikuntanathan, Overcoming the hole in the bucket: public-key cryptography resilient to continual memory leakage, in: *2010 IEEE 51st Annual Symposium on Foundations of Computer Science—FOCS 2010*, IEEE Computer Soc., Los Alamitos, CA (2010), 501–510.

[10] S. Chakraborty, J. Alawatugoda and C. P. Rangan, Leakage-resilient non-interactive key exchange in the continuous-memory leakage setting, in: *Provable Security*, Lecture Notes in Comput. Sci. 10592, Springer, Cham (2017), 167–187.

[11] R. Chen, Y. Mu, G. Yang, W. Susilo and F. Guo, Strongly leakage-resilient authenticated key exchange, in: *Topics in Cryptology—CT-RSA 2016*, Lecture Notes in Comput. Sci. 9610, Springer, Cham (2016), 19–36.

[12] R. Chen, Y. Mu, G. Yang, W. Susilo and F. Guo, Strong authenticated key exchange with auxiliary inputs, *Des. Codes Cryptogr.* **85** (2017), no. 1, 145–173.

[13] W. Diffie and M. E. Hellman, New directions in cryptography, *IEEE Trans. Inform. Theory* **IT-22** (1976), no. 6, 644–654.

[14] Y. Dodis, K. Haralambiev, A. López-Alt and D. Wichs, Cryptography against continuous memory attacks, in: *2010 IEEE 51st Annual Symposium on Foundations of Computer Science—FOCS 2010*, IEEE Computer Soc., Los Alamitos, CA (2010), 511–520.

[15] Y. Dodis, K. Haralambiev, A. López-Alt and D. Wichs, Efficient public-key cryptography in the presence of key leakage, in: *Advances in Cryptology—ASIACRYPT 2010*, Lecture Notes in Comput. Sci. 6477, Springer, Berlin (2010), 613–631.

[16] Y. Dodis, Y. T. Kalai and S. Lovett, On cryptography with auxiliary input, in: *STOC'09—Proceedings of the 2009 ACM International Symposium on Theory of Computing*, ACM, New York (2009), 621–630.

[17] Y. Dodis, R. Ostrovsky, L. Reyzin and A. Smith, Fuzzy extractors: How to generate strong keys from biometrics and other noisy data, *SIAM J. Comput.* **38** (2008), no. 1, 97–139.

[18] S. Dziembowski and S. Faust, Leakage-resilient circuits without computational assumptions, in: *Theory of Cryptography Conference*, Springer, Berlin (2012), 230–247.

[19] E. S. V. Freire, D. Hofheinz, E. Kiltz and K. G. Paterson, Non-interactive key exchange, in: *Public-Key Cryptography—PKC 2013*, Lecture Notes in Comput. Sci. 7778, Springer, Heidelberg (2013), 254–271.

[20] A. Fujioka, K. Suzuki, K. Xagawa and K. Yoneyama, Strongly secure authenticated key exchange from factoring, codes, and lattices, *Des. Codes Cryptogr.* **76** (2015), no. 3, 469–504.

[21] D. Galindo, Boneh-Franklin identity based encryption revisited, in: *Automata, Languages and Programming*, Lecture Notes in Comput. Sci. 3580, Springer, Berlin (2005), 791–802.

[22] S. Goldwasser and G. N. Rothblum, Securing computation against continuous leakage, in: *Advances in Cryptology—CRYPTO 2010*, Lecture Notes in Comput. Sci. 6223, Springer, Berlin (2010), 59–79.
[23] B. Hale, T. Jager, S. Lauer and J. Schwenk, Speeding: On low-latency key exchange, preprint (2015), https://eprint.iacr.org/2015/1214.
[24] S. Halevi and H. Lin, After-the-fact leakage in public-key encryption, in: *Theory of Cryptography*, Lecture Notes in Comput. Sci. 6597, Springer, Heidelberg (2011), 107–124.
[25] A. Juma and Y. Vahlis, Protecting cryptographic keys against continual leakage, in: *Advances in Cryptology—CRYPTO 2010*, Lecture Notes in Comput. Sci. 6223, Springer, Berlin (2010), 41–58.
[26] J. Katz and V. Vaikuntanathan, Signature schemes with bounded leakage resilience, in: *Advances in Cryptology—ASIACRYPT 2009*, Lecture Notes in Comput. Sci. 5912, Springer, Berlin (2009), 703–720.
[27] E. Kiltz and K. Pietrzak, Leakage resilient ElGamal encryption, in: *Advances in Cryptology—ASIACRYPT 2010*, Lecture Notes in Comput. Sci. 6477, Springer, Berlin (2010), 595–612.
[28] B. LaMacchia, K. Lauter and A. Mityagin, Stronger security of authenticated key exchange, in: *International Conference on Provable Security*, Springer, Berlin (2007), 1–16.
[29] T. Malkin, I. Teranishi, Y. Vahlis and M. Yung, Signatures resilient to continual leakage on memory and computation, in: *Theory of Cryptography*, Lecture Notes in Comput. Sci. 6597, Springer, Heidelberg (2011), 89–106.
[30] S. Micali and L. Reyzin, Physically observable cryptography (extended abstract), in: *Theory of Cryptography*, Lecture Notes in Comput. Sci. 2951, Springer, Berlin (2004), 278–296.
[31] H. Morita, J. C. N. Schuldt, T. Matsuda, G. Hanaoka and T. Iwata, On the security of non-interactive key exchange against related-key attacks, *IEICE Trans. Fundam. Electron. Comm. Comput. Sci.* **100** (2017), no. 9, 1910–1923.
[32] D. Moriyama and T. Okamoto, Leakage resilient eck-secure key exchange protocol without random oracles, in: *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security—ASIACCS 2011*, ACM, New York (2011), 441–447.
[33] M. Naor and G. Segev, Public-key cryptosystems resilient to key leakage, in: *Advances in Cryptology—CRYPTO 2009*, Lecture Notes in Comput. Sci. 5677, Springer, Berlin (2009), 18–35.
[34] N. Nisan and D. Zuckerman, Randomness is linear in space, *J. Comput. System Sci.* **52** (1996), no. 1, 43–52.
[35] B. Qin and S. 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*, Springer, Heidelberg (2013), 381–400.
[36] B. Qin and S. Liu, Leakage-flexible CCA-secure public-key encryption: Simple construction and free of pairing, in: *Public-Key Cryptography—PKC 2014*, Lecture Notes in Comput. Sci. 8383, Springer, Heidelberg (2014), 19–36.
[37] V. Shoup, OAEP reconsidered, *J. Cryptology* **15** (2002), no. 4, 223–249.
[38] V. Shoup, Sequences of games: A tool for taming complexity in security proofs, preprint (2004), https://eprint.iacr.org/2004/332.
[39] Y. Wang and K. Tanaka, Generic transformation to strongly existentially unforgeable signature schemes with leakage resiliency, in: *Provable Security*, Lecture Notes in Comput. Sci. 8782, Springer, Cham (2014), 117–129.
[40] J.-D. Wu, Y.-M. Tseng, S.-S. Huang and W.-C. Chou, Leakage-resilient certificateless key encapsulation scheme, *Informatica (Vilnius)* **29** (2018), no. 1, 125–155.