Forward Security under Leakage Resilience, Revisited

Suvradip Chakraborty*, Harish Karthikeyan**, Adam O'Neill***, and C. Pandu Rangan†

Abstract. As both notions employ the same key-evolution paradigm, Bellare *et al.* (CANS 2017) study combining forward security with leakage resilience. The idea is for forward security to serve as a hedge in case at some point the full key gets exposed from the leakage. In particular, Bellare *et al.* combine forward security with *continual* leakage resilience, dubbed FS+CL. Our first result improves on Bellare *et al.*'s FS+CL secure PKE scheme by building one from any continuous leakage-resilient binary-tree encryption (BTE) scheme; in contrast, Bellare *et al.* require extractable witness encryption. Our construction also preserves leakage rate of the underlying BTE scheme and hence, in combination with existing CL-secure BTE, yields the first FS+CL secure encryption scheme with optimal leakage rate from standard assumptions.

We next explore combining forward security with other notions of leakage resilience. Indeed, as argued by Dziembowski *et al.* (CRYPTO 2011), it is desirable to have a *deterministic* key-update procedure, which FS+CL does not allow for arguably pathological reasons. To address this, we combine forward security with *entropy-bounded* leakage (FS+EBL). We construct FS+EBL non-interactive key exchange (NIKE) with deterministic key update based on indistinguishability obfuscation ($i\mathcal{O}$), and DDH or LWE. To make the public keys constant size, we rely on the Superfluous Padding Assumption (SuPA) of Brzuska and Mittelbach (ePrint 2015) *without* auxiliary information, making it more plausible. SuPA notwithstanding, the scheme is also the first FS-secure NIKE from $i\mathcal{O}$ rather than multilinear maps. We advocate a future research agenda that uses FS+EBL as a hedge for FS+CL, whereby a scheme achieves the latter if key-update randomness is good and the former if not.

1 Introduction

1.1 Background and Motivation

Leakage Resilience. When a cryptographic algorithm is implemented and run, it must be done on some *physical* system. This introduces *side channel*

* Visa Research, Palo Alto, California, USA. `suvchakr@visa.com`

** J.P. Morgan AI Research, New York, USA. `harish.karthikeyan@jpmchase.com`. Work done while at New York University, New York, USA

*** Manning College of Information and Computer Sciences, University of Massachusetts Amherst, USA. `adamo@cs.umass.edu`

† Kotak Mahindra Visiting Chair Professor, IISc Bangalore, INDIA. `prangan55@gmail.com`. Research partially supported by KIAC research grant

attacks where the adversary obtains some leakage about secrets, via execution time, power consumption, and even sound waves [23,30,31]. The cryptographic community responded by extending the attack model so that the adversary gets some "bounded" leakage about the secrets [2,9,29,36]. Works further extended this new model to consider "continual" leakage (CL) attacks [10,14]. In that model, the life of a secret key is divided into time periods, and in time period $t+1$ one runs an update algorithm on the secret key of time period t to derive the new secret key for time period $t+1$. (The old secret key is erased.) In each time period, the adversary queries for a function with a bounded output length applied to the current secret key.

Forward Security. *Forward security* (FS) [5] employs the same key-evolution paradigm as CL to address the threat of exposure of the secret key *in whole*. This can happen due to too much leakage. If a break-in happens during time period i, it is required that security still holds relative to keys in time periods $1 \leq i' < i$. Initial work on forward security has been extended and optimized in numerous works, *e.g.* [1,7,8,21,22,27,32,35].

Combining Leakage Resilience and Forward Security. As advocated by Bellare *et al.* [6], one ought to use FS as a hedge in the context of leakage resilience. Specifically, one would like to "fall back" to forward security if the secret key at every time period up to some i' is partially leaked, but then in time period i' the leakage happens to be so much that this time period's entire key is revealed. This combination of forward security and continual leakage resilience was dubbed FS+CL by Bellare *et al.* [6]. For constructions, Bellare *et al.* start by examining tree-based constructs as in [5] and give such a construction of FS+CL signatures based on CL-signatures. They also provide a generic approach to construct FS+CL encryption and signature schemes by combining what they call a key-evolution scheme (KE) that is forward one-way under continual-leakage (FOWCL KE) with witness primitives, namely (extractable) witness encryption [20] and witness signatures [4,13] respectively.

1.2 Our Contributions in Brief

Our Goals. Extractable witness encryption is a suspect assumption [19] that we would like to eliminate. We would also like to improve the leakage rates of the Bellare *et al.*'s schemes. Finally, we would like to study complementary notions of leakage resilience in this context. In this paper, we focus on asymptotic efficiency and feasibility rather than practical efficiency. The design of more practical constructions is an interesting question for future work.

Improved FS+CL Encryption Scheme. We improve upon Bellare *et al.*'s FS+CL PKE scheme by carefully re-examining tree-based constructs. In particular, in their effort to construct FS+CL PKE, Bellare *et al.* explicitly dismissed the idea of using a CL-secure binary-tree encryption (BTE) [12], because the underlying hierarchical identity-based encryption scheme (HIBE) scheme must tolerate *joint* leakage on multiple keys, whereas CL-security only allows leakage on each such key *individually*. We show this intuition is *false* and construct an FS+CL-secure PKE scheme from any CL-secure BTE scheme. This, in turn, can be realized from any CL-secure HIBE. CL-secure HIBE is known from simple assumptions on composite-order bilinear groups [10]. We also show that our construction preserves the leakage rate of the base scheme. Hence, we obtain FS+CL encryption enjoying optimal leakage rate from standard assumptions.

Alternative Models: FS+(C)EBL. Dziembowski *et al.* [17] argue it is desirable to have a deterministic key-update procedure. Indeed, randomness generation in practice can be buggy, either subverted or by poor implementation. CL does not guarantee any security in such a case, as there is a trivial attack under it if key-update is determinitic: the adversary just leaks some future key bit-by-bit across time periods. Yet this attack is arguably contrived; if the key-update procedure is a complicated cryptographic operation, it's unlikely real-world leakage would compute it, let alone a non-noisy version of it. Accordingly, we seek meaningful security notion that can be achieved when key-update is deterministic.

At a high-level, we combine forward security with *entropic bounded leakage* (EBL) [36] instead of CL. To this end, we introduce a model called FS+EBL (pronounced *ee-bull*). The FS+EBL model is defined with respect to any key evolving scheme equipped with an update function. Our definition requires forward security in the presence of leakage such that the current key always meets an entropy bound. In particular, we require that the secret key in each time period prior to the period of exposure retain enough residual entropy conditioned on the leakage from each of the keys.

Such a restriction on key entropy seems overly severe, however, and motivates additional consideration of *computational entropy*. The point is that if considering information-theoretic entropy, leakage on the current key necessarily reduces the entropy of all other keys. But consider leaking (noisy) hamming weight or physical bits of the current key, or even some one-way function of the key. After an appropriate update function is applied, it is plausible that the computational entropy of the new key is restored. To profit in such a case, we introduce the FS+CEBL (pronounced *see-bull* — 'C' for computational) that parallels FS+EBL but uses computational entropy.

FS+EBL and FS+CEBL NIKE. Broadening our set of primitives considered, we study non-interactive key exchange (NIKE) in the FS+(C)EBL model. We give an FS+EBL-secure NIKE in the common reference string (CRS) model from indistinguishability obfuscation ($i\mathcal{O}$), either DDH or LWE and a relaxed variant of the *Superfluous Padding Assumption* (SuPA) on $i\mathcal{O}$ introduced in [11].

We remark that, before this work, even *FS*-NIKE was not known from $i\mathcal{O}$. Similar to the prior FS-NIKE construction from multilinear maps [37], our construction of FS+EBL NIKE supports an a-priori bounded (but an arbitrary polynomial) number of time periods. However, our construction achieves much better parameters than the construction of [37]. In particular, the size of the public parameter in [37] is $O(T)$, the secret key size is $O(\log T)$, and the public key size is constant (here T denotes the maximum number of time periods supported by the scheme). In contrast, our FS+EBL NIKE achieves constant-size secret keys and public parameters, and the size of our public keys is $O(\log T)$. Ours also enjoys an optimal leakage rate. Hence, relaxed SuPA notwithstanding, our construction improves on [37].

NIKE in FS+CEBL Model. A nice feature of our FS+EBL NIKE construction is that the key update function can be instantiated by any entropic-leakage resilient one-way function [9]. In the FS+CEBL setting, we suggest using the PRG of Zhandry [39], because it is secure for any computationally unpredictable seed. The issue is that leakage from time period i could leak from secret key $i+1$ which is the output of the PRG. Existing results do not explore the case where the output of the PRG is also susceptible to leakage. We leave constructing FS+CEBL NIKE for future work.

Discussion. A drawback of FS+EBL and FS+CEBL is that they are *scheme-dependent*. This is because the entropy bound is required to hold with respect to the *specific* update function of the underlying key evolving primitive. Thus, the meaning of these security models in practice remains somewehat unclear. Therefore, we raise the open question of devising a notion combining forward security and leakage resilience that (1) admits schemes with deterministic key update, and (2) is *not* scheme dependent. We leave this for future work. Importantly, we conjecture that such a model would *not* deem our FS+EBL NIKE scheme insecure, but rather admit an improved security proof for it. We view our result as a step towards resolution of the above question. Another direction we suggest for future work is to design FS+CL schemes that simultaneously meet FS+(C)EBL or another notion as a hedge when key-update randomness is subverted or buggy.

2 Technical Overview

High-level Idea of the FS+CL PKE. Recall that a binary-tree encryption (BTE) has a master public key (MPK) associated with the root node of a binary tree and all the nodes have an associated secret key. Moreover, the secret key of any node can be used to derive the secret keys for the children of that node. To encrypt a message for a particular node, one uses MPK and the identity of that node. The security notion requires the attacker to commit to a target node w^* in advance (i.e., before seeing MPK) and it gets the secret keys of all nodes except for those which lie on the path from the root to the "target" node (including both). Under CL, the adversary can also leak continuously from the secrets keys of all these nodes. The goal of the adversary is then to win the indistinguishability game with respect to the target node w^*.

To construct a FS+CL PKE scheme for $T \leq 2^\ell - 1$ time period, we use a continuous leakage-resilient BTE (CLR-BTE) scheme of depth ℓ and associate the time periods with all nodes of the tree according to a *pre-order* traversal. Let w^i denote the node corresponding to time period i. The public key of the FS+CL PKE scheme consists of the root public key MPK and the secret key for time period i consists of sk_{w^i} (the secret key of w^i) and the secret keys of all right siblings of the nodes on the path from the root to w^i. At the end of time period i the secret key is updated as follows: If w^i is an internal node, then the secret keys of node w^{i+1} (the next node according to the pre-order traversal) and its sibling (i.e., the two children of w^i) are derived; otherwise the secret key for w^{i+1} is already stored as part of the secret key. In either case, sk_{w^i} is erased. The secret keys of the all the nodes corresponding to time period $i+1$ are then refreshed by running the key update algorithm of the underlying CLR-BTE scheme.

Proof Strategy. In our proof, the reduction (which is an adversary $\mathcal{A}_{\mathsf{clr\text{-}bte}}$ of the underlying CLR-BTE scheme) simply guesses the time period i^* in which the FS+CL adversary $\mathcal{A}_{\mathsf{kee}}$ will attack.[1] This corresponds to a challenge node w^{i^*} which $\mathcal{A}_{\mathsf{clr\text{-}bte}}$ forwards to its own challenger. If the guess is incorrect, the reduction aborts outputting a random bit. $\mathcal{A}_{\mathsf{clr\text{-}bte}}$ then receives the secret keys of all the nodes that are right siblings of the nodes that lie in the path $P_{w^{i^*}}$ from the root node to w^{i^*} (the target node) and also the secret keys of both the children of w^{i^*}. Using the knowledge of these keys $\mathcal{A}_{\mathsf{clr\text{-}bte}}$ can simulate the update queries of $\mathcal{A}_{\mathsf{kee}}$. Now, let us see how to simulate the leakage queries of $\mathcal{A}_{\mathsf{kee}}$. Note that, the secret key corresponding to some time period i in the FS+CL scheme is of the form $SK_i = (sk_{w^i}, \{sk_{rs(P_{w^i})}\})$, where sk_{w^i} and $\{sk_{rs(P_{w^i})}\}$ denote the (possibly refreshed versions of the) secret keys corresponding to the node w^i and the right siblings of all the nodes that lie on the path P_{w^i} respectively. Now, either of the following two cases arise: (i) either the node w^i lies in the path $P_{w^{i^*}}$ (the path from root to the target node w^{i^*}) or (ii) w^i does not lie in the path $P_{w^{i^*}}$. In the first case, $\mathcal{A}_{\mathsf{clr\text{-}bte}}$ already knows all the keys $\{sk_{rs(P_{w^i})}\}$ and hence it can translate the leakage function f (queried by $\mathcal{A}_{\mathsf{kee}}$) to a related leakage function f' *only* on the key sk_{w^i} (by hard-wiring the keys $\{sk_{rs(P_{w^i})}\}$ into f'). For the later case, $\mathcal{A}_{\mathsf{clr\text{-}bte}}$ knows the key sk_{w^i} and all the keys $\{sk_{rs(P_{w^i})}\}$, except exactly one key corresponding to a node w that lies in the path $P_{w^{i^*}}$. So, $\mathcal{A}_{\mathsf{clr\text{-}bte}}$ can again translate the joint leakage function f to leakage just on the secret key of node w.

To summarize, the key observation is that: in either case, the reduction knows the secret keys of all nodes except one, and hence it can simulate the joint leakage by leaking only on one node at a time. However, the adversary may also get multiple (continuous) leakages on the secret key of a node. For e.g., consider the secret key sk_1 (corresponding to the right child of the root node). The secret key sk_1 is included in each secret key sk_{0w} for any suffix w. However, note that, when the secret keys from one time period are updated to the next time period they are also refreshed by running the underlying key refresh algorithm of the CLR-BTE scheme. Hence the CLR property of the BTE scheme allows us to tolerate multiple leakages on the same node by making use of its leakage oracle.

[1] We stress that our scheme supports an *exponential* number of time periods; however, the adversary can only run for a polynomial number of them. Hence we incur a polynomial security loss in making this guess.

Constructing NIKE in the FS+EBL Model. The starting point of our NIKE construction is the bounded leakage-resilient NIKE construction of [34] (henceforth called the *LMQW protocol*) from indistinguishability obfuscation ($i\mathcal{O}$) and other standard assumptions (DDH/LWE) in the CRS model.

The main idea of the LMQW construction is as follows: Each user samples a random string s as its secret key and sets its public key as $x = G(s)$, where G is a function whose description is a part of the CRS and can be indistinguishably created in either *lossy* or in *injective* mode. In the real construction, the function G is set to be injective. To generate a shared key with an user j, user i inputs its own key pair (x_i, s_i) and the public key x_j of user j to an obfuscated program $\widehat{C}$ (which is also included as part of the CRS) which works as follows: The circuit C (which is obfuscated) simply checks if s_i is a valid pre-image of either x_i or x_j under G, i.e, it checks if either $x_i = G(s_i)$ or $x_j = G(s_i)$. If so, it returns $\mathsf{PRF}_K(x_i, x_j)$ (where the PRF key K is embedded inside the obfuscated program); else it outputs $\perp$.

Lifting the LMQW protocol to the FS setting. It is easy to see that the LMQW protocol is *not* forward-secure. This is because, each public key is an injective function of its corresponding secret key, and hence if a secret key s is updated to s', the public key no longer stays the same. We now describe how to modify the above construction to achieve security in the FS+EBL setting. Similarly to the LMQW protocol, the (initial) secret key of each user i in our construction is also a random string $s_i^{(1)}$. The CRS also contains the description of the obfuscated program $\widehat{C}$ and the function G (as described above). However, the public key of each party i is now an obfuscated circuit $\widehat{C_i}$ (corresponding to a circuit C_i, whose size is determined later) which has the initial/root secret key $s_i^{(1)}$ (corresponding to base time period 1) of party i embedded in it. It takes as input a key $s_j^{(t)}$ of user j (corresponding to some time period t) and works as follows: (a) First, it updates the secret key $s_i^{(1)}$ of user i (hard-coded in it) to $s_i^{(t)}$ by running the (deterministic) NIKE update function (to be defined shortly) $t-1$ times, (b) computes $x_i^{(t)} = G(s_i^{(t)})$ and $x_j^{(t)} = G(s_j^{(t)})$, and finally (c) internally invokes the obfuscated circuit $\widehat{C}$ (included as part of CRS) on input the tuple $(s_j^{(t)}, x_i^{(t)}, x_j^{(t)})$. To generate the shared key with an user i corresponding to time period t, user j runs $\widehat{C_i}$ with input its secret key $s_j^{(t)}$ corresponding to time period t to obtain the shared key $\mathsf{PRF}_K(x_i^{(t)}, x_j^{(t)})$. It is easy to see that user i also derives the same shared key for time period t by running the program $\widehat{C_j}$ (public key of user j) on input its own $s_i^{(t)}$ corresponding to time period t. The key update function for our FS+EBL NIKE can be any entropic-leakage resilient OWF [9]. This is so that it remains hard to compute the prior key even given entropic leakage on the pre-image of the OWF.

Security Proof. The security proof of our construction follows the proof technique of the LMQW protocol with some major differences as explained below. The main idea of the proof of the LMQW protocol follows the punctured programming paradigm [38], where they puncture the PRF key K at the point (x_i, x_j) and program a random output y. However, instead of hard-coding y directly they hard-core $y \oplus s_i$ and $y \oplus s_j$, i.e, the one-time pad encryption of y under s_i and s_j respectively. This allows the obfuscated program to decrypt y given either s_i or s_j as input. At this point, they switch the function G to be in *lossy* mode and argue that the shared key y retain high min-entropy, even given the obfuscated program with hard-coded ciphertexts, the public keys and leakages on the secret keys s_i and s_j. The entropic key k is then converted into a uniformly random string by using an appropriate extractor.

However, for our construction, we *cannot* argue the last step of the above proof, i.e., the shared key y (for time period t) retains enough entropy given all the public information (CRS and public keys) and entropic leakage on the keys. This is because the public keys $\hat{C}_i$ and $\hat{C}_j$ completely determine the keys $s_i^{(t)}$ and $s_j^{(t)}$ respectively, even after switching the function G to be in lossy mode. Indeed, the obfuscated programs $\hat{C}_i$ and $\hat{C}_j$ contains the base secret keys $s_i^{(1)}$ and $s_j^{(1)}$ hard-coded in them, and hence, given the public keys, the secret keys have no entropy left. To this end, we switch the public key $\hat{C}_i$ to an obfuscation of a program that, instead of embedding the base secret key $s_i^{(1)}$ embeds all possible public keys $(x_i^{(1)}, \cdots, x_i^{(T)})$ in it, where T is the total number of time period supported by our scheme and $x_i^{(j)} = G(s_i^{(j)})$ for $j \in [T]$. Note that this program is functionally equivalent but we need to pad C_i up to its size. Now, since, the function G is lossy the shared key y still retains enough entropy, even given the public key. A similar argument can be made for party j. By setting the parameters appropriately, we can prove FS+EBL security of our NIKE construction with optimal leakage rate.

Compressing the size of the public key using relaxed SuPA. Note that, in the above proof step we needed to embed T values and hence the public key of each user (which consists of the above obfuscated and padded circuit) scales linearly with T. With linear public key size, FS is trivial. However, what makes our scheme different from the trivial one is that for us this issue is a *proof problem* formally captured via the *Superfluous Padding (SuP) Assumption* [11]. Intuitively the SuP assumption (SuPA) states that if two distributions are indistinguishable relative to an obfuscated circuit C which was padded before obfuscation, then the two distributions are also indistinguishable relative to the obfuscated circuit C without padding. Or in other words, if an obfuscation of a padded circuit hides something, then so does an obfuscation of the unpadded circuit.

Although non-standard, it is shown in [11] that SuPA holds for virtual blackbox obfuscation (VBB) as evidence it holds for $i\mathcal{O}$. Unfortunately, as shown in [25], assuming $i\mathcal{O}$ and one-way functions SuPA does not hold for $i\mathcal{O}$ if the distinguisher is given *auxiliary information.* Crucially, we get around this by using a *relaxed* variant of SuPA that does not give the distinguisher auxiliary information. This relaxed SuPA is enough to prove the security of our NIKE construction. We stress the impossibility result of [25] does not apply to this relaxed SuPA, and in fact, we conjecture that, in the absence of any auxiliary information SuPA does hold for $i\mathcal{O}$. In this case, the size of the public keys in our NIKE scheme *is not linear in* T, but is only $O(\log T)$. This is because the obfuscated circuit just needs to know the maximum number of times it will need to update its keys.

3 Preliminaries

3.1 Notations

Let $x \in \mathcal{X}$ denote an element x in the support of $\mathcal{X}$. For a probability distribution $\mathcal{X}$, let $|\mathcal{X}|$ denote the size of the support of $\mathcal{X}$, i.e., $|\mathcal{X}| = | \{x \,|\, \mathsf{Pr}[\mathcal{X} = x] > 0\} |$. If x is a string , we denote $|x|$ as the length of x. Let $x \leftarrow \mathcal{X}$ be the process of sampling x from the distribution $\mathcal{X}$. For $n \in \mathbb{N}$, we write $[n] = \{1, 2, \cdots, n\}$. When A is an algorithm, we write $y \leftarrow A(x)$ 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. 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. For a set S, we let U_S denote the uniform distribution over S. For an integer $\alpha \in \mathbb{N}$, let U_α denote the uniform distribution over $\{0,1\}^\alpha$, the bit strings of length α. Throughout this paper, we denote the security parameter by κ, which is implicitly taken as input by all the algorithms. For two random variables X and Y drawn from a finite set $\mathcal{X}$, let $\delta(X, Y) = \frac{1}{2} | \sum_{x \in \mathcal{X}} \mathsf{Pr}(X = x) - \mathsf{Pr}(Y = x)|$ denote the statistical distance between them. Given a circuit D, define the computational distance δ^D between X and Y as $\delta^D(X, Y) = |\mathbb{E}[D(X)] - \mathbb{E}[D(Y)]|$.

3.2 Different Notions of Entropy

In this section, we recall some the definitions of information-theoretic and computational notions of entropy that are relevant to this work and also state the results related to them.

Unconditional (Information-theoretic) Entropy

Definition 1 (Min-entropy). *The min-entropy of a random variable* X*, denoted as* $\mathrm{H}_\infty(\mathrm{X})$ *is defined as* $\mathrm{H}_\infty(\mathrm{X}) \stackrel{\text{def}}{=} -\log\big(\max_x \Pr[X = x]\big)$.

Definition 2 (Conditional Min-entropy [16]). *The average-conditional min-entropy of a random variable* X *conditioned on a (possibly) correlated variable* Z*, denoted as* $\widetilde{\mathrm{H}}_\infty(X|Z)$ *is defined as*

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

Lemma 1 (Chain Rule for min-entropy [16]). *For any random variable* X*,* Y *and* Z*, if* Y *takes on values in* $\{0,1\}^\ell$*, then*

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

One may also define a more general notion of conditional min-entropy $\widetilde{H}_\infty(X|\mathcal{E})$, where the conditioning happens over an arbitrary experiment $\mathcal{E}$, and not just a "one-time" random variable Y [3].

Computational Entropy a.k.a pseudo-entropy

Computational entropy or pseudo-entropy is quantified with two parameters-*quality* (i.e., how much distinguishable a random variable is from a source with true min-entropy to a size-bounded (poly-time) distinguisher)) and *quantity* (i.e., number of bits of entropy).

Definition 3 (Hill Entropy [24, 26]). *A distribution $\mathcal{X}$ has* **HILL entropy** *at least k, denoted by $\mathrm{H}^{\mathsf{HILL}}_{\epsilon,s}(\mathcal{X}) \geq k$, if there exists a distribution $\mathcal{Y}$, where $\mathrm{H}_\infty(\mathcal{Y}) \geq k$, such that $\forall\, \mathcal{D} \in \mathcal{D}_s^{\mathsf{rand},\{0,1\}}$, $\delta^{\mathcal{D}}(\mathcal{X}, \mathcal{Y}) \leq \epsilon$. By, $\mathcal{D}_s^{\mathsf{rand},\{0,1\}}$ we refer to the set of all probabilistic circuits without $\{0,1\}$.*
Let $(\mathcal{X}, \mathcal{Y})$ be a pair of random variables. Then, we say that $\mathcal{X}$ has **conditional HILL entropy** *at least k conditioned on $\mathcal{Y}$, denoted $\mathrm{H}^{\mathsf{HILL}}_{\epsilon,s}(\mathcal{X}|\mathcal{Y}) \geq k$, if there exists a collection of distributions $\mathcal{Z}_y$ for each $y \in \mathcal{Y}$, yielding a joint distribution $(\mathcal{Z}, \mathcal{Y})$ such that $\widetilde{\mathrm{H}}_\infty(\mathcal{Z}|\mathcal{Y}) \geq k$, and $\forall\, \mathcal{D} \in \mathcal{D}_s^{\mathsf{rand},\{0,1\}}$, $\delta^{\mathcal{D}}\big((\mathcal{X}, \mathcal{Y}), (\mathcal{Z}, \mathcal{Y})\big) \leq \epsilon$.*

3.3 Primitives required for our constructions.

In this section, we briefly outline the primitives required for our constructions. A puncturable PRF (pPRF) allows one to evaluate the PRF on all but a subset of points (on which the master key is punctured). We require pseudorandomness to hold on the punctured points, even given the punctured key. We also require indistinguishability obfuscation ($i\mathcal{O}$), and lossy functions for our construction of the NIKE protocol in the $\mathsf{FS} + \mathsf{EBL}$ model. We refer the reader to the full version of our paper for the definitions of pPRF, $i\mathcal{O}$, and lossy functions. For our construction of $\mathsf{FS} + \mathsf{EBL}$-secure NIKE protocol, we also require an *entropic leakage-resilient* one-way function (ELR-OWF) to instantiate the update function. We also require the Superfluous padding assumption to hold for $i\mathcal{O}$. Below we present their formal definitions.

Entropic Leakage-resilient OWF. In this section, we recall the definition of leakage-resilient one-way functions (LR-OWF) from [9]. Informally, a one-way function (OWF) $g : \{0,1\}^n \to \{0,1\}^m$ is leakage-resilient if it remains one-way, even in the presence of some leakage about pre-image. In entropy-bounded leakage model, instead of bounding the length of the output of leakage functions (as in bounded leakage model), we bound the entropy loss that happens due to seeing the output of the leakage functions. We follow the definition of [14] to consider the entropy loss over the uniform distribution as a measure of leakiness. We follow this definition since it has nice composability properties as stated below.

Definition 4. [14]. A (probabilistic) function $h : \{0,1\}^* \to \{0,1\}^*$ is ℓ-leaky, if for all $n \in \mathbb{N}$, we have $\widetilde{H}_\infty(U_n|h(U_n)) \geq n - \ell$, where U_n denote the uniform distribution over $\{0,1\}^n$.

As observed in [14], if a function is ℓ-leaky, i.e, it decreases the entropy of uniform distribution by at most ℓ bits, then it decreases the entropy of every distribution by at most ℓ bits. Moreover, this definition composes nicely in the sense that, if the adversary adaptively chooses different ℓ_i-leaky functions, it learns only $\sum_i \ell_i$ bits of information. We now define the security model for weak PRFs in this entropy-bounded leakage model.

Definition 5. (Entropic leakage-resilient one-wayness). *Let $\mathcal{A}$ be an adversary against $g : \{0,1\}^n \to \{0,1\}^m$. We define the advantage of the adversary $\mathcal{A}$ as $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{LR\text{-}OWF}}(\kappa) = \mathsf{Pr}[g(x) = y \,|\, x^* \xleftarrow{\$} \{0,1\}^n, y^* = g(x^*); x \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Leak}}(\cdot)}(y^*)]$.* Here $\mathcal{O}_{\mathsf{Leak}}$ is an oracle that on input $h : \{0,1\}^n \to \{0,1\}^*$ returns $f(x^*)$, subject to the restriction that h is λ-entropy leaky. We say that g is λ-entropic leakage-resilient one-way function (λ-ELR-OWF) if not any PPT adversary $\mathcal{A}$ its advantage defined as above is negligible in κ.

As shown in [15], a second-preimage resistant (SPR) function with $n(\kappa)$ bits input and $m(\kappa)$ bits output is also a $\lambda(\kappa)$-entropy leaky OWF for $\lambda(\kappa) = n(\kappa) - m(\kappa) - \omega(\log \kappa)$.

The Superfluous Padding Assumption Following [11], we present the Superfluous Padding Assumption (SuPA). Intuitively SuPA states that if two distributions are indistinguishable relative to an obfuscated circuit C which was padded before obfuscation, then the two distributions are also indistinguishable relative to the obfuscated circuit C without padding. In other words, if an obfuscation of a padded circuit hides something, then so does an obfuscation of the unpadded circuit. Unfortunately, as shown in [25] assuming $i\mathcal{O}$ and one-way functions SuPA does not hold for $i\mathcal{O}$ if the distinguisher is given *arbitrary auxiliary* information. We present a relaxed version of the SuP assumption where the distinguisher is *not* given access to any auxiliary input and we observe that this relaxed variant of SuPA is enough to prove the security of our NIKE construction.

Following [11], we state the assumption in two steps: First, we define admissible sampler and then define the SuP assumption with respect to such an admissible sampler.

Definition 6 (Relaxed SuP-admissible Samplers). *Let* Obf *be an obfuscation scheme and let* $\mathsf{PAD} : \mathbb{N} \times \{0,1\}^* \to \{0,1\}^*$ *be a deterministic padding algorithm that takes as input an integer s and and a description of a circuit C and outputs a functionally equivalent circuit of size $s + |C|$. We say that a pair of PPT samplers* $(\mathsf{Samp}_0, \mathsf{Samp}_1)$ *is* SuP*-admissible for obfuscator* Obf *, if there exists a polynomial s such that for any PPT distinguisher D its advantage in the* $\mathsf{SuP}[s]$ *game (see Figure 1) is negligible:*

$$\mathsf{Adv}^{\mathsf{SuP}[s]}_{\mathsf{Obf},\mathsf{Samp}_0,\mathsf{Samp}_1,\mathcal{D}}(\kappa) = 2 \cdot \mathsf{Pr}\big[\mathsf{SuP}[s]^{\mathcal{D}}_{\mathsf{Obf},\mathsf{Samp}_0,\mathsf{Samp}_1}(\kappa)\big] - 1 \leq \mathsf{negl}(\kappa).$$

Definition 7 (The Relaxed SuP assumption). *Let* Obf *be an obfuscation scheme and let* Samp_0 *and* Samp_1 *be two* SuP-admissible *samplers. Then, the relaxed* Superfluous Padding Assumption *states that no efficient distinguisher $\mathcal{D}$ has a non-negligible advantage in the* $\mathsf{SuP}[0]$ *game without padding:*

$$\mathsf{Adv}^{\mathsf{SuP}[0]}_{\mathsf{Obf},\mathsf{Samp}_0,\mathsf{Samp}_1,\mathcal{D}}(\kappa) = 2 \cdot \mathsf{Pr}\big[\mathsf{SuP}[0]^{\mathcal{D}}_{\mathsf{Obf},\mathsf{Samp}_0,\mathsf{Samp}_1}(\kappa)\big] - 1 \leq \mathsf{negl}(\kappa).$$

Game $\mathsf{SuP}[s]^{\mathcal{D}}_{\mathsf{Obf},\mathsf{Samp}_0,\mathsf{Samp}_1}$

$b \leftarrow \{0,1\}$
$C \leftarrow \mathsf{Samp}_b(1^\kappa)$
If $s(\kappa) > 0$, then return $\widehat{C} \leftarrow \mathsf{Obf}(\mathsf{PAD}(s(\kappa), C))$
Else, return $\widehat{C} \leftarrow \mathsf{Obf}(C)$
$b' \leftarrow \mathcal{D}(1^\kappa, s, \widehat{C}, |C|)$
Return $(b' = b)$

Fig. 1. The SuP game parameterized by a polynomial $s(\kappa)$. According to s, the circuit C is padded (if $s = 0$ the original circuit C is used) before it is obfuscated and given to distinguisher $\mathcal{D}$, who additionally gets s as well as the size of the original circuit C.

4 Our Results in the FS+CL Model

4.1 Encryption in the FS+CL Model

In this section, following [6] we recall the syntax and security definition of encryption schemes in the FS+CL Model.

Encryption in the FS+CL Model. A key-evolving encryption scheme KEE specifies the following PPT algorithms $\mathsf{KEE.Kg}$, $\mathsf{KEE.Upd}$, $\mathsf{KEE.Enc}$, and $\mathsf{KEE.Dec}$, where $\mathsf{KEE.Dec}$ is deterministic. The encryption scheme KEE is associated with the maximum number of time periods $T = T(\kappa)$. Here, $\mathsf{KEE.Kg}(1^\kappa)$ is used to generate the initial key pair (sk_1, pk). The key update algorithm $\mathsf{KEE.Upd}(1^\kappa, pk, i, sk_i)$ is used to evolve/update the key from time period i to $i+1$, outputting sk_{i+1} in the process. $\mathsf{KEE.Enc}$ is used to encrypt a message m in time period i using the public key pk. $\mathsf{KEE.Dec}$ is used to decrypt a ciphertext c, produced in time i, with the help of secret key sk_i. We require the standard correctness condition from KEE. The security game is presented in Figure 2. In this game defining forward indistinguishability of key-evolving encryption scheme KEE under continual leakage (FINDCL), an attacker is given access to three oracles : Up (which it uses to update the key), Leak (which it uses to leak on the key with its choice of leakage function L, and a one-time access to Exp which gives the entire secret key sk_{t^*}. One additional constraint is that the attacker $\mathcal{A}$ is δ*-bounded*, i.e., $\mathcal{A}$ is allowed to leak at most $\delta(\kappa)$ bits from the secret keys *per time period*. The attacker provides challenge messages i, m_0, m_1 and a time period i. It receives an encryption of m_b for a randomly chosen bit b. $\mathcal{A}$ wins the game if it correctly guesses the bit b and if $i < t^*$. An encryption scheme is FINDCL-secure if the advantage of $\mathcal{A}$ in winning the above game is negligible.

4.2 Our Construction

In this section, we provide the details of our FS+CL encryption scheme. To this end, we first abstract out a notion of continuous leakage-resilient binary tree encryption (CLR-BTE) and use it to construct our FS+CL encryption scheme achieving optimal leakage rate, i.e., $1 - o(1)$.

Game $\mathrm{FINDCL}^{\mathcal{A}}_{\mathsf{KEE}}(\kappa)$	$\underline{\mathsf{Up}()}$
$b \leftarrow\!\!\$\ \{0,1\}$; $t \leftarrow 1$; $t^* \leftarrow T(\kappa)+1$	If $t < T(\kappa)$ then
$(pk, sk_1) \leftarrow\!\!\$\ \mathsf{KEE.Kg}(1^\kappa)$	$\quad sk_{t+1} \leftarrow\!\!\$\ \mathsf{KEE.Upd}(1^\kappa, pk, t, sk_t)$
$(i, m_0, m_1, state) \leftarrow\!\!\$\ \mathcal{A}_1^{\mathsf{Up},\mathsf{Leak},\mathsf{Exp}}(1^\kappa, pk)$	$\quad t \leftarrow t+1$
If not $(1 \le i < t^*)$ then return false	Else return $\perp$
If $\lvert m_0\rvert \ne \lvert m_1\rvert$ then return false	$\underline{\mathsf{Leak}(L)}$
$(i, c) \leftarrow\!\!\$\ \mathsf{KEE.Enc}(1^\kappa, pk, i, m_b)$	Return $L(sk)$
$b' \leftarrow\!\!\$\ \mathcal{A}_2(1^\kappa, state, (i, c))$	$\underline{\mathsf{Exp}()}$
Return $(b' = b)$	$t^* \leftarrow t$; Return sk_t

Fig. 2. Game defining forward indistinguishability of key-evolving encryption scheme KEE under continual leakage.

Continuous Leakage-resilient Binary Tree Encryption. We now introduce our notion of binary tree encryption in the continuous leakage model. Our security model of the CLR-BTE scheme generalizes the definition of binary tree encryption (BTE) (proposed by Canetti et al. [12]) in the setting of continuous leakage. A BTE can be seen as a restricted version of HIBE, where the identity tree is represented as a *binary* tree.[2] In particular, as in HIBE, a BTE is also associated with a "master" public key MPK corresponding to a tree, and each node in the tree has its respective secret keys. To encrypt a message for a node, one specifies the identity of the node and the public key MPK. The resulting ciphertext can be decrypted using the secret key of the target node.

Definition 8. *(Continuous leakage-resilient BTE).* A continuous leakage-resilient binary tree encryption scheme (CLR-BTE) consists of a tuple of the PPT algorithms $(\mathsf{Gen}, \mathsf{Der}, \mathsf{Upd}, \mathsf{Enc}, \mathsf{Dec})$ such that:

1. $\mathsf{Gen}(1^\kappa, 1^\ell)$: The key generation algorithm Gen takes as input the security parameter κ and a value ℓ for the depth of the tree. It returns a master public key MPK and an initial (root) secret key SK_ε.
2. $\mathsf{Der}(MPK, w, Sk_w)$: The key derivation algorithm Der takes as input MPK, the identity of a node $w \in \{0,1\}^{\le \ell}$, and its secret key SK_w. It returns secret keys SK_{w_0}, SK_{w_1} for the two children of w.
3. $\mathsf{Upd}(w, Sk_w)$: The key update algorithm Upd takes as input the secret key SK_w of a node w and outputs a re-randomized key SK'_w for the same node w, such that $\lvert SK'_w\rvert = \lvert SK_w\rvert$.
4. $\mathsf{Enc}(MPK, w, M)$: The encryption algorithm Enc takes as input MPK, the identity of a node $w \in \{0,1\}^{\le \ell}$ and a message M to return a ciphertext C.
5. $\mathsf{Dec}(MPK, w, Sk_w, C)$: The decryption algorithm Dec takes as input MPK, the identity of a node $w \in \{0,1\}^{\le \ell}$, its secret key SK_w, and a ciphertext C. It returns a message M or $\perp$ (to denote decryption failure).

[2] Recall that in HIBE the tree can have an arbitrary degree.

Correctness: For all (MPK, SK_ε) output by Gen, any node $w \in \{0,1\}^{\leq \ell}$, any secret key SK_w correctly generated for this node (which can be the output of (multiple invocations of) Upd also), and any message M, we have

$$\mathsf{Dec}(MPK, w, SK_w, \mathsf{Enc}(MPK, w, M)) = M.$$

Security model for CLR-BTE. Our security model for CLR-BTE generalizes the notion of *selection-node chosen-plaintext attacks* (SN-CPA) put forward by Canetti et al. [12] to define the security of BTE. In our model, the adversary first specifies the identity of the target node[3] $w^* \in \{0,1\}^{\leq \ell}$. The adversary receives the public key MPK and the secret keys of all the nodes that do not trivially allow him/her to derive the secret key of w^*[4]. Besides, the adversary is also allowed to continuously leak from the secret keys of all the nodes that lie on the path from the root node and w^* (including both). The goal of the adversary is then to win the indistinguishability game with respect to the target node w^*.

Definition 9. A CLR-BTE scheme is secure against *continuous leakage selective-node, chosen-plaintext attacks* ($\lambda(\kappa)$-CLR-SN-CPA) if for all polynomially-bounded functions $\ell(\cdot)$, and leakage bound $\lambda(\kappa)$, the advantage of any PPT adversary $\mathcal{A}$ in the following game is negligible in the security parameter κ:

1. The adversary $\mathcal{A}(1^\kappa, \ell)$ outputs the name of a node $w^* \in \{0,1\}^{\leq \ell}$. We will denote the path from the root node to the target node w^* by P_{w^*}.
2. The challenger runs the algorithm $\mathsf{Gen}(1^\kappa, \ell)$ and outputs (MPK, SK_ε). In addition, it runs $\mathsf{Der}(\cdot,\cdot,\cdot)$ to generate the secret keys of all the nodes on the path P_{w^*}, and also the secret keys for the two children w_0^* and w_1^*. The adversary is given MPK and the secret keys $\{SK_w\}$ for all nodes w of the following form:
   - $w = w'\bar{b}$, where $w'b$ is a prefix of w^* and $b \in \{0,1\}$ (i.e., w is a sibling of some node in P_{w^*}).
   - $w = w_0^*$ or $w = w_1^*$ (i.e., w is a child of w^*; this is only when $|w^*| < \ell$).

   The challenger also creates a set $\mathcal{T}$ that holds tuples of all the (node) identities, secret keys and the number of leaked bits from each key so far.
3. The adversary $\mathcal{A}_{\mathsf{clr\text{-}bte}}$ may also ask leakage queries. The adversary runs for arbitrarily many leakage rounds. In each round:
   - The adversary provides the description of a probabilistic leakage function $h : \{0,1\}^* \to \{0,1\}^{\lambda(\kappa)}$, and an identity of a node w in the path P_{w^*}

[3] Note that, this model where the adversary specifies the target node w^* ahead of time is weaker than the model where the adversary may choose the target *adaptively* (analogous to the adaptive security of HIBE schemes). However, as we will show, this model already suffices to construct of a FS+CL encryption scheme.

[4] In particular, the adversary receives the secret keys of all the nodes that are siblings of all the nodes that are on the path from the root node to the target node w^*.

(that may also include both the root note and the target node w^*).[5] The challenger scans $\mathcal{T}$ to find the tuple with identity w. It should be of the form (w, SK_w, L_w). The challenger then checks if $L_w + |h(SK_w)| \leq \lambda(\kappa)$. If this is true, it responds with $h(SK_w)$ and updates $L_w = L_w + |h(SK_w)|$. If the check fails, it returns $\perp$ to the adversary.

– At the end of each round, the challenger computes $SK'_w \leftarrow \mathsf{Upd}(w, SK_w)$ and updates $SK_w = SK'_w$.

4. The adversary $\mathcal{A}$ then sends two messages M_0 and M_1 to the challenger such that $|M_0| = |M_1|$. The challenger samples a random bit $b \xleftarrow{\$} \{0,1\}$, and computes $C^* \leftarrow \mathsf{Enc}(MPK, w^*, M_b)$. It then returns C^* to the adversary $\mathcal{A}$. The adversary is not allowed to ask any further leakage queries after receiving the challenge ciphertext C^*.[6]

At the end of this game, the adversary outputs a bit $b' \in \{0,1\}$; it succeeds if $b' = b$. The advantage of the adversary is the absolute value of the difference between its success probability and 1/2.

Construction of CLR-BTE scheme. Our construction of the CLR-BTE scheme can be instantiated in a straightforward manner from the continuous leakage-resilient HIBE (CLR-HIBE) construction of Lewko et al. [33], tuned to the setting of a binary tree. The resulting CLR-BTE is *adaptively secure*, since the CLR-HIBE of [33] enjoys security against adaptive adversaries employing the dual-system encryption technique. The security of the CLR-BTE scheme can be proven under static assumptions over composite-order bilinear groups. We refer the reader to [33] for the details of the CLR-HIBE construction and its proof. As shown in [33], for appropriate choice of parameters, their CLR-HIBE scheme achieves the optimal leakage rate of $1 - o(1)$.

FINDCL encryption from CLR-BTE scheme. We now show a generic construction of a FINDCL-secure encryption scheme starting from any CLR-BTE scheme. The main idea of our construction is very simple: use the Canetti-Halevi-Katz (CHK) transform [12] to the underlying CLR-BTE scheme to construct a FINDCL encryption scheme. In particular, we show the applicability of the CHK transform[7] even in the setting of continuous leakage. However, as we show later, the analysis of the CHK transform in the setting of leakage turns out to be quite tricky.

[5] This is equivalent to a definition where, in each round, the adversary asks for multiple leakage functions adaptively, such that the output length of all these functions sum up to $\lambda(\kappa)$.

[6] If the adversary is allowed to ask leakage queries after receiving the challenge ciphertext, it can encode the entire decryption algorithm of C^* as a function on a secret key, and thus win the game trivially.

[7] The original CHK transform [12] is used to construct a forward-secure PKE scheme starting from a BTE scheme.

Let $(\mathsf{Gen}, \mathsf{Der}, \mathsf{Upd}, \mathsf{Enc}, \mathsf{Dec})$ be a CLR-BTE scheme. We construct our FINDCL PKE scheme ($\mathsf{KEE.Kg}$, $\mathsf{KEE.Upd}$, $\mathsf{KEE.Enc}$, $\mathsf{KEE.Dec}$) as shown below. The construction is identical to the CHK transform, with the underlying building blocks appropriately changed.

Some additional notation: To obtain a FINDCL-secure encryption scheme with $T = 2^\ell - 1$, time periods (labeled through 1 to T), we use a CLR-BTE of depth ℓ. We associate the time periods with all nodes of the tree according to a pre-order traversal. The node associated with time period i is denoted by w^i. In a pre-order traversal, $w^1 = \varepsilon$ (the root node), if w^i is an internal node then $w^{i+1} = w^i 0$ (i.e., left child of w^i). If w^i is a leaf node and $i < T - 1$ then $w^{i+1} = w'1$, where w' is the longest string such that $w'0$ is a prefix of w^i.

1. $\mathsf{KEE.Kg}(1^\kappa, T)$: Run $\mathsf{Gen}(1^\kappa, \ell)$, where $T \leq 2^\ell - 1$, and obtain (MPK, SK_ε). Set $pk = (MPK, T)$, and $sk_1 = SK_\varepsilon$.
2. $\mathsf{KEE.Upd}(1^\kappa, pk, i, sk_i)$: The secret key sk_i is organized as a stack of node keys, with the secret key SK'_{w^i} on top, where SK'_{w^i} is obtained by running Upd of the CLR-BTE scheme (potentially multiple times) on the key SK_{w^i}. We first pop this key off the stack. If w^i is a leaf node, the next node key on top of the stack is $SK'_{w^{i+1}}$ (a refreshed version of the key $SK_{w^{i+1}}$). If w^i is an internal node, compute $(SK_{w^i 0}, SK_{w^i 1}) \leftarrow \mathsf{Der}(pk, w^i, SK_{w^i})$ Then for $b \in \{0,1\}$, compute $SK'_{w^i b} \leftarrow \mathsf{Upd}(w, SK_{w^i b})$. Further, for all other node keys SK_w remaining in the stack (corresponding to sk_{i+1}), run $SK'_w \leftarrow \mathsf{Upd}(w, SK_w)$. Then push $SK'_{w^i 1}$ and then $SK'_{w^i 0}$ onto the stack. In either case, the node key SK'_{w^i} is erased.
3. $\mathsf{KEE.Enc}(pk, i, m)$: Run $\mathsf{Enc}(pk, w^i, m)$. Note that w^i is publicly computable given i and T.
4. $\mathsf{KEE.Dec}(1^\kappa, pk_i, sk_i, c_i)$: Run $\mathsf{Dec}(pk, w, SK'_{w^i}, c_i)$. Note that, SK'_{w^i} is stored as part of sk_i.

Theorem 1. *Let* $\lambda : \mathbb{N} \to [0,1]$. *Let* $\Pi = (\mathsf{Gen}, \mathsf{Der}, \mathsf{Upd}, \mathsf{Enc}, \mathsf{Dec})$ *be a* $\lambda(\kappa)$-CLR-SN-CPA *continuous leakage-resilient binary-tree encryption* (CLR-BTE) *scheme. Let* $\ell : \mathbb{N} \to \mathbb{N}$ *be a polynomial such that* $T \leq 2^\ell - 1$. *Then* $\Pi' = (\mathsf{KEE.Kg}, \mathsf{KEE.Upd}, \mathsf{KEE.Enc}, \mathsf{KEE.Dec})$ *is a* $\lambda(\kappa)$-FINDCL *secure encryption scheme supporting up to* T *time periods.*

Proof. Our proof follows the template of the CHK transformation for converting a BTE scheme to forward-secure encryption scheme, with the crucial difference in simulating the leakage queries.

Assume that we have an adversary $\mathcal{A}_{\mathsf{kee}}$ with advantage $\epsilon(\kappa)$ in an $\lambda(\kappa)$-FINDCL security game of $\Pi' = (\mathsf{KEE.Kg}, \mathsf{KEE.Upd}, \mathsf{KEE.Enc},$ $\mathsf{KEE.Dec})$. We construct an adversary $\mathcal{A}_{\mathsf{clr\text{-}bte}}$ that obtains an advantage $\epsilon(\kappa)/T$ in the corresponding attack against the underlying the CLR-BTE scheme $\Pi = (\mathsf{Gen}, \mathsf{Der}, \mathsf{Upd}, \mathsf{Enc}, \mathsf{Dec})$. The leakage rate tolerated by Π is exactly the same as Π'. We now describe how $\mathcal{A}_{\mathsf{clr\text{-}bte}}$ simulates the environment for $\mathcal{A}_{\mathsf{kee}}$:

1. $\mathcal{A}_{\mathsf{clr\text{-}bte}}$ chooses uniformly at random a time period $i^* \in [T]$. This define the node w^{i^*} (the identity of the node corresponding to i^*). $\mathcal{A}_{\mathsf{clr\text{-}bte}}$ then forwards w^{i^*} to its challenger and obtains MPK and $\{SK_w\}$ for all the appropriate nodes w[8] from its challenger. $\mathcal{A}_{\mathsf{clr\text{-}bte}}$ then sets $pk = (MPK, T)$, and forwards the public key pk to the adversary $\mathcal{A}_{\mathsf{kee}}$.
2. When $\mathcal{A}_{\mathsf{kee}}$ decides to break into the system, it provides the time period, say j. If $j \le i^*$, then $\mathcal{A}_{\mathsf{clr\text{-}bte}}$ outputs a random bit and halts. Otherwise, $\mathcal{A}_{\mathsf{clr\text{-}bte}}$ computes the appropriate secret key sk_j and gives it to $\mathcal{A}_{\mathsf{kee}}$. Note that, $\mathcal{A}_{\mathsf{clr\text{-}bte}}$ can efficiently compute the secret keys sk_j for any $j > i^*$ from the knowledge of $\{SK_w\}$ (the set of secret keys received in Step 1).
3. $\mathcal{A}_{\mathsf{kee}}$ may ask leakage queries on the secret key corresponding to any time period, say i. The node associated with time period i is w^i. The secret key sk_i can be seen as a stack of node keys (derived using the underlying CLR-BTE scheme) with the key SK'_{w^i} on top of the stack. The other node keys in the stack are secret keys corresponding to the right siblings of all the nodes in the path P_{w^i} from the root node to w^i. Let us denote the secret key as $sk_i = (SK'_{w^i}, \{SK\}'_{rs(P_{w^i})})$, where $\{SK\}'_{rs(P_{w^i})}$ denote the (refreshed) secret keys of the right siblings of all nodes in path from the root to the node w^i (we denote this path by P_{w^i}). Now, either one of the two cases must be true: (1) $w^i \in P_{w^{i^*}}$ or (2) $w^i \notin P_{w^{i^*}}$, where $P_{w^{i^*}}$ is the path containing the nodes from the root node to the target node w^{i^*} (including both).

   For the first case, $\mathcal{A}_{\mathsf{clr\text{-}bte}}$ already knows all the keys $\{SK\}_{rs(P_{w^i})})$ and it does the following:
   - Receive as input the leakage function f from $\mathcal{A}_{\mathsf{kee}}$. Next, it calls Upd on all the node keys $\{SK\}_{rs(P_{w^i})}$ and receive the set of refreshed keys $\{SK\}'_{rs(P_{w^i})}$. It then modifies the description of the function as $h = f_{\{SK\}'_{rs(P_{w^i})}}(\cdot) = f\big(\cdot, \{SK\}'_{rs(P_{w^i})}\big)$. In other words, $\mathcal{A}_{\mathsf{clr\text{-}bte}}$ hardwires the secret keys $\{SK\}'_{rs(P_{w^i})}$ in the function f, and forwards h as the leakage function to its challenger.
   - On input the answer $h(SK'_{w^i}, \{SK\}'_{rs(P_{w^i})})$ from its challenger, $\mathcal{A}_{\mathsf{clr\text{-}bte}}$ forwards this answer as the output of the leakage function f to $\mathcal{A}_{\mathsf{kee}}$.

   For the second case (i.e., when $w^i \notin P_{w^{i^*}}$), there exists at most one node $w \in P_{w^{i^*}}$ whose secret key is included $\{SK\}'_{rs(P_{w^i})}$. Apart from the secret key of w, $\mathcal{A}_{\mathsf{clr\text{-}bte}}$ knows the secret key of w^i, SK_{w^i}, and the keys $\{SK\}_{rs(P_{w^i})})$. Thus, similar to above it can transform the joint leakage function f to a leakage function h on SK'_w. It then returns the result to $\mathcal{A}_{\mathsf{kee}}$.

   It is clear that in both cases, $\mathcal{A}_{\mathsf{clr\text{-}bte}}$ perfectly simulates the answers to the leakage queries of the adversary $\mathcal{A}_{\mathsf{kee}}$.
4. When $\mathcal{A}_{\mathsf{kee}}$ asks an update query $\mathsf{KEE.Upd}(i)$, $\mathcal{A}_{\mathsf{blr\text{-}bte}}$ can easily compute the key for the next time period using the knowledge of the keys $\{SK_w\}$ received from its challenger in the beginning.

[8] Recall that $\mathcal{A}_{\mathsf{clr\text{-}bte}}$ receives the secret keys of all the nodes that are right siblings of the nodes that lie on the path P from the root node to w^{i^*}.

5. When $\mathcal{A}_{\mathsf{kee}}$ asks a challenge query with input (i, m_0, m_1), if $i \neq i^*$ then $\mathcal{A}_{\mathsf{clr\text{-}bte}}$ outputs a random bit and halts. Otherwise, it forwards the tuple (m_0, m_1) to its challenger and obtains the challenge ciphertext C^*. It then gives C^* to $\mathcal{A}_{\mathsf{kee}}$.
6. When $\mathcal{A}_{\mathsf{kee}}$ outputs b', $\mathcal{A}$ outputs b' and halts.

It is easy to see that, if $i = i^*$, the above simulation by $\mathcal{A}_{\mathsf{clr\text{-}bte}}$ is perfect. Since, $\mathcal{A}_{\mathsf{clr\text{-}bte}}$ guesses i^* with probability $1/T$, we have that $\mathcal{A}_{\mathsf{clr\text{-}bte}}$ correctly predicts the bit b with advantage $\epsilon(\kappa)/T$. □

5 Our Results in the FS+(C)EBL Model

In this section we present the FS+(C)EBL model and present a construction of NIKE in the FS+EBL model.

5.1 The FS+EBL Model

The Entropy Bounded Leakage (EBL) model was designed to capture security against adversary who leaked on the secret key. However, to make the attack non-trivial, it defines the legitimacy of the adversary. An adversary is legitimate if the secret key *sk* still contains enough min-entropy, parametrized by α, even after the leakage. This is a generalization of length-bounded leakage model where a leakage function can leak at most, say, δ bits. Implicitly, the EBL model is defined in the setting of a single time period (as there is only the one secret key). The notion of length bounded leakage model was extended to the setting of multiple time periods and this was called continual leakage model. In this setting, the secret key is updated (using a randomized update function) across time periods and an adversary can leak at most δ bits in every time period. In this section, we consider *deterministic* key update functions and take the idea of entropic bounded leakage model and extend it to the setting of multiple time periods. We consider the combined problem of FS+EBL, i.e., schemes that are forward secure and which are resilient to entropic bounded leakage. Specifically, we consider the Forward Secure + Entropic Bounded Leakage Model, abbreviated as FS+EBL. It is parametrized by T and α, where T is the maximum number of time periods and α is the minimum residual entropy required. As before, one can define the legitimacy of the attacker in this model.

Definition 10 (Definition of Legitimacy - FS+EBL Model). *Let Π be any key-evolving scheme with a* deterministic *key update algorithm. Let SK_i denote the random variable produced by the key update algorithm for time period i. Then, any PPT adversary $\mathcal{A}$ making leakage queries denoted by $\mathbb{L}_i(SK_i)$ for $i = 1, \ldots, T$, is legitimate in the (T, α)-FS+EBL model if:*

$$\forall j \in [t^*], \mathrm{H}_\infty\big(SK_j \,|\, \mathbb{L}_1, \cdots, \mathbb{L}_T, R_{L_1}, \cdots, R_{L_T}\big) \geq \alpha \tag{1}$$

where R_{L_i} denote the random coins of the adversary corresponding to the leakage function L_i, $t^* \leq T$ is the time period at which $\mathcal{A}$ is given sk_{t^*} in full.

Remark 1. In the above definition, one can also have H to be the computational notion of entropy such as HILL entropy or unpredictability entropy [24, 26].

5.2 NIKE in $\mathsf{FS} + \mathsf{EBL}$ Model.

Non-interactive key exchange (NIKE) protocols allow two (or more) parties to establish a shared key between them, without any interaction. It is assumed that the public keys of all the parties are pre-distributed and known to each other. In this work, we consider two-party NIKE protocols and extend them to the setting of *forward-security* under *entropy-bounded* leakage model ($\mathsf{FS} + \mathsf{EBL}$). We provide the definition of NIKE in this model (we will often call such a NIKE scheme as FS-EBLR-NIKE). To bypass the black-box impossibility result of constructing leakage-resilient NIKE protocol in the plain model [34], we consider the NIKE protocols in the *common reference string* (CRS) model, where we rely on leak-free randomness to generate the CRS. Our security model for FS-EBLR-NIKE scheme can be seen as a leakage-resilient adaptation of the model of forward-secure NIKE (FS-NIKE) of Pointcheval and Sanders (dubbed as $\mathcal{PS}$ model) [37]. Hence, we call our model of NIKE as the $\mathcal{EBL}$-$\mathcal{PS}$ model.

5.3 Syntax of FS-EBLR NIKE.

A NIKE scheme NIKE in the $\mathsf{FS} + \mathsf{EBL}$ model consists of the tuple of algorithms (NIKE.Setup, NIKE.Gen, NIKE.Upd, NIKE.Key). We associate to NIKE a public parameter space $\mathcal{PP}$, public key space $\mathcal{PK}$, secret key space $\mathcal{SK}$, shared key space $\mathcal{SHK}$, and an identity space $\mathcal{IDS}$. Identities are used to track which public keys are associated with which users; we are *not* in the identity-based setting.

- $\mathsf{NIKE.Setup}(1^\kappa, (\alpha, T))$: This is a randomized algorithm that takes as input the security parameter κ (expressed in unary), parameters α and T of the (T, α)-$\mathsf{FS} + \mathsf{EBL}$ model (where α is the leakage parameter and T denotes the maximum number of time period supported by the system[9]) and outputs public parameters $params \in \mathcal{PP}$.
- $\mathsf{NIKE.Gen}(1^\kappa, ID)$: On input an identity $ID \in \mathcal{IDS}$, the key generation outputs a public-secret key pair (pk, sk_t) for the current time period t. We assume that the secret keys implicitly contain the time periods. The current time period t is initially set to 1.
- $\mathsf{NIKE.Upd}(sk_t)$: The (deterministic) update algorithm takes as input the secret key sk_t at time period t and outputs the updated secret key sk_{t+1} for the next time period $t + 1$, if $t < T$. We require that the updated key $sk_{t+1} \neq sk_t$. The key sk_t is then securely erased from memory. If $t = T$, then the secret key is erased and there is no new key.
- $\mathsf{NIKE.Key}(ID_A, pk^A, ID_B, sk_t^B)$: On input an identity $ID_A \in \mathcal{IDS}$ associated with public key pk_A, and another identity $ID_B \in \mathcal{IDS}$ with secret key sk_t^B corresponding to the current time period t, output the shared key

[9] Our construction will achieve security for arbitrary polynomial T.

Game $\mathsf{EBL\text{-}PS}_{\mathsf{NIKE}}^{\mathcal{A}_{\mathsf{nike}}}(\kappa, \alpha, T)$

$params \leftarrow \mathsf{NIKE.Setup}(1^{\kappa}, \alpha, T)$; $S, C, Q \leftarrow \emptyset$ // S, C and Q maintains the list of *honest*, *corrupt* and *exposed* users respectively.

$(ID_A, ID_B, \tilde{t}) \leftarrow \mathcal{A}_1^{\mathsf{RegHon,RegCor,CorrReveal,Leak,Exp}}(params)$ // The descriptions of the oracles RegHon, RegCor, CorrReveal, Leak and Exp are provided below the description of this game.

$b \leftarrow\!\!\$ \{0, 1\}$
If $b = 0$ then Return $shk_{\tilde{t}}^{AB} \leftarrow \mathsf{NIKE.Key}(ID_A, pk_A, ID_B, sk_{\tilde{t}}^{B})$
Else Return $shk_{\tilde{t}}^{AB} \leftarrow\!\!\$ \mathcal{SHK}$
$b' \leftarrow\!\!\$ \mathcal{A}_2^{\mathsf{RegHon,RegCor,CorrReveal,Leak,Exp}}(shk_{\tilde{t}}^{AB})$

If $(ID_A, -, -, \mathsf{corrupt}) \in C$ or $(ID_B, -, -, , \mathsf{corrupt}) \in C$, then return $\perp$
If $(ID_A, t^*) \in Q$ and $t^* \leq \tilde{t}$, then return $\perp$
If $(ID_B, t^*) \in Q$ and $t^* \leq \tilde{t}$, then return $\perp$
Return $(b' = b)$

Fig. 3. Game defining security of NIKE scheme NIKE in the $\mathsf{FS} + \mathsf{EBL}$ model.

$shk_t^{AB} \in \mathcal{SHK}$ or a failure symbol $\perp$. If $ID_A = ID_B$, the algorithm outputs $\perp$. Since the secret key sk_t^B is associated with time period t, the shared key shk_t^{AB} between the two users ID_A and ID_B also corresponds to the same time period t.

Correctness: The correctness requirement states that the shared keys computed by any two users ID_A and ID_B in the *same* time period are *identical*. In other words, for any time period $t \geq 1$, and any pair (ID_A, ID_B) of users having key pairs (pk^A, sk_t^A) and (pk^B, sk_t^B) respectively, it holds that:

$$\mathsf{NIKE.Key}(ID_A, pk^B, ID_B, sk_t^A) = \mathsf{NIKE.Key}(ID_B, pk^A, ID_A, sk_t^B).$$

5.4 Security Model for FS-EBLR NIKE.

Our security model for NIKE generalizes the model of forward-secure NIKE of [37] (often referred to as the $\mathcal{PS}$ model). We refer to our model as the $\mathcal{EBL}$-$\mathcal{PS}$ model. Security of a NIKE protocol NIKE in the $\mathcal{EBL}$-$\mathcal{PS}$ model is defined by a game EBL-PS between an adversary $\mathcal{A}_{\mathsf{nike}} = (\mathcal{A}_1, \mathcal{A}_2)$ and a challenger $\mathcal{C}$ (see Figure 3). Before the beginning of the game, the challenger $\mathcal{C}$ also initializes three sets S, C and Q to be empty sets. The adversary $\mathcal{A}_{\mathsf{nike}}$ can query the following oracles:

1. $\mathsf{RegHon}(ID)$: This oracle is used by $\mathcal{A}_{\mathsf{nike}}$ to register a new honest user ID at the initial time period. The challenger runs the NIKE.Gen algorithm with the current time period as 1, and returns the public key pk to $\mathcal{A}_{\mathsf{nike}}$. It also adds the tuple $(ID, sk_1, pk, \mathsf{honest})$ to the set S. This implicitly defines all the future keys $sk_2, \cdots, sk_T$ (since the update function is deterministic). This query may be asked at most *twice* by $\mathcal{A}_{\mathsf{nike}}$. Users registered by this query are called “honest”.

2. $\mathsf{RegCor}(ID, pk)$: This oracle allows the adversary to register a new corrupted user ID with public key pk. The challenger adds the tuple $(ID, --, pk, \mathsf{corrupt})$ to the set C. We call the users registered by this query as "corrupt".
3. $\mathsf{CorrReveal}(ID_A, ID_B, t)$: $\mathcal{A}_{\mathsf{nike}}$ supplies two indices where ID_A was registered as *corrupt* and ID_B was registered as *honest*. The challenger looks up the secret key sk_1^B (corresponding to ID_B) and computes the updated key sk_t^B corresponding to time period t. Then it runs $\mathsf{NIKE.Key}(ID_A, pk^A, ID_B, sk_t^B)$ to get the shared key shk_t^{AB} for time period t and returns shk_t^{AB} to $\mathcal{A}_{\mathsf{nike}}$.
4. $\mathsf{Leak}(L, ID, t)$: The adversary $\mathcal{A}_{\mathsf{nike}}$ submits a leakage function $L : \mathcal{PP} \times \mathcal{SK} \rightarrow \{0,1\}^*$ to leak on the secret key of user ID for time period t, provided that $\mathcal{A}_{\mathsf{nike}}$ belongs to the class of legitimate adversaries (see Definition 10).
5. $\mathsf{Exp}(ID, t^*)$: This query is used by $\mathcal{A}_{\mathsf{nike}}$ to get the secret key of an honestly registered user ID corresponding to time period t^*. The challenger looks for a tuple $(ID, sk_1, pk, \mathsf{honest})$. If there is a match, it computes sk_{t^*} corresponding to t^* and returns sk_{t^*} to $\mathcal{A}_{\mathsf{nike}}$. Else, it returns $\perp$. The challenger adds (ID, t^*) to the set Q.

The formal details of our EBL-PS game is given in Figure 3.

Definition 11 ($\mathsf{FS} + \mathsf{EBL}$-secure NIKE). *A NIKE protocol* NIKE *is* (T, α)-forward-secure under computational-entropy-bounded leakage model ((T, α) - $\mathsf{FS} + \mathsf{EBL}$) *with respect to any legitimate adversary* $\mathcal{A}_{\mathsf{nike}}$ *playing the above* EBL-PS *game (see Figure 3), if the advantage defined below is negligible in* κ.

$$\mathsf{Adv}^{\mathsf{fs\text{-}ebl}}_{\mathcal{A}_{\mathsf{nike}}}(\kappa) = |\Pr[\mathsf{EBL\text{-}PS}^{\mathcal{A}_{\mathsf{nike}}}_{\mathsf{NIKE}}(\kappa, \alpha, T)) = 1] - 1/2|$$

.

In other words, the adversary $\mathcal{A}_{\mathsf{nike}}$ succeeds in the above experiment if it is able to distinguish a valid shared key between two users from a random session key. To avoid trivial win, some restrictions are enforced, namely: (i) both the targeted (or test) users needs to be *honestly registered* (ii) the adversary $\mathcal{A}_{\mathsf{nike}}$ is not allowed to obtain the secret keys corresponding to any of the test users prior to the challenge time period $\tilde{t}$, (iii) $\mathcal{A}_{\mathsf{nike}}$ is allowed to leak on the secret keys of both the target users ID_A and ID_B, as long as it satisfies the legitimacy condition (see Definition 10). We emphasize that the adversary can still obtain the secret keys of the target users ID_A and ID_B for time periods $t^* > \tilde{t}$, which models *forward security.*

Variants of NIKE. Similar to [34], we consider different variants of NIKE depending on whether the setup algorithm just outputs a uniformly random coins or sample from some structured distributions. In particular, we say a NIKE scheme is:

- a *plain* NIKE, if $\mathsf{NIKE.Setup}(1^\kappa)$ just outputs (some specified number of) uniform random coins. In particular, $\mathsf{NIKE.Setup}(1^\kappa; r) = r$.
- a NIKE in the *common reference string* model, if $\mathsf{NIKE.Setup}(1^\kappa)$ can be arbitrary (i.e., sample from an arbitrary distribution). In this case, we rely on *leak-free randomness* to run the setup algorithm.

Remark 2. We note that, in the original $\mathcal{PS}$ model of forward secure NIKE, there can be multiple honest users, and the adversary is allowed to obtain the secret keys of the honest users other than the target users (even prior to the challenge time period $\tilde{t}$). In this work, we consider a simplified version where there are only *two* honest users. The above simplified model can be shown to be polynomially equivalent to the full-fledged $\mathcal{PS}$ model by following the same reduction strategy as in [18][Theorem 8, Appendix B], where they show that the CKS-light model (with two honest users) is polynomially equivalent to the CKS-heavy model (where they can be multiple honest users). We emphasize that, in our application of constructing $\mathsf{FS}+\mathsf{EBL}$-secure PKE scheme from $\mathsf{FS}\text{-}\mathsf{EBLR}$ NIKE, we only require the above simplified model.

5.5 Construction of NIKE scheme in the $\mathsf{FS}+\mathsf{EBL}$ model

In this section, we present our construction of forward-secure NIKE protocol resilient to entropy-bounded leakage in the common reference string model.

Let $i\mathcal{O}$ be an indistinguishability obfuscator for circuits, $\mathsf{pPRF} = (\mathsf{pPRF.keygen}, \mathsf{pPRF.puncture}, \mathsf{pPRF.eval})$ be a puncturable PRF with image space $\mathcal{Y} = \{0,1\}^y$, $\mathsf{LF} = (\mathsf{Inj}, \mathsf{Lossy}, f)$ be a (κ, k, m)[10]-lossy function, and $\mathsf{LF}' = (\mathsf{Inj}', \mathsf{Lossy}', f')$ be a (κ', k', m')-lossy function, where $\kappa' \geq m$.

- $\mathsf{NIKE.Setup}(1^\kappa, T)$: Choose a random key $K \leftarrow \mathsf{pPRF.keygen}(1^\kappa)$. Sample two injective evaluation keys $\mathsf{ek} \leftarrow \mathsf{Inj}(1^\kappa)$, $\mathsf{ek}' \leftarrow \mathsf{Inj}'(1^\kappa)$. Consider the circuit $C(r, X_i, X_j)$ that has the key K hard-coded (see Figure 4) and compute $\widehat{C} = i\mathcal{O}(C)$. Set $params = (\widehat{C}, \mathsf{ek}, \mathsf{ek}')$.

Inputs: r, X_i, X_j.
Constant: $K, \mathsf{ek}, \mathsf{ek}'$
If $f_{\mathsf{ek}}(r) = X_i$ or $f_{\mathsf{ek}}(r) = X_j$, output $\mathsf{pPRF.eval}(K, (f'_{\mathsf{ek}'}(X_i), f'_{\mathsf{ek}'}(X_j))$;
Else output $\perp$.

Fig. 4. The Circuit $C(r, X_i, X_j)$

- $\mathsf{NIKE.Gen}(1^\kappa, params, ID_i)$: To compute the key pair of an user ID_i, sample $sk_1^i \xleftarrow{\$} \{0,1\}^\kappa$. Consider the circuit $C_i(sk_t, t)$ that has the keys ek, ek', the base secret key sk_1^i, and the obfuscated circuit $\widehat{C}$ (which is part of $params$) hard-coded (See Figure 5) and compute $\widehat{C_i} = i\mathcal{O}(C_i)$. Set the public key as $pk^i = \widehat{C_i}$.
- $\mathsf{NIKE.Upd}(1^\kappa, sk_t^i)$: On input of the user $ID_i's$ secret key sk_t^i at time period t, computes sk_{t+1}^i, the secret key for the next time period $t+1$. The instantiation of the update function is mentioned below.

[10] A (κ, k, m)-lossy function maps an input from $x \in \{0,1\}^\kappa$ to an output $y \in \{0,1\}^m$. In the lossy mode, the image size of the function is at most $2^{\kappa-k}$ with high probability.

Inputs: sk_t, t.
Constants: sk_1^i, ek, ek', $\hat{C}$, T.

1. Check if $t \le T$. If not, output $\perp$.
2. Update $sk_t^i = \mathsf{NIKE.Upd}^{t-1}(sk_1^i)$.
3. Compute $X_t^i = f_{\mathsf{ek}}(sk_t^i)$ and $X_t^j = f_{\mathsf{ek}}(sk_t)$.

Output the shared key $shk_t^{ij} = \widehat{C}(sk_t, X_t^i, X_t^j)$.

Fig. 5. Circuit $C_i(sk_t, t)$

- $\mathsf{NIKE.Key}(ID_i, pk^i = \widehat{C_i}, ID_j, sk_t^j)$: The user ID_j runs the obfuscated circuit $\widehat{C_i} = i\mathcal{O}(C_i)$ on inputs the secret key sk_t^j corresponding to time period t to obtain the shared key shk_t^{ij} at time period t.

Note on Update Function: The update function $\mathsf{NIKE.Upd}$ is one which takes a secret key of the current period and produces a new secret key. As defined in the security model, the adversary can issue leakage queries provided the keys are α-entropic conditioned on the set of all leakage queries. It is not hard to see that the update function should necessarily satisfy the one-wayness property, essentially guaranteeing the non-invertibility of the earlier keys once the secret key is exposed. Interestingly, for the above construction, we can abstract away the update function to any *entropic leakage resilient one-way function*, i.e., $\mathsf{NIKE.Upd}(\cdot) = g(\cdot)$, where $g : \{0,1\}^\kappa \to \{0,1\}^\kappa$ be a α-entropic leakage-resilient one-way function (α-ELR-OWF). The definition of entropic leakage-resilient OWF is given in Section 3.3.

Correctness. It is not hard to see that both the parties ID_i and ID_j end up with the *same* shared key.

Shared key computation by party P_i: Party P_i computes the shared key as:

$$\begin{aligned}
shk_t^{ij} &= \mathsf{NIKE.Key}(ID_j, pk^j = \widehat{C_j}, ID_i, sk_t^i)\\
&= \widehat{C_j}(sk_t^i, (X_t^i, X_t^j))\\
&= \mathsf{pPRF.eval}\big(K, f'_{\mathsf{ek}'}(X_t^i), f'_{\mathsf{ek}'}(X_t^j)\big)\\
&= \mathsf{pPRF.eval}\big(K, f'_{\mathsf{ek}'}(f_{\mathsf{ek}}(sk_t^i)), f'_{\mathsf{ek}'}(f_{\mathsf{ek}}(\mathsf{NIKE.Upd}^{t-1}(sk_1^j)))\big)\\
&= \mathsf{pPRF.eval}\big(K, f'_{\mathsf{ek}'}(f_{\mathsf{ek}}(\mathsf{NIKE.Upd}^{t-1}(sk_1^i))), f'_{\mathsf{ek}'}(f_{\mathsf{ek}}(\mathsf{NIKE.Upd}^{t-1}(sk_1^j)))\big)
\end{aligned}$$

Shared key computation by party P_j: Party P_j computes the shared key as:

$$\begin{aligned}
shk_t^{ij'} &= \mathsf{NIKE.Key}(ID_i, pk^i = \widehat{C_i}, ID_j, sk_t^j)\\
&= \widehat{C_i}(sk_t^j, (X_t^i, X_t^j))\\
&= \mathsf{pPRF.eval}\big(K, f'_{\mathsf{ek}'}(X_t^i), f'_{\mathsf{ek}'}(X_t^j)\big)\\
&= \mathsf{pPRF.eval}\big(K, f'_{\mathsf{ek}'}(f_{\mathsf{ek}}(\mathsf{NIKE.Upd}^{t-1}(sk_1^i))), f'_{\mathsf{ek}'}(f_{\mathsf{ek}}(sk_t^j))\big)\\
&= \mathsf{pPRF.eval}\big(K, f'_{\mathsf{ek}'}(f_{\mathsf{ek}}(\mathsf{NIKE.Upd}^{t-1}(sk_1^i))), f'_{\mathsf{ek}'}(f_{\mathsf{ek}}(\mathsf{NIKE.Upd}^{t-1}(sk_1^j)))\big)
\end{aligned}$$

Hence, we can see that shared keys computed by both parties P_i and P_j corresponding to time period t are *same*, i.e., $shk_t^{ij} = shk_t^{ij'}$.

Instantiations. Our $\mathsf{FS\text{-}EBLR}$ NIKE construction from above can be instantiated based on the recent construction of $i\mathcal{O}$ from well-founded assumptions [28]. One can construct lossy functions from DDH or LWE [34]. Besides, we need to rely on the relaxed varaint of the Superfluous Padding Assumption (SuPA). In particular we obtain FS+ECL NIKE from either DDH or LWE along with sub-exponential SXDH on asymmetric bilinear groups, sub-exponential LPN, Boolean PRGs in NC^0 and relaxed SuPA.

5.6 Security Proof

Theorem 2. *Let κ be the security parameter, and $T = T(\kappa)$ be an arbitrary but fixed polynomial. Assume that $i\mathcal{O}$ is an indistinguishability obfuscator for circuits, and the superfluous padding assumption holds for $i\mathcal{O}$. Let LF is an (κ, k, m)-lossy function, LF' is an (κ', k', m')-lossy function where $\kappa' \geq m$, pPRF is a family of puncturable PRFs with image size $\mathcal{Y} = \{0,1\}^y$. Then, Construction 5.5 is a (α, T)-forward-secure entropy-bounded leakage-resilient NIKE in the $\mathcal{EBL}$-$\mathcal{PS}$ model with $\alpha \geq y + rT + r' - 2\kappa$, where $r = (\kappa - k)$, $r' = (\kappa' - k')$, and T denote the total number of time periods supported by the scheme.*

The proof of the above theorem is presented in the full version of our paper. The high level idea of the proof strategy is presented in the introduction.

References

1. M. Abdalla and L. Reyzin. A new forward-secure digital signature scheme. In T. Okamoto, editor, *ASIACRYPT 2000*, volume 1976 of *LNCS*, pages 116–129. Springer, Heidelberg, Dec. 2000.
2. A. Akavia, S. Goldwasser, and V. Vaikuntanathan. Simultaneous hardcore bits and cryptography against memory attacks. In O. Reingold, editor, *TCC 2009*, volume 5444 of *LNCS*, pages 474–495. Springer, Heidelberg, Mar. 2009.
3. J. Alwen, Y. Dodis, and D. Wichs. Leakage-resilient public-key cryptography in the bounded-retrieval model. In S. Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 36–54. Springer, Heidelberg, Aug. 2009.
4. M. Bellare, S. Meiklejohn, and S. Thomson. Key-versatile signatures and applications: RKA, KDM and joint enc/sig. In P. Q. Nguyen and E. Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 496–513. Springer, Heidelberg, May 2014.
5. M. Bellare and S. K. Miner. A forward-secure digital signature scheme. In M. J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 431–448. Springer, Heidelberg, Aug. 1999.
6. M. Bellare, A. O'Neill, and I. Stepanovs. Forward-security under continual leakage. *In 16th International Conference on Cryptology And Network Security*, 2017.
7. C. Boyd and K. Gellert. A modern view on forward security. Cryptology ePrint Archive, Report 2019/1362, 2019. https://eprint.iacr.org/2019/1362.
8. X. Boyen, H. Shacham, E. Shen, and B. Waters. Forward-secure signatures with untrusted update. In A. Juels, R. N. Wright, and S. Vimercati, editors, *ACM CCS 06*, pages 191–200. ACM Press, Oct. / Nov. 2006.

9. E. Boyle, G. Segev, and D. Wichs. Fully leakage-resilient signatures. In K. G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 89–108. Springer, Heidelberg, May 2011.
10. Z. Brakerski, Y. T. Kalai, J. Katz, and V. Vaikuntanathan. Overcoming the hole in the bucket: Public-key cryptography resilient to continual memory leakage. In *51st FOCS*, pages 501–510. IEEE Computer Society Press, Oct. 2010.
11. C. Brzuska and A. Mittelbach. Universal computational extractors and the superfluous padding assumption for indistinguishability obfuscation. Cryptology ePrint Archive, Report 2015/581, 2015. http://eprint.iacr.org/2015/581.
12. R. Canetti, S. Halevi, and J. Katz. A forward-secure public-key encryption scheme. In E. Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 255–271. Springer, Heidelberg, May 2003.
13. M. Chase and A. Lysyanskaya. On signatures of knowledge. In C. Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 78–96. Springer, Heidelberg, Aug. 2006.
14. Y. Dodis, K. Haralambiev, A. López-Alt, and D. Wichs. Cryptography against continuous memory attacks. In *51st FOCS*, pages 511–520. IEEE Computer Society Press, Oct. 2010.
15. Y. Dodis, K. Haralambiev, A. López-Alt, and D. Wichs. Efficient public-key cryptography in the presence of key leakage. In M. Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 613–631. Springer, Heidelberg, Dec. 2010.
16. Y. Dodis, L. Reyzin, and A. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In C. Cachin and J. Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 523–540. Springer, Heidelberg, May 2004.
17. S. Dziembowski, T. Kazana, and D. Wichs. Key-evolution schemes resilient to space-bounded leakage. In P. Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 335–353. Springer, Heidelberg, Aug. 2011.
18. E. S. V. Freire, D. Hofheinz, E. Kiltz, and K. G. Paterson. Non-interactive key exchange. In K. Kurosawa and G. Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 254–271. Springer, Heidelberg, Feb. / Mar. 2013.
19. S. Garg, C. Gentry, S. Halevi, and D. Wichs. On the implausibility of differing-inputs obfuscation and extractable witness encryption with auxiliary input. *Algorithmica*, 79(4):1353–1373, 2017.
20. S. Garg, C. Gentry, A. Sahai, and B. Waters. Witness encryption and its applications. In D. Boneh, T. Roughgarden, and J. Feigenbaum, editors, *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 467–476. ACM, 2013.
21. M. D. Green and I. Miers. Forward secure asynchronous messaging from puncturable encryption. In *2015 IEEE Symposium on Security and Privacy*, pages 305–320. IEEE Computer Society Press, May 2015.
22. F. Günther, B. Hale, T. Jager, and S. Lauer. 0-RTT key exchange with full forward secrecy. In J. Coron and J. B. Nielsen, editors, *EUROCRYPT 2017, Part III*, volume 10212 of *LNCS*, pages 519–548. Springer, Heidelberg, Apr. / May 2017.
23. J. A. Halderman, S. D. Schoen, N. Heninger, W. Clarkson, W. Paul, J. A. Calandrino, A. J. Feldman, J. Appelbaum, and E. W. Felten. Lest we remember: cold-boot attacks on encryption keys. *Commun. ACM*, 52(5):91–98, 2009.
24. J. Håstad, R. Impagliazzo, L. A. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.
25. J. Holmgren. On necessary padding with io. *Cryptology ePrint Archive*, 2015.

26. C.-Y. Hsiao, C.-J. Lu, and L. Reyzin. Conditional computational entropy, or toward separating pseudoentropy from compressibility. In M. Naor, editor, *EUROCRYPT 2007*, volume 4515 of *LNCS*, pages 169–186. Springer, Heidelberg, May 2007.
27. G. Itkis and L. Reyzin. Forward-secure signatures with optimal signing and verifying. In J. Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 332–354. Springer, Heidelberg, Aug. 2001.
28. A. Jain, H. Lin, and A. Sahai. Indistinguishability obfuscation from well-founded assumptions. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 60–73, 2021.
29. J. Katz and V. Vaikuntanathan. Signature schemes with bounded leakage resilience. In M. Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 703–720. Springer, Heidelberg, Dec. 2009.
30. P. C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In N. Koblitz, editor, *CRYPTO'96*, volume 1109 of *LNCS*, pages 104–113. Springer, Heidelberg, Aug. 1996.
31. P. C. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In M. J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 388–397. Springer, Heidelberg, Aug. 1999.
32. H. Krawczyk. Simple forward-secure signatures from any signature scheme. In S. Jajodia and P. Samarati, editors, *ACM CCS 00*, pages 108–115. ACM Press, Nov. 2000.
33. A. B. Lewko, Y. Rouselakis, and B. Waters. Achieving leakage resilience through dual system encryption. In Y. Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 70–88. Springer, Heidelberg, Mar. 2011.
34. X. Li, F. Ma, W. Quach, and D. Wichs. Leakage-resilient key exchange and two-seed extractors. In H. Shacham and A. Boldyreva, editors, *CRYPTO 2020, Part I*, LNCS, pages 401–429. Springer, Heidelberg, Aug. 2020.
35. T. Malkin, D. Micciancio, and S. K. Miner. Efficient generic forward-secure signatures with an unbounded number of time periods. In L. R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 400–417. Springer, Heidelberg, Apr. / May 2002.
36. M. Naor and G. Segev. Public-key cryptosystems resilient to key leakage. In S. Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 18–35. Springer, Heidelberg, Aug. 2009.
37. D. Pointcheval and O. Sanders. Forward secure non-interactive key exchange. In M. Abdalla and R. D. Prisco, editors, *SCN 14*, volume 8642 of *LNCS*, pages 21–39. Springer, Heidelberg, Sept. 2014.
38. A. Sahai and B. Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In D. B. Shmoys, editor, *46th ACM STOC*, pages 475–484. ACM Press, May / June 2014.
39. M. Zhandry. The magic of ELFs. In M. Robshaw and J. Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 479–508. Springer, Heidelberg, Aug. 2016.