Forward-Security Under Continual Leakage with Deterministic Key Updates

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

Abstract. In the setting of continual-leakage (CL) — Brakerski *et al.*, Dodis *et al.*, FOCS 2010 — the secret key of a cryptographic scheme evolves according to time periods; the adversary gets some bounded leakage function of its choice applied to the current secret key in each time period. This model necessitates a *randomized* key update procedure, as otherwise the adversary can leak a future secret key bit by bit over time. Unfortunately, this is a major source of difficulty, for example in handling leakage on updates. On the other hand, the above reason why a randomized key update procedure is required is arguably unsatisfying, since in practice a leakage function will not continually compute the update procedure and leak a future key in whole.

Our goal is to provide a general security model for continual leakage with deterministic key updates, and constructions that improve in various respects on prior work. In fact, as described below we incorporate forward security into our model as well. For our basic security model we take an *entropy-based* approach, leading to a model we call *entropic continual leakage* (ECL). In the ECL model, the adversary is allowed to make a limited total number of leakage queries that, as in CL, can depend arbitrarily on other keys (in particular, we do not completely bar the leakage function from "computing the update procedure"), but an *unlimited* total number of what we call "local" leakage queries. The latter does not decrease computational entropy of other keys. Hence, in some sense, the local leakage queries do not compute the key update procedure.

Another major benefit of allowing deterministic key updates is that we can more readily incorporate forward security (FS) in our constructions, recently pointed out by Bellare *et al.* (CANS 2017) to be an important security hedge in this context. This is because techniques for achieving FS often require deterministic updates. Accordingly, we also introduce the FS+ECL model (which is in fact incomparable to the CL model). We target this enhanced model for our constructions and provide constructions of public-key encryption (based on non-interactive key exchange) and digital signatures (based on identification schemes) that improve over the assumptions or leakage rates of the FS+CL schemes of Bellare *et al.*. These results demonstrate the feasibility of improved constructions in our more realistic model. Finally, as a result of independent interest, we present a public-key encryption scheme in the FS+CL model (with randomized update) that improves on both the assumptions and leakage rates compared to the scheme of Bellare *et al.*

1 Introduction

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

* Institute of Science and Technology Austria. `suvradip.chakraborty@ist.ac.at`
** Dept. of Computer Science, New York University. `karthik@cs.nyu.edu`
*** Dept. of Computer Science, University of Massachusetts Amherst. `adamo@cs.umass.edu`
† Dept. of Computer Science, Indian Institute of Technology Madras. `rangan@cse.iitm.ac.in`

channel attacks where the adversary obtains some leakage about secrets, such as execution time, power consumption, and even sound waves [28, 33, 34]. The cryptographic community has responded by extending the attack model, which had previously only considered black-box access to the algorithms, so that in the extended model the adversary gets some "bounded" leakage about the secrets [3, 9, 32, 38]. Because there is no inherent reason for the leakage to stay bounded throughout the entire lifetime of the key (if the adversary can get a large fraction of the key, what prevents it from completely recovering it?), works further extended the model to consider "continual" leakage (CL) attacks [10, 17]. In this model, the life of a secret key is divided into time periods/epochs, 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 bounded output length applied to the current secret key.

Motivation For Deterministic Updates. In the continual leakage model, the update algorithm needs to be *randomized*; otherwise, simply consider an adversary that request bits of some future secret key, one by one, in earlier rounds (also called *future key pre-computation* attack). Randomization seems like a simple enough requirement, but it leads to a number of issues. For one, a challenging problem has been then handling *leakage on updates* (*i.e.*, leakage on the coins of the update algorithm), which has required complex schemes or heavy-weight tools to address this [15, 19, 35]. Further, it is difficult to integrate with machinery for *forward security* (FS) [6], which is an important hedge against full key exposure in this context [7], since FS schemes typically have deterministic update. In particular, looking ahead it will allow us to use non-tree-based techniques to achieve FS (all constructions in [7] are tree-based), which has various advantages.

However, we observe that the reason above that the update algorithm of a CL resilient scheme requires randomized update is contrived in that the leakage function not only repeatedly computes the update algorithm itself, it leaks an entire future key. We assert that in the real world this would not happen, as exemplified by typical sources of leakage "controlled by nature." Could it not be possible to have deterministic update in the continual leakage setting and achieve reasonable security? What security notion could one aim for?

Prior Work and Our Approach. One approach is to assume that the leakage function lies in a strictly less powerful complexity class than the update function of the scheme. This was previously done for key-evolution schemes by Dziembowski *et al.* [21], who assume space-bounded leakage functions. However, such a requirement is hard to guarantee in practice. For example, timing or cache attacks may be a function of the entire memory. Furthermore, it is complicated in the following sense: If leakage is an arbitrary efficiently computable function of the secret key then leakage from cryptographic operations like signing and encrypting can be captured directly. However, when the leakage function is too weak leakage from such computation has to be captured *separately*.

1.1 The ECL and the FS+ECL Model.

This motivates us to find an approach that more directly captures the above intuition about contrived leakage functions. To this end, we classify leakage functions according to whether, intuitively speaking, they *compute the update function or not*, limiting the amount of the former. Namely, we categorize the leakage functions queried by the adversary into (i) *local* and (ii) *non-local* queries. The above categorization is based on the property of entropy loss of the keys due to the leakage functions. In particular, we ask whether the outcome of a given leakage query (on one key) reduces the entropy of other keys (which are derived from that key using a (deterministic) key update procedure) or not. Very roughly, we call a leakage query that does not reduce the entropy of other keys "local". Of course, to allow deterministic key updates, such a notion of entropy must be computational, as each leakage query *necessarily* reduces information-theoretic entropy of the other (deterministically defined) keys. The formal definition is given in Definition 9. On the other hand, as the name suggests, non-local queries may potentially reduce the entropy of the other keys. In our corresponding model, which we call the "Entropic" Continual Leakage (ECL) model, there is no restriction on the type or the number of such queries the adversary can make over time. On the other hand, to guarantee security we place the following restrictions on these queries as: (i) The output length of each local leakage query is upper bounded by a parameter $\lambda(\kappa)$, where κ is the security parameter, (ii) the set of all non-local leakage queries made by the adversary should not reduce the min-entropy of any key beyond $\alpha(\kappa)$, which is also a parameter of the model. This nevertheless allows the leakage to still have some dependence on future keys as a hedge (note that, if the adversary can make unlimited no. of non-local queries, then achieving security with deterministic update functions is impossible due to the future key pre-computation attack mentioned before).

Incorporating FS. As enabled to allowing deterministic updates, all of our constructions actually achieve a much stronger definition that incorporates another hedge, namely forward security (FS). FS was recently studied as a hedge for continual leakage security by Bellare *et al.* [7]. FS refers to the fact that if the full secret key is recovered in some time period then security in prior time periods is maintained. As argued by [7], it is important to consider forward security in this context, since the update may not happen soon enough in some time period to avoid full leakage of the current secret key. To address this, Bellare *et al.* proposed the Forward security under Continual Leakage (FS+CL) model, where the adversary gets bounded leakage on the current secret key in each time period, and FS in the sense that if full key exposure occurs in one time period then executions in previous time periods remain secure. Note that the FS+CL model is actually stronger than either CL or FS individually.

1.2 Our Contributions.

In this work, we propose the "Entropic" Continual Leakage (ECL) and Forward security under "Entropic" Continual Leakage (FS+ECL) model and present new constructions of public key encryption (PKE) and digital signature schemes in the FS+ECL model.

Besides, we also improve upon the construction of PKE scheme in the FS+CL model (due to [7]) in terms of the required hardness assumptions. We expand on each of these contributions in greater detail below.

The FS+ECL Model. We propose the FS+*E*CL model. The FS+ECL model is very similar to the ECL model (informally discussed above) with the following exception: we require that the security of the underlying primitive be maintained, given the set of local and non-local leakage queries asked by the adversary till the period of break-in/exposure (i.e., when the adversary breaks in and recovers the full secret key of that time period). This is because, after recovering a key in full the adversary can compute the future keys by itself. This model raises some interesting questions: Can one construct schemes with better efficiency or under more standard assumptions in the FS+ECL model compared to the FS+CL constructs of Bellare *et al.* [7]? Intuitively, this should be the case because deterministic updates allow exploiting a host of FS machinery (optimized FS schemes often have deterministic update) already developed. We show that improvements are possible in the FS+ECL setting for both signatures and encryption.

PKE scheme in the FS+ECL Model. We define and construct key evolving encryption (KEE) scheme in the FS+ECL model. We obtain these results by considering an *intermediate* primitive, namely *non-interactive key exchange* (NIKE) in the FS+ECL model, and then show a generic transformation from a FS+ECL-secure NIKE scheme to a FS+ECL-secure PKE scheme. The resulting PKE scheme inherits the *same* leakage rate as of the underlying NIKE scheme. Our security model for FS+ECL NIKE can be seen as a generalization of the model of forward-secure NIKE (FS-NIKE) of Pointcheval and Sanders [40]. However, as we discuss later (see Section 1.3), defining an appropriate model for leakage-resilient NIKE is *non-trivial* and requires care.

We show a construction of FS+ECL-secure NIKE from indistinguishability obfuscation ($i\mathcal{O}$) and one-way functions, enjoying optimal leakage rate, i.e., $1 - o(1)$. Similar to [40], our construction of FS+ECL NIKE supports an a-priori *bounded* (but an arbitrary polynomial) number of time periods. While our current instantiation of FS+ECL NIKE rely on $i\mathcal{O}$, there is hope that this can also be instantiated under weaker assumptions in the future. However, we remark that, before this work, construction of even FS-NIKE was not known from $i\mathcal{O}$. The previously known construction of FS-NIKE by [40] relied on graded multilinear maps. However, the candidate constructions of such maps are prone to various attacks (see [12, 13, 37]). Moreover, it is not clear how to effectively leverage this construction to achieve leakage-resilience.

Signature scheme in the FS+ECL Model. We define and construct key evolving signature (KES) scheme in the FS+ECL model. To this end, we introduce another *intermediate* primitive, namely *identification* (ID) scheme in the FS+ECL model, and then show how to transform such an ID scheme to a FS+ECL-secure signature scheme.

We show a construction of FS+ECL-secure ID scheme from the hardness of the (standard) RSA problem, enjoying optimal leakage rate, i.e., $1 - o(1)$. Our ID scheme supports an a-priori bounded (but an arbitrary polynomial) number of time periods. Finally, we show how to use the Fiat-Shamir transform to convert an FS+ECL-secure

ID scheme to a FS+ECL-secure signature. The resulting signature scheme is secure in the Random Oracle (RO) model and has leakage rate half as that of the underlying FS+ECL-secure ID scheme, i.e., $1/2 - o(1)$.

PKE scheme in the FS+CL Model [7]. Finally, as a result of independent interest, we consider the problem of constructing an FS+CL encryption scheme. Bellare *et al.* [7] showed a construction of FS+CL secure encryption scheme from extractable witness encryption (EWE), which is believed to be a suspect assumption [26]. It was left as an open problem in [7] to construct a FS+CL-secure PKE scheme from standard assumptions. We significantly improve upon the construction of [7] and show how to construct a FS+CL-secure PKE scheme from standard assumptions (namely, *static* assumptions over composite-order bilinear maps). Besides, the leakage rate of our construction is also *optimal*, i.e., $1 - o(1)$, whereas the PKE scheme of [7] could only tolerate a leakage rate of $1/\mathsf{poly}(\kappa)$, where κ is the security parameter.

1.3 Technical Overview.

Modelling FS+ECL-secure NIKE. We observe that constructing a NIKE scheme even in the setting of bounded leakage (letting alone CL or ECL model) is *impossible* in general. This is because, the adversary while leaking from the secret key of one party involved in the NIKE protocol can encode the (description of the) shared-key derivation function along with the public key of the other party hard-coded in the function to leak directly from the shared key. To bypass this impossibility result, we must impose some meaningful restrictions on the class of allowable leakage functions.

One way to circumvent the above impossibility result would be to consider the *non-adaptive* leakage model, where the adversary needs to specify the leakage functions before seeing the public parameters *params* (and hence the public keys of the parties). However, this leakage model is not satisfactory, as typically one assumes that if a value(s) is public (public key, parameters), everyone, including the adversary will be able to see it at all times. Also, as discussed in [42], non-adaptive leakage model is a natural choice for (stateless) cryptographic schemes that do not allow to evolve the secret state (e.g., PRFs, PRPs). In this work, we consider key-evolving primitives and hence the non-adaptive leakage model may not be well suited for these applications. Besides, a construction of NIKE in the non-adaptive leakage model would also lead to constructions of PKE scheme in the non-adaptive leakage model.

Instead, we propose an alternate model for leakage-resilient NIKE which provides a better "middle" ground, in the sense that, the choice of the leakage functions can be adaptive and can even depend on the public keys of the users, subject to the following restriction. In our model, we impose the restriction that the leakage functions (queried by the adversary) in any of their invocations cannot take as argument the keys of both users participating in the NIKE protocol. In other words, the leakage functions can be of the form $f(x_i, s_i, params)$, or $f(x_j, s_j, params)$, but not $f(x_i, s_j, params)$ or $f(x_j, s_i, params)$ respectively, where (x_i, s_i) and (x_j, s_j) are the key pairs of any two (distinct) users i and j respectively. In other words, the leakage functions are allowed to work with the keys of a *single user* at a time. However, we stress that the choice of the leakage functions can depend on both the public keys. At first glance, it might not be clear why such a model for NIKE will be useful. However, we show that this leakage model for NIKE is already powerful enough to construct FS+ECL CCA-secure PKE scheme (in the adaptive leakage model). Finally, we note that there is no efficient way for the challenger (in the security game) to check that the leakage functions submitted by the adversary adheres to the above specification[1]. Instead, we define a class of admissible adversaries (see Section 4.3) and require that the security of the NIKE protocol only holds with respect to the adversaries from this class.

Constructing FS+ECL-secure NIKE Scheme. The starting point of our construction of FS+ECL-secure NIKE scheme is the multi-party NIKE construction of Boneh and Zhandry [8] (henceforth referred to as the BZ protocol) based on indistinguishability obfuscation ($i\mathcal{O}$). The main idea of the (two-party version of the) BZ protocol is as follows: Each user samples a random seed s of a length-doubling PRG G as its secret key, and announces $x = G(s)$ as its public key. 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 P (which is included as part of the common reference string) which works as follows: P simply checks if $x_i \stackrel{?}{=} G(s_i)$. If so, it runs $\mathsf{PRF}_K(x_i, x_j)$ (where the PRF key K is embedded inside the obfuscated program).

It is easy to see that the BZ protocol is not even *forward-secure*. This is because, if the secret key s is updated, the public key also changes. We now describe how to modify the above construction to achieve FS+ECL security. The main idea of our construction is as follows: Similar to the BZ protocol, the (initial) secret key of each user i in our construction is also a random seed s_1^i of a length doubling PRG G. However, the public key of each party i is now an obfuscated program P_i, which has the initial/root secret key s_1^i (corresponding to time period 1) of that party i and a PRF key K embedded in it. To generate the shared key with an user j corresponding to some time period (say t), user i runs the obfuscated program P_j (public key of user j) with input s_t^i (secret key of i corresponding to time period t), where s_t^i is obtained from s_1^i using a key update procedure Upd[2]. The program P_j does the following: (1) Compute $s_t^j = \mathsf{Upd}^{t-1}(s_1^j)$, where the root key s_1^j of user j is hardwired inside P_j and Upd^{t-1} corresponds to running Upd sequentially $t-1$ times, (2) then it applies length doubling PRG on the (updated) secret keys of both the parties i and j to obtain $X_t^i = G(sk_t^i)$ and $X_t^j = G(sk_t^j)$, and (3) runs $\mathsf{PRF}_K(X_t^i, X_t^j)$ (using the key K embedded inside P_j) to obtain the shared key (for time period t). A salient feature of our construction is that, it does not require a trusted setup like a common reference string (unlike the BZ protocol). Also, the size of the public key of each user is *succinct*, in the sense that it depends only logarithmically in the number of time periods supported by the scheme.

The security proof of our construction follows the punctured programming technique of Sahai and Waters [41]; however, with a more subtle analysis. The main problem is

[1] For e.g., if the leakage function f is an obfuscation of some circuit.

[2] Looking ahead, in our construction Upd will be an entropic leakage-resilient one-way function (ELR-OWF) [9,18]

that, we cannot simply switch the outputs of the PRG G (i.e., the values X_t^i and X_t^j) to be uniformly random, since the adversary obtains leakage from each of the keys s_t^i and s_t^j via local and non-local queries. However, the entropic condition of the FS+ECL model ensures that each of the secret keys are entropic, even given the leakage. The application of the PRG on these entropic keys guarantees that the output of the PRG has certain computational min-entropy or pseudo-entropy in it (to be precise, metric* entropy). By appropriately setting our parameters, we can ensure that the (computationally) entropy of the output of this (length doubling) PRG is more than the entropy of its input. The proof now follows from the standard puncturing technique.

From FS+ECL-secure NIKE to FS+ECL-secure PKE scheme. We provide a generic transformation from a FS+ECL-secure NIKE scheme to a FS+ECL-secure PKE scheme. The transformation follows the blueprint of [24], who showed how to transform a secure NIKE scheme (in an appropriate security model) to a CCA-secure PKE scheme, additionally using one-time signatures. We show how to appropriately modify their proof technique to deal with forward security and entropic continual leakage simultaneously.

Constructing FS-ECL ID and Signature Schemes. Our definition of FS+ECL-secure ID schemes generalizes the prior definitions of ID schemes, which were either forward-secure [1,6] or only leakage-resilient [4]. Conceptually, we view our construction of FS+ECL-secure ID scheme as a modular combination of two basic steps: (a) prove the (entropic) leakage-resilience of the generalized Guillou-Quisquater (GQ) identification scheme [16, 27] based on the hardness of the RSA problem, and (b) use a variant of the Itkis-Reyzin (IR) transform to convert the generalized GQ scheme to a forward-secure version of itself. We actually show that the generalized GQ scheme is a secure 3-round public-coin Σ-protocol satisfying two additional properties – (a) each statement (or theorem) has exponentially many witnesses, and (b) the uncertainty of the witness conditioned on the statement is high. These properties help us to prove entropic leakage-resilience.

Finally, we show how to transform the FS+ECL-secure ID scheme to a FS+ECL-secure signature scheme using the generalized Fiat-Shamir (FS) transform. This shows the applicability of the FS transform even in the FS+ECL setting.

FS+CL PKE Scheme from Simpler Assumptions As a result of independent interest, we consider the problem of constructing a FS+CL-secure PKE scheme (as proposed by Bellare *et al.* [7]). As an intermediate step to achieving this, we abstract out a new notion of *continuous leakage-rate binary tree encryption* (CLR-BTE), which is a continuous leakage-resilient analogue of the notion of binary tree encryption (BTE), introduced by Canetti, Halevi and Katz (CHK) [11]. We also observe that the construction of CLR-BTE follows in a straightforward manner from the CLR-HIBE scheme of Lewko et al. [36] (which is based on static assumptions over composite-order bilinear groups). For appropriate choice of parameters, the CLR-HIBE scheme achieves the optimal leakage rate of $1 - o(1)$. Finally, we show how to transform a CLR-BTE scheme to a FS+CL-secure PKE scheme using the Canetti-Halevi-Katz (CHK) transform.

This approach of constructing FS+CL secure encryption scheme was already suggested in [7]. However, it was intuitively claimed in [7] that the above approach does not work, due to the following reason: "The problem is that FS+CL security of the resulting scheme requires that multiple nodes of the BTE construction can be leaked on jointly, whereas the CL security of HIBE only buys us leakage on each such node individually." Surprisingly, we prove the contrary and show that, indeed, it is possible to *simulate* the joint leakage by just leaking on a *single* node. This requires a careful analysis of the CHK transform in the FS+CL setting.

2 Preliminaries

2.1 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 $\mathcal{X}$. With $X \sim Y$, we denote that that X and Y have the same distribution. 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. Vectors are written in boldface. 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 $\delta^D(X, Y) = |\mathbb{E}[D(X)] - \mathbb{E}[D(Y)]|$. We denote the size of a circuit D as $|D|$. We denote by $\mathcal{D}_s^{\mathsf{det},\{0,1\}}$ the class of all deterministic circuits of size s with boolean output $\{0,1\}$, and $\mathcal{D}_s^{\mathsf{rand},\{0,1\}}$ denote the class of all probabilistic (randomized) circuits of size s with boolean output.

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

This is a standard notion of entropy used in cryptography, since it measures the worst-case predictability of X.

Definition 2 (Conditional Min-entropy [20]). *The average-conditional min-entropy of a random variable X conditioned on a (possibly) correlated variable Z, denoted as $\widetilde{\mathrm{H}}_\infty(X|Z)$ is defined as*

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

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

Lemma 1 (Chain Rule for min-entropy [20]). *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 [4].

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 [29, 31]). *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 $\forall\, \mathcal{D} \in \mathcal{D}^{\mathsf{rand},\{0,1\}}_s$, $\delta^{\mathcal{D}}(\mathcal{X},\mathcal{Y}) \leq \epsilon$.*
For a distribution $(\mathcal{X},\mathcal{Z})$ 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 joint distribution $(\mathcal{Z},\mathcal{Y})$ such that $\widetilde{\mathrm{H}}_\infty(\mathcal{Z}|\mathcal{Y}) \geq k$, and $\forall\, \mathcal{D} \in \mathcal{D}^{\mathsf{rand},\{0,1\}}_s$, $\delta^{\mathcal{D}}\big((\mathcal{X},\mathcal{Y}),(\mathcal{Z},\mathcal{Y})\big) \leq \epsilon$.*

In the above definition, if the distinguisher $\mathcal{D}$ is drawn from the set $\mathcal{D}^{\mathsf{det},\{0,1\}}_s$, it gives rise to another notion of HILL entropy, called the "HILL-star" entropy, denoted by $\mathrm{H}^{\mathsf{HILL}^*}_{\epsilon,s}$. It has been proved in [22, 25] that both these notions of HILL and "HILL-star" entropy are essentially equivalent up to a very small additive loss in the circuit size.

Switching the quantifiers of $\mathcal{Y}$ and $\mathcal{D}$ in the definition of HILL entropy gives us the following, weaker notion of "Metric entropy", as defined in [5].

Definition 4 (Metric entropy [5, 25]). *A distribution $\mathcal{X}$ has* **Metric entropy** *at least k, denoted by $\mathrm{H}^{\mathsf{Metric}}_{\epsilon,s}(X) \geq k$, if $\forall\, \mathcal{D} \in \mathcal{D}^{\mathsf{rand},\{0,1\}}_s$, there exists a distribution $\mathcal{Y}$ with $\mathrm{H}_\infty(\mathcal{Y}) \geq k$, and $\delta^{\mathcal{D}}(\mathcal{X},\mathcal{Y}) \leq \epsilon$.*

For a distribution $(\mathcal{X},\mathcal{Z})$ we say that $\mathcal{X}$ has **conditional metric entropy** *at least k conditioned on $\mathcal{Y}$, denoted $\mathrm{H}^{\mathsf{Metric}}_{\epsilon,s}(X|Y)$, if $\forall\, \mathcal{D} \in \mathcal{D}^{\mathsf{rand},\{0,1\}}_s$, there exists a joint distribution $(\mathcal{Z},\mathcal{Y})$ such that $\widetilde{\mathrm{H}}_\infty(\mathcal{Z}|\mathcal{Y}) \geq k$, and $\delta^{\mathcal{D}}\big((\mathcal{X},\mathcal{Y}),(\mathcal{Z},\mathcal{Y})\big) \leq \epsilon$.*

Similar to HILL entropy, drawing the distinguisher $\mathcal{D}$ from $\mathcal{D}_s^{\mathsf{det},\{0,1\}}$ gives the notion of "metric-star" entropy, denoted by $\mathrm{H}_{\epsilon,s}^{\mathsf{Metric}^*}(X)$.

A chain-rule analogous to the chain rule for min-entropy was shown in [25] for Metric* entropy. Intuitively, the theorem says that the quality and quantity of entropy reduce by the number of leakage values.

Theorem 1. [25] *Let X and Y be discrete random variables. Then:*

$$\mathrm{H}_{\epsilon|Y|,s'}^{\mathsf{Metric}^*}(X|Y) \geq \mathrm{H}_{\epsilon,s}^{\mathsf{Metric}^*}(X) - \log|Y|.$$

Metric vs. HILL. We will use the fact that Metric* entropy implies the same amount of HILL entropy, albeit with a loss in quality. This was proved for the unconditional case by [5] and for the conditional case by [25].

Theorem 2. [5] *Let X be a discrete distribution over a finite set $\mathcal{X}$. For every $\epsilon, \epsilon_{\mathsf{HILL}} > 0$, $\epsilon' \geq \epsilon + \epsilon_{\mathsf{HILL}}$, k and s if $\mathrm{H}_{\epsilon,s}^{\mathsf{Metric}^*}(X) \geq k$, then $H_{\epsilon',s_{\mathsf{HILL}}}^{\mathsf{HILL}}(X) \geq k$, where $s_{\mathsf{HILL}} = \Omega(\frac{\epsilon_{\mathsf{HILL}}^2 s}{\log|\mathcal{X}|)})$.*

Theorem 3. [25] *For any joint distribution $(X, Z) \in \mathcal{X} \times \mathcal{Z}$, and every $\epsilon, \delta > 0$, $\epsilon'' \geq \epsilon + \delta$, k and s if $\mathrm{H}_{\epsilon,s}^{\mathsf{Metric}^*}(X|Z) \geq k$, then $H_{\epsilon'',s'_{\mathsf{HILL}}}^{\mathsf{HILL}}(X|Z) \geq k$, where $s'_{\mathsf{HILL}} = \Omega(\frac{s\delta^2}{\log|\mathcal{X}|+\log|\mathcal{Z}|)})$.*

2.3 Puncturable Pseudo-Random Functions

In this section, we define the syntax and security properties of a puncturable pseudo-random function family. We follow the definition given in [30].

A *puncturable* family of PRF $\mathsf{pPRF} : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ is given by a triple of polynomial time algorithms ($\mathsf{pPRF.setup}$, $\mathsf{pPRF.puncture}$, $\mathsf{pPRF.eval}$) and equipped with an additional (punctured) key space $\mathcal{K}_p$ defined as follows:

- $\mathsf{pPRF.setup}(1^\kappa)$: This is a randomized algorithm that takes the security parameter κ as input and outputs a description of the key space $\mathcal{K}$, the punctured key space $\mathcal{K}_p$ and the PRF pPRF.
- $\mathsf{pPRF.puncture}(K, x)$: This is also a randomized algorithm that takes as input a (master) PRF key $K \in \mathcal{K}$, and an input $x \in \mathcal{X}$, and outputs a key $K_x \in \mathcal{K}_p$, often denoted as the punctured key (punctured at the point x). Without loss of generality, we can think of $F.\mathsf{puncture}$ as a deterministic algorithm also. This is because we can de-randomize the algorithm by generating its random bits using a PRF keyed by a part of the master key K and given the point x as input.
- $\mathsf{pPRF.eval}(K_x, x')$: This is deterministic algorithm that takes as input the punctured key $K_x \in \mathcal{K}_p$, and an input $x' \in \mathcal{X}$. Let $K \in \mathcal{K}$, $x \in \mathcal{X}$ and $K_x \leftarrow \mathsf{pPRF.puncture}(K, x)$. The correctness guarantee stipulates that:

$$\mathsf{pPRF.eval}(K_x, x') = \begin{cases} \mathsf{pPRF}(K, x) & \text{if } x \neq x' \\ \perp & \text{otherwise} \end{cases}$$

Security of Puncturable PRFs: The security of puncturable PRFs is depicted by a game between a challenger and an adversary. The security game consists of the following four stages:

1. **Setup Phase:** The challenger chooses uniformly at random a (master) PRF key $K \in \mathcal{K}$ and a bit $b \in \{0,1\}$.
2. **Evaluation Query Phase:** In this phase the adversary $\mathcal{A}$ queries a point $x \in \mathcal{X}$. The challenger sends back the evaluation $\mathsf{pPRF}(K, x)$ to $\mathcal{A}$. These queries can be made arbitrarily and adaptively by $\mathcal{A}$ polynomially many times. Let $E \subset \mathcal{X}$ be the set of evaluation queries.
3. **Challenge Phase:** In this phase, the adversary $\mathcal{A}$ chooses a challenge point $x^* \in \mathcal{X}$. The challenger computes $K_{x^*} \leftarrow \mathsf{pPRF.puncture}(K, x^*)$. If bit $b = 0$, $\mathcal{C}$ sends back $(K_{x^*}, \mathsf{pPRF}(K, x^*))$. Else, the challenger samples a uniformly random $y^* \leftarrow \mathcal{Y}$, and sends back (K_{x^*}, y^*) to $\mathcal{A}$.
4. **Guess Phase:** $\mathcal{A}$ outputs a guess b' for the bit b chosen by the challenger.

The advantage of $\mathcal{A}$ in the above game is defined by:

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{pPRF}}(\kappa) = \Pr\big[b' = b \,\big|\, x^* \leftarrow \mathcal{A}^{\mathsf{pPRF.eval(K,\cdot)}}(\kappa, K_x^*) \ \wedge \ x^* \notin E\big].$$

Definition 5. *The punctured PRF* pPRF *is said to be secure if for all PPT adversaries* $\mathcal{A}$ *participating in the above game,* $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{pPRF}}(\kappa)$ *is negligible in* κ.

2.4 Indistinguishability Obfuscation

A uniform PPT machine $i\mathcal{O}$ is called an indistinguishability obfuscator for a circuit class $\{\mathcal{C}_\kappa\}_{\kappa \in \mathbb{N}}$ if it satisfies the following conditions:

- (*Functionality Preserving*). For all security parameters $\kappa \in \mathbb{N}$, for all $C \in \mathcal{C}_\kappa$, for all inputs x, we have that:

$$\Pr\big[C'(x) = C(x) : C' \leftarrow i\mathcal{O}(\kappa, C)\big] = 1$$

- (*Indistinguishability of Obfuscation*). For any (not necessarily uniform) PPT adversaries *Samp*, $\mathcal{D}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that the following holds: if for all security parameters $\kappa \in \mathbb{N}$, $\Pr\big[\forall x, C_0(x) = C_1(x) : (C_0, C_1, \mathsf{st}) \leftarrow Samp(\kappa)\big] > 1 - \mathsf{negl}(\kappa)$, then we have:

$$\Big|\Pr\big[\mathcal{D}(\mathsf{st}, i\mathcal{O}(\kappa, C_0)) = 1 : (C_0, C_1, \mathsf{st}) \leftarrow Samp(\kappa)\big] - \Pr\big[\mathcal{D}(\mathsf{st}, i\mathcal{O}(\kappa, C_1)) = 1 : (C_0, C_1, \mathsf{st}) \leftarrow Samp(\kappa)\big]\Big| \leq \mathsf{negl}(\kappa).$$

  where the probability is over the coins of $\mathcal{D}$ and $i\mathcal{O}$.

We remark that the algorithms *Samp* and $\mathcal{D}$ pass state st, which can equivalently be viewed as a single stateful algorithm $\mathcal{B} = (Samp, \mathcal{D})$.

2.5 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 [17] 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 6. [17]. 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 [17], 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 7. (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 [18], 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 $\lambda(\kappa) = n(\kappa) - m(\kappa) - \omega(\log \kappa)$.

2.6 Σ-Protocols

A Σ-protocol for an NP relation $\mathcal{R}$ is a special form of *3-move honest verifier zero knowledge* protocol, that enables a prover to prove knowledge to a witness w associated to a statement $x \in \mathcal{L}_{\mathcal{R}}$, without revealing any additional information about the witness W.

Let $\mathcal{R} = \{(x, w)\}$ be an efficiently computable binary relation, with $|w| \leq p(|x|)$, for some polynomial $p = p(\kappa)$. Also let $L_{\mathcal{R}} = \{x : (w, x) \in \mathcal{R}\}$ be the language corresponding to the relation $\mathcal{R}$. A Sigma protocol $\Pi = (P, V)$ is a 3-round interactive protocol between a prover P (holding a private input w corresponding to a statement $x \in L_{\mathcal{R}}$) and a verifier V (holding the statement x). The protocol proceeds as follows: (i) the prover's first message (also called *commitment*) is denoted by $a = P(x, w)$; (ii) the verifier then chooses a random *challenge* $c \in_R \mathsf{Ch}$, where Ch denotes the challenge space, (iii) the prover's second message (also called *response*) is denoted by $z = P(x, w, a, c)$. The tuple (a, c, z) is known as a transcript/proof. We write $V(x, a, c, z) = 1$ *iff* verifier V accepts, and we say that the transcript (a, c, z) is *accepting* for x. A Σ-protocol is *public-coin*, which means that the challenge c is chosen uniformly at random by the verifier V, without having to store any private information about it. We now formally define the properties to be satisfied by a Σ-protocol.

Definition 8. A protocol $\Pi = (P, V)$ is a three-round public-coin protocol satisfying the following properties:

- **Completeness:** *If P and V follow the protocol on input x and private input w to P where $(x, w) \in \mathcal{R}$, then V always accepts.*
- **Special soundness:** *There exists a polynomial time algorithm (or knowledge extractor) K, that on input x and a pair of accepting transcripts $(a, c, z), (a, c', z')$ for x, where $c' \neq c$, outputs w such that $(x, w) \in \mathcal{R}$.*
- **Perfect honest verifier zero knowledge (HVZK):** *There exists a probabilistic polynomial time simulator* Sim *such that the following two distributions are identically distributed:*

$$\Big\{\mathsf{Sim}(x, c)\Big\}_{x \in L_{\mathcal{R}}, c \in_R \mathsf{Ch}} \equiv \Big\{\langle P(x, w), V(x, c)\rangle\Big\}_{x \in L_{\mathcal{R}}, c \in_R \mathsf{Ch}}$$

where $\mathsf{Sim}(x, c)$ *denotes the output of the simulator upon input x and c, and $\langle P(x, w), V(x, c)\rangle$ denotes the output transcript of an execution between P and V, where P has input (x, w), V has input x, and V's random tape (challenge) equals c.*

As shown in [14], HVZK implies witness indistinguishability. The authors in [4] rephrased this property in a slightly different way. In particular, they showed that the oracle access to a prover $P(x, w)$ does not decrease the min-entropy of w in any experiment in which x is given to the predictor. We also follow this formulation in our work.

Lemma 2. [4]. *Let (P, V) be a HVZK protocol for the relation $\mathcal{R}$, (X, W) be random variables over $\mathcal{R}$, and let $\mathcal{A}$ be a PPT adversary. Let $\mathcal{E}_1$ be an arbitrary experiment in which $\mathcal{A}$ is given X at the start of the experiment, and let $\mathcal{E}_2$ be the same as $\mathcal{E}_1$, except that $\mathcal{A}$ is also given oracle access to $P(x, w)$ throughout the experiment. Then $\widetilde{\mathrm{H}}_\infty(W|\mathcal{E}_2) = \widetilde{\mathrm{H}}_\infty(W|\mathcal{E}_1)$.*

2.7 Representation problem.

Let $\mathcal{L} \in \mathbf{NP}$ be a language with an efficiently computable binary relation $\mathcal{R} \subset \{0,1\}^* \times \{0,1\}^*$, i.e, $x \in \mathcal{L}$ if and only if there exists w such that $(x, w) \in \mathcal{R}$ and $|w| = \mathsf{poly}(|x|)$. The value w is called a witness for the statement/theorem $x \in \mathcal{L}$. We say that a representation problem for the relation $\mathcal{R}$ is (t, ε)-hard if for all PPT adversaries running in time t, we have:

$$\mathsf{Pr}[w \neq w' \wedge (x, w), (x, w') \in \mathcal{R} : \ (x, w, w') \leftarrow \mathcal{A}(1^\kappa)] \leq \varepsilon$$

In most of the cases, the hardness of the representation problem for $\mathcal{R}$ is equivalent to the hardness of the underlying relation $\mathcal{R}$. We will require the hardness of the RSA representation problem as defined below:

RSA Representation problem. Let $N = p \cdot q$, where p and q are prime numbers. Let (e, d) be chosen in such a way that $e \cdot d = 1 \mod \phi(N)$, and e is a prime. Consider the language $\mathcal{L}_{\mathsf{RSA}} = \{(g_1, \cdots, g_\ell, h) : \exists (\rho, \omega_1, \cdots, \omega_\ell) \in \mathbb{Z}_N^* \times \mathbb{Z}_e^\ell \text{ s.t. } h = \prod_{i=1}^{\ell} g_i^{\omega_i} \cdot \rho^e \mod N\}$, where $(g_1, \cdots, g_\ell)$ are generators of a prime-order cyclic subgroup of $\mathbb{Z}_N^*$. We say that the tuple $w = (\rho, \omega_1, \cdots, \omega_\ell)$ is a representation of h. The ℓ-representation problem asks to compute two different representations w, w' for some $x = (g_1, \cdots, g_\ell, h) \in \mathcal{L}_{\mathsf{RSA}}$. As shown in [39], the ℓ-representation problem is hard for $\mathcal{L}_{\mathsf{RSA}}$ if and only if the RSA problem[3] is hard in $\mathbb{Z}_N^*$.

3 The Entropic Continual Memory Leakage (ECL) Model

As discussed before, no cryptographic primitive can be secure against continual leakage if it has *deterministic* update/ key-evolution procedure. To this end, we introduce a variant of continual memory leakage (CL) model, which we call the *entropic continual memory leakage* (ECL) model. Similar to the CL model, the ECL model also follows a key-evolving paradigm where the secret key is periodically updated, keeping the public key unchanged. We assume that when the secret key evolves to the next time period, the prior keys are securely erased from the system. Indeed, given the choice of different security models, it might be prudent to look at them through the lens of adversarial classes. In particular, we first present a unified security model that guarantees different levels of security by considering different classes of adversaries. Our unified security framework is powerful enough to capture the existing leakage security models, for e.g., the bounded leakage, continual leakage and entropy bounded leakage models. In addition, this gives rise to some natural intermediate notions, one of which we call the entropic continual leakage (ECL) model. Looking ahead, we will present this approach for the primitives of Key Evolving Signature Schemes (KES) and Key Evolving Encryption (KEE) schemes.

We will now proceed to describe classes of adversaries that capture not only existing notions but allow us to define new intermediate ones as well. These classes are defined via types of leakage functions. These are classified as *local* and *non-local* leakage functions. Informally, a leakage function is local when the adversary does not compute the update function as a part of the function. We find this to be a natural requirement in practice and it is critical in the scenario of deterministic update. This is formally defined in Definition 9.

Before we proceed with the definition, we will introduce some additional notations. We will define the random variable representing the secret key at time i as SK_i. Clearly, $SK_1, \ldots, SK_T$ are the random variables representing the secret keys at time periods $1, \ldots, T$ respectively defined by a key update scheme, i.e., for all $1 \leq i \leq T-1$, SK_{i+1} is obtained from SK_i by using the key update algorithm. Let L_i be the leakage query

[3] The RSA problem is: Given (N, u, e), such that $u = \rho^e \mod N$, compute ρ.

issued by the adversary at time i. Let $\mathbb{L}_i$ denote the random variable which represents the leakage $L_i(SK_i)$. Also, let R_{L_i} denote the random coins of the adversary corresponding to the function L_i. Further, it is helpful to define Σ_i as the random variable representing the state of the adversary. This captures the cumulative knowledge of the adversary based on all leakage queries and responses, except the query at time period i, and also includes the random coins of the adversary. In other words, $\Sigma_i = \{\{\mathbb{L}_j\}_{j=1}^{T} \setminus \mathbb{L}_i\} \cup \{\{R_{L_j}\}_{j=1}^{T} \setminus R_{L_i}\}$.

Definition 9 (Local Leakage). *A leakage query L_t acting on a key sk_t, at time period t is a local leakage query if any key $SK_j \neq SK_t$ does not lose its entropy due to $\mathbb{L}_t = L_t(sk_t)$. In other words,*

$$\mathrm{H}(SK_j \,|\, \Sigma_t, \mathbb{L}_t = L_t(sk_t), R_{L_t}) = \mathrm{H}(SK_j \,|\, \Sigma_t)$$

Here, H *is a suitable notion of* computational *entropy.*

In other words, the above definition says that: If the entropy of a key at a time period j is the *same* with and without conditioning on the leakage from time $t \neq j$, then the key and the leakage are *independent.* A local leakage query is one which is independent of every other key.

Remark 1. Before we proceed, we would like to point out that while the above definition of "local leakage" is defined for computational entropy, it can be generalized to include *information-theoretic* notions of entropy. However, for our application of *deterministic* update, it is not hard to see that the notion of entropy needs to be *computational* to avoid a trivial reduction to the bounded leakage model (as explained later).

Now, we present a unified definition of leakage security models based on the allowed adversarial classes.

Admissible adversaries: It is prudent to define different classes of adversaries based on the nature of their leakage queries - based on the length of the output of the leakage functions or the residual entropy of the key(s). More formally, let us denote the maximum number of time periods to be T. Let κ denote the security parameter, and let $\alpha = \alpha(\kappa)$, and $\lambda = \lambda(\kappa)$. We define the following classes of adversaries:

- $\mathcal{A}^{\alpha}$: It is the class of *α-entropic adversaries* where *all* the leakage queries made by $\mathcal{B}$ satisfy the following condition:

$$\forall j \in [T], \quad \mathrm{H}\big(SK_j \,|\, \mathbb{L}_1, \cdots, \mathbb{L}_T, R_{L_1}, \cdots, R_{L_T}\big) \geq \alpha,$$

  for some parameter α, and H being a suitable notion of entropy (either information-theoretic or computational). The random variables $\mathbb{L}_i$ represents the leakage from the i^{th} secret key sk_i, i.e., $\mathbb{L}_i = L_i(sk_i)$, and R_{L_i} denotes the random coins of the adversary. Next, we look at conditioning on a subset of these queries such as the set of non-local leakage queries, as defined below.

- $\mathcal{A}_N^{\alpha}$: The class *α-entropic non-local* adversaries is the class of PPT adversaries $\mathcal{B}$ such that the non-local leakage queries made by $\mathcal{B}$ satisfy the following definition:

$$\forall j \in [T], \mathrm{H}\big(SK_j \,|\, \mathbb{L}_{a_1}, \cdots, \mathbb{L}_{a_i}, R_{L_{a_1}}, \cdots, R_{L_{a_i}}\big) \geq \alpha \tag{1}$$

  where $(R_{L_{a_1}}, \cdots, R_{L_{a_i}})$ are random coins of the adversary corresponding to the subset $(L_{a_1}, \ldots, L_{a_i}) \subseteq (L_1, \ldots, L_T)$ of non-local leakage queries. Note that this *does not* restrict the local leakage queries in any way.
- $\mathcal{A}^{\lambda}$: The class of *λ-length adversaries* is the set of PPT adversaries $\mathcal{B}$ making leakage queries such that $\forall j \in [T]$, $|L_i(sk_i)| < \lambda$.
  We now look at imposing this length bound condition only on a subset of the leakage queries. We get,
- $\mathcal{A}_L^{\lambda}$: The class of *λ-length local adversaries* is the set of all PPT adversaries $\mathcal{B}$ making leakage queries such that, for all local leakage queries L_i we have that, $|L_i(sk_i)| < \lambda$. This does not restrict the output length of the non-local leakage queries.

In the above definitions, the L and N in subscript indicate local and non-local respectively. In addition, the presence of a 1 or 0 in the superscript indicates the number of queries the adversary can make. By default, the adversary can make a polynomial number of queries. From the above notations, it is it is easy to recover the class of adversaries for the Continual Leakage Model, Bounded Leakage Model and Entropy Bounded Leakage Model as follows.

- *Continual Leakage (CL) Model*: The adversary is not constrained by the number or the type of queries. The only constraint is on the length of the output of the leakage functions in each time period. Let this be denoted by $\lambda < |sk|$. Therefore, $\mathcal{A}_{\mathsf{CL}} = \mathcal{A}^{\lambda}$. Note that, in this context the right notion of entropy that must be considered in the definition of local leakage (see Def. 9) is *average conditional min-entropy*, since the update function is randomized.
- *Bounded Leakage (BL) Model*: Let us recall that the BL model is envisioned for the setting where there is a single secret key and there is no key update procedure. Cast in our setting, this can be thought of as a scheme over multiple time periods where the key evolves to itself, i.e the update is the *identity* function. Note that, the only constraint is that the length of the leakage cannot exceed λ. Therefore, $\mathcal{A}_{\mathsf{BL}} = \mathcal{A}_L^{\lambda,1} \cap \mathcal{A}_N^0$.
- *Entropic Bounded Leakage (EBL) Model*: This is a generalization of the BL model where the constraint is on residual entropy of the secret key conditioned on the leakage. Thus, $\mathcal{A}_{\mathsf{EBL}} = \mathcal{A}_L^{\alpha,1} \cap \mathcal{A}_N^{\alpha,0}$, and the right notion of entropy to be considered in Def. 9 is *average conditional min-entropy.*

The Entropic Continual Leakage Model. This brings us to the definition of the Entropic Continual Leakage (ECL) model.

An initial attempt. Recall that in the CL model the update function is randomized. In contrast, in the ECL model, we want to consider *deterministic* update functions.

As a first attempt, suppose we impose the following restriction: there are a maximum of T time periods and the keys need to have residual entropy of α conditioned on the leakage information, at each time period. In other words, this model considers the class of adversary $\mathcal{A}_{\mathsf{ECL}'} = \mathcal{A}^{\alpha}$.

However, it is not hard to see that this model actually reduces to the *one-time* entropy-bounded leakage (EBL) model (without any key updates). To see this, consider a *trivial* scheme where the update function is simply the identity function. This, along with the requirement that each of the keys be entropic implies that any key-evolving scheme that is secure against *one-time* entropic leakage (i.e, in the EBL model) is also secure in this model. We thus seek something stronger.

Defining the ECL model. Our approach is as follows:

1. We allow the adversary to make *local* leakage queries, provided it leaks only a maximum of λ bits per time period. This rules out the existence of the trivial scheme (where the update function is an identity function), because there exists an adversary which leaks λ physical bits from the keys in each time period and thereby recovering the entire key sk in $|sk|/\lambda$ time periods.
2. In addition, we allow the adversary to make *non-local* leakage queries, provided that the keys have minimum residual entropy of α conditioned on the leakage information at each time period. Since the update function is deterministic, the notion of entropy we consider is computational, namely *average conditional HILL entropy.*
   Thus, from the above we get that, $\mathcal{A}_{\mathsf{ECL}} = \mathcal{A}_L^{\lambda} \cap \mathcal{A}_N^{\alpha}$. The notion of entropy considered while defining the classes $\mathcal{A}_L^{\lambda}$ and $\mathcal{A}_N^{\alpha}$ is $\mathrm{H}_{\epsilon,s}^{\mathsf{HILL}}$. Here $\mathcal{A}_{\mathsf{ECL}}$ denote the class of adversary for the ECL model. When a scheme is secure against $\mathcal{A}_{\mathsf{ECL}}$, we call that scheme to be (α, λ)-ECL-secure.

The next question is the relation between λ and α. By our ECL model, the keys at each time periods are guaranteed to have (computational) entropy at least α, even conditioned on the leakage. Let us fix one such key. At some point, a local leakage can reduce its entropy further by at most λ bits. In other words, the secret key would have a residual entropy of at least $(\alpha - \lambda)$ and we need this to satisfy the relation: $\alpha - \lambda = \omega(\log(\kappa))$. In other words, a scheme which is $(\alpha_{\mathsf{ECL}}, \lambda_{\mathsf{ECL}})$-$\mathsf{ECL}$-secure is also $\alpha_{\mathsf{ECL}'}$-ECL'-secure where $\alpha_{\mathsf{ECL}'} = \alpha_{\mathsf{ECL}} - \lambda_{\mathsf{ECL}}$. This shows us that $\alpha_{\mathsf{ECL}'} < \alpha_{\mathsf{ECL}}$.

3.1 Forward Security under Entropic Continual Memory Leakage (FS + ECL) Model

One of the key advantages of considering continuous leakage-resilient cryptographic primitives with deterministic update function is that, they can be seamlessly amalgamated with the notion of forward security (FS). To this end, we provide a unified model that captures both these aspects simultaneously– forward security and resilience to entropic continual leakage. We call this model as *forward-security under entropic continual leakage* ("FS+ECL"), motivated by the FS+CL model of [7]. In simpler terms, we also require FS to hold under the ECL model.

The key difference from the weaker ECL model is that an adversary also has access to the oracle Exp which allows it to break into the system at some point in time (say time period $t^* \in [T]$) and obtain the secret key sk_{t^*} in full. The access to oracle Lk allows it to obtain leakage from the secret keys of all the time periods prior to the break-in period t^*, subject to the validity of the attacker[4].

The security under this model is guaranteed against the class of "valid/admissible adversaries" $\mathcal{A}_{\mathsf{FS+ECL}}$, which we define next. Recall that, for the ECL model the class of admissible adversaries was defined as $\mathcal{A}_{\mathsf{ECL}} = \mathcal{A}_L^{\lambda} \cap \mathcal{A}_N^{\alpha}$. However, before formally defining the class $\mathcal{A}_{\mathsf{FS+ECL}}$ of adversaries, we will need to modify the definitions of "local leakage" (see Def. 9) and the α-entropic condition of the ECL model (see Eq. 1) for the FS+ECL model as follows:

Definition 10 (Local Leakage - FS+ECL Model). *We will define the random variable representing the secret key at time i as SK_i, then $SK_1, \ldots, SK_T$ are the random variables representing the secret keys at time periods $1, \ldots, T$ respectively defined by a Key Update scheme. Let $t^* \in [T]$ denote the period of break-in. Let L_i be the leakage query issued by the adversary at time i. Let $\mathbb{L}_i$ denote the random variable which represents the leakage $L_i(SK_i)$. Also, let R_{L_i} denote the random coins of the adversary corresponding to the function L_i We can then define the leakage query L_t acting on a key sk_t, at time period t as a local leakage query if any key $SK_j \neq SK_t$ does not lose its entropy due to $\mathbb{L}_t = L_t(sk_t)$. In other words,*

$$\mathrm{H}(SK_j \,|\, \Sigma_t, \mathbb{L}_t = L_t(sk_t), R_{L_t}) = \mathrm{H}(SK_j \,|\, \Sigma_t)$$

where we let Σ_t represent the other queries and the respective responses, i.e, $\Sigma_i = \{\mathbb{L}_j\}_{j=1}^{t^*-1} \setminus \mathbb{L}_t \cup \{R_{L_j}\}_{j=1}^{t^*-1} \setminus R_{L_t}$.

Note that, as remarked before, we only include the leakage queries until time period $t^* - 1$ in this definition[4]. This is a subtle difference from the definition of local leakage in Definition 9.

Definition 11 (λ-length, local adversary - FS+ECL Model). *The class of λ-length, local adversary in the* FS+ECL *model ($\mathcal{A}_{L,t^*}^{\lambda}$) is the class of all PPT adversaries such that: for all local leakage queries L_i where $i \in [t^* - 1]$, $|L_i(sk_i)| < \lambda$. This adversary does not have any constraints on the non-local queries, by definition.*

Note that, this is similar to the definition $\mathcal{A}_L^{\lambda}$ defined in the ECL model.

Definition 12 (α-entropic, non-local adversary - FS+ECL Model). *The class of α-entropic non-local adversaries in the* FS+ECL *model ($\mathcal{A}_{N,t^*}^{\alpha}$) is the class of PPT adversaries such that the leakage functions satisfy the following condition:*

$$\forall j \in [t^* - 1], \mathrm{H}\big(SK_j \,|\, \mathbb{L}_{a_1}, \cdots, \mathbb{L}_{a_i}, R_{L_{a_1}}, \cdots, R_{L_{a_i}}\big) \geq \alpha \tag{2}$$

[4] It is vacuous to look at leakage queries starting from time t^* as the adversary has the secret key sk_{t^*} in full and can compute future keys under the regime of deterministic updates.

where where $(R_{L_{a_1}}, \cdots, R_{L_{a_i}})$ *are random coins of the adversary corresponding to the subset* $(\mathbb{L}_{a_1}, \ldots, \mathbb{L}_{a_i}) \subseteq (\mathbb{L}_1, \ldots, \mathbb{L}_{(t^*-1)})$ *of non-local leakage queries made by the adversary before the time period* t^**. This adversary does not have any constraints on the local queries, by definition.*

Remark 2. Note that, both the definitions above use computational notions of entropy, namely *average conditional HILL entropy.* This is needed, since the update function is deterministic.

Defining the FS+ECL model. Combining Definitions 11 and 12 we define the class of admissible adversaries $\mathcal{A}_{\mathsf{FS+ECL}}$ in the FS+ECL model as:

$$\mathcal{A}_{\mathsf{FS+ECL}} = \mathcal{A}^{\lambda}_{L,t^*} \cap \mathcal{A}^{\alpha}_{N,t^*}$$

Remark 3. Finally, we note that, we model the adversary by classifying types of leakage functions rather than providing different oracles (corresponding to local and non-local queries) because it sidesteps the issues of entropy estimation which is currently not known to be doable in polynomial time.

3.2 Modelling Signature Schemes in the FS+ECL Model.

Any signature scheme in the FS+ECL model must follow a key-evolving paradigm. A key-evolving signature (KES) scheme consists of the following algorithms ($\mathsf{KES.Kg}$, $\mathsf{KES.Upd}$, $\mathsf{KES.Sign}$, $\mathsf{KES.Vfy}$). The key generation algorithm $\mathsf{KES.Kg}$ takes as input the security parameter 1^κ (in unary), and outputs a key pair (vk, sk_1), where sk_1 is the base signing key (corresponding to time period 1). The **deterministic** key update algorithm $\mathsf{KES.Upd}$ takes as input $(1^\kappa; vk, t, sk_t)$ and outputs sk_{t+1}, the secret key for the next time period $t+1 \in [T]$. Here T denotes the total number of time periods supported by the scheme. We stress that the update algorithm is *deterministic*. The signing algorithm $\mathsf{KES.Sign}$ takes as input $(1^\kappa, vk, t, sk_t)$ and a message $m \in \mathcal{M}$ (where $\mathcal{M}$ is the message space) and outputs a signature σ_t for the t^{th} time period. The verification algorithm $\mathsf{KES.Vfy}$ takes as input $(1^\kappa, vk, m, (t, \sigma_t))$ to return a decision in $\{\mathsf{true}, \mathsf{false}\}$ regarding whether σ_t is a valid signature of m relative to vk in time period $t \in [T]$.

Correctness: The correctness requirement for a KES scheme states that: for all $\kappa \in \mathbb{N}$, all $(vk, sk_1,) \leftarrow \mathsf{KES.Kg}(1^\kappa)$, all $i \in [T]$, all $m \in M$, all $sk_2, \cdots, sk_i$ satisfying $sk_j \leftarrow \mathsf{KES.Upd}(1^\kappa, vk, j-1, sk_{j-1})$, $\forall\, 2 \leq j \leq i$, and all $\sigma_i \leftarrow \mathsf{KES.Sign}(1^\kappa, vk, i, sk_i, m)$, it should hold that $\mathsf{KES.Vfy}(1^\kappa, vk, m, (i, \sigma_i)) = \mathsf{true}$.

Forward unforgeability under entropic continual leakage (FUFECL**).** Given a key evolving signature scheme KES, consider the experiment $\mathrm{FUFECL}^{\mathcal{A}_{\mathsf{sig}}}_{\mathsf{KES}}(\kappa, \alpha, \lambda, T)$ as defined in Figure 1 running between a challenger $\mathcal{C}$ and a PPT adversary $\mathcal{A}_{\mathsf{sig}}$, parametrized by the security parameter κ, an entropy parameter α (maximum leakage bound), a length parameter λ, and the total number of time periods T that can be supported by the scheme. To achieve a definition devoid of trivial attacks, we define the validity of an adversary in Definition 13.

Game $\mathrm{FUFECL}_{\mathsf{KES}}^{\mathcal{A}_{\mathsf{sig}}}(\kappa,\alpha,\lambda,T)$	Game $\mathrm{FINDECL}_{\mathsf{KEE}}^{\mathcal{A}_{\mathsf{pke}}}(\kappa,\alpha,\lambda,T)$
$S \leftarrow \emptyset\,;\ t \leftarrow 1\,;\ t^* \leftarrow T+1$	$t \leftarrow 1\,;\ t^* \leftarrow T+1$
$(vk, sk_1) \leftarrow\$\ \mathsf{KES.Kg}(1^\kappa)$	$(pk, sk_1) \leftarrow\$\ \mathsf{KEE.Kg}(1^\kappa)$
$(i,m,\sigma) \leftarrow\$\ \mathcal{A}_{\mathsf{sig}}^{\mathsf{Up,\,Exp,Sign}}(1^\kappa, vk)$	$(i, m_0, m_1, state) \leftarrow\$\ \mathcal{A}_1^{\mathsf{Up,Lk,Exp,Dec}}(1^\kappa, pk)$
$\mathsf{win}_1 \leftarrow (1 \le i < t^*) \wedge ((i,m,\sigma) \notin S)$	$b \leftarrow\$\ \{0,1\}\,;\ (i,c) \leftarrow\$\ \mathsf{KEE.Enc}(1^\kappa, pk, i, m_b)$
$\mathsf{win}_2 \leftarrow \mathsf{KES.Vfy}(1^\kappa, vk, m, (i,\sigma))$	$b' \leftarrow\$\ \mathcal{A}_2^{\mathsf{Up,Exp,Dec}}(1^\kappa, state, (i,c))$
Return $(\mathsf{win}_1 \wedge \mathsf{win}_2)$	If not $(1 \le i < t^*)$ then return false
	If $\lvert m_0\rvert \neq \lvert m_1\rvert$ then return false
	Return $(b' = b)$
$\underline{\mathsf{Up}()}$	$\underline{\mathsf{Up}()}$
If $t < T$ then	If $t < T$ then
$\quad sk_{t+1} := \mathsf{KES.Upd}(1^\kappa, vk, t, sk_t)$	$sk_{t+1} := \mathsf{KEE.Upd}(1^\kappa, pk, t, sk_t)$
$\quad t \leftarrow t+1$	$\quad t \leftarrow t+1$
$\underline{\mathsf{Lk}(L)}$	$\underline{\mathsf{Lk}(L)}$
Return $L(sk_t)$	Return $L(sk_t)$
$\underline{\mathsf{Exp}()}$	$\underline{\mathsf{Exp}()}$
$t^* \leftarrow t\,;$ Return sk_t	$t^* \leftarrow t\,;$ Return sk_t
$\underline{\mathsf{Sign}(t,m)}$	$\underline{\mathsf{Dec}(t,c_t)}$
$(t,\sigma) \leftarrow\$\ \mathsf{KES.Sign}(1^\kappa, vk, t, sk_t, m)$	If $(t, c_t) \neq (i,c)$
$S \leftarrow S \cup \{(t,m,\sigma)\}\,;$ Return (t,σ)	$\quad$ Return $\mathsf{KEE.Dec}(1^\kappa, pk, (t, sk_t), c_t)$

Fig. 1. Games defining forward unforgeability of key-evolving signature scheme KES under entropic continual leakage, and forward indistinguishability of key-evolving encryption scheme KEE under entropic continual leakage.

Definition 13 (Valid Adversary against the FUFECL Game). *We call an adversary $\mathcal{A}_{\mathsf{sig}}$ "*valid*" if the following holds true: (i) it makes at most one query to its* Exp *oracle and this is the last query, (ii) the adversary $\mathcal{A}_{\mathsf{sig}}$ belongs to the class $\mathcal{A}_{\mathsf{FS+ECL}}$ of admissible adversaries (see Section 3.1 for the definition of the class $\mathcal{A}_{\mathsf{FS+ECL}}$.), and (iii) it does not invoke* $\mathsf{KES.Sign}$ *in its leakage functions.*

Definition 14. (Forward unforgeability under entropic continual leakage). *We say that* $\mathsf{KES} = (\mathsf{KES.Kg}, \mathsf{KES.Upd}, \mathsf{KES.Sign}, \mathsf{KES.Vfy})$ *is* (λ,α)-forward unforgeable under entropic continual leakage ((λ,α)-FUFECL) *with respect to the class $\mathcal{A}_{\mathsf{FS+ECL}}$ of adversaries, if the advantage defined as $\mathsf{Adv}^{\mathsf{fufecl}}_{\mathsf{KES},\mathcal{A}_{\mathsf{sig}}}(\kappa) = \Pr\big[\mathrm{FUFECL}_{\mathsf{KES}}^{\mathcal{A}_{\mathsf{sig}}}(\kappa,\alpha,\lambda,T) = 1\big]$ (see Figure 1) is negligible for all "*valid*" PPT adversaries $\mathcal{A}_{\mathsf{sig}}$ (as defined in Definition 13).*

3.3 Modelling Public Key Encryption in the FS+ECL Model.

A key-evolving encryption (KEE) scheme consists of the following algorithms ($\mathsf{KEE.Kg}$, $\mathsf{KEE.Upd}$, $\mathsf{KEE.Enc}$, $\mathsf{KEE.Dec}$). The key generation algorithm $\mathsf{KEE.Kg}$ takes as input the security parameter 1^κ (in unary), and output a public-secret key pair (pk, sk_1), where sk_1

is secret key corresponding to the base time period. The key update algorithm $\mathsf{KEE.Upd}$ takes as input $(1^\kappa, pk, i)$ and secret key sk_i for time period i, and outputs a secret key sk_{i+1} for the next time period. Assume that the total number of time period supported by the scheme is $T \in \mathbb{N}$. The encryption algorithm $\mathsf{KEE.Enc}$ takes as input $(1^\kappa, pk, i)$ and a message m to return c_i, where c_i is the ciphertext encrypting m under pk for time period $i \in [T]$. The decryption algorithm $\mathsf{KEE.Dec}$ takes $(1^\kappa, pk, i, sk_i, c_i)$ as input to return an output in $m \cup \{\perp\}$.

Correctness: The correctness requirement for a KEE scheme states that: for all $\kappa \in \mathbb{N}$, for all $(pk, sk_1) \leftarrow \mathsf{KEE.Kg}(1^\kappa, T)$, all $m \in \mathcal{M}$, all $sk_2, \cdots, sk_i$ satisfying $sk_j \leftarrow \mathsf{KEE.Upd}(1^\kappa, pk, j-1, sk_{j-1})$, $\forall\, 2 \leq j \leq i$, and all $c_i \leftarrow \mathsf{KEE.Enc}(1^\kappa, pk, i, m)$, it should hold that $\mathsf{KEE.Dec}(1^\kappa, pk, i, sk_i, c_i) = 1$.

Forward Indistinguishability under entropic continual leakage (FINDECL)**.** Given a key-evolving encryption scheme KEE, consider the experiment $\text{FINDECL}^{\mathcal{A}}_{\mathsf{KEE}}(\kappa, \alpha, \lambda, T)$ as defined in Figure 1, running between a challenger $\mathcal{C}$ and a PPT adversary $\mathcal{A}_{\mathsf{pke}} = (\mathcal{A}_1, \mathcal{A}_2)$, parametrized by the security parameter κ, the maximum leakage bound λ in a single time-period, an entropy parameter α, and the total number of time periods T that can be supported by the scheme. To achieve a definition devoid of trivial attacks, we define the validity of an adversary in Definition 15.

Definition 15 (Valid Adversary against the FINDECL Game). *We call an adversary $\mathcal{A}_{\mathsf{pke}} = (\mathcal{A}_1, \mathcal{A}_2)$ "*valid*" if the following holds true: (i) it makes at most one query to its* Exp *oracle and this is the last query, (ii) the adversary $\mathcal{A}_{\mathsf{pke}}$ belongs to the class $\mathcal{A}_{\mathsf{FS+ECL}}$ of admissible adversaries (see Section 3.1 for the definition of the class $\mathcal{A}_{\mathsf{FS+ECL}}$.).*

Remark 4. Note that, in our $\text{FINDECL}^{\mathcal{A}_{\mathsf{pke}}}_{\mathsf{KEE}}(\kappa, \alpha, \lambda, T)$ game, the adversary $\mathcal{A}_2$ does *not* get access to the leakage oracle. This is related to *after-the-fact* leakage, where the adversary, after receiving the challenge ciphertext c, can use the leakage oracle to specifically leak the challenge bit b hidden by c. To address this impossibility due to after-the-fact leakage, one can consider weaker leakage models, like the split-state leakage model. However, in this case the machinery would become significantly more complex and take away from our core focus.

Definition 16. (Forward indistinguishability under entropic continual leakage). *We say that* $\mathsf{KEE} = (\mathsf{KEE.Kg}, \mathsf{KEE.Upd}, \mathsf{KEE.Enc}, \mathsf{KEE.Dec})$ *is* (λ, α)-forward indistinguishable under entropic continual leakage ((λ, α)-$\mathsf{FINDECL}$) *with respect to the class $\mathcal{A}_{\mathsf{FS+ECL}}$ of adversaries, if the advantage defined as* $\mathsf{Adv}^{\mathsf{findecl}}_{\mathsf{KEE},\mathcal{A}_{\mathsf{pke}}}(\kappa) = \Pr[\text{FINDECL}^{\mathcal{A}_{\mathsf{pke}}}_{\mathsf{KEE}}(\kappa, \alpha, \lambda, T) = 1]$ *(see Figure 1) is negligible for all "*valid*" PPT adversaries $\mathcal{A}_{\mathsf{pke}} = (\mathcal{A}_1, \mathcal{A}_2)$ (as defined above).*

4 Construction of FS+ECL Encryption Scheme.

In this section, we present our construction of key-evolving encryption scheme in the FS+ECL model. To this end, we first introduce and formalize a notion of *forward-secure* *entropic leakage-resilient non-interactive key exchange* (FS-ECLR-NIKE) protocol (see Section 4.1). In Section 4.5, we show how to construct a FS-ECLR-NIKE protocol from indistinguishability obfuscation and one-way functions. Finally, in Section 4.7, we show a *generic transformation* from a FS-ECLR-NIKE protocol to a FINDECL secure encryption scheme. We view our security model and construction of FS-ECLR-NIKE as a result of independent interest, that may find more useful applications beyond the one shown in this work.

4.1 NIKE in FS+ECL 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 and entropic continual leakage model. We provide the definition of forward secure (FS) non-interactive key exchange (NIKE) protocol in the entropic continual leakage model (dubbed as "FS-ECLR-NIKE"). Our security model for FS-ECLR-NIKE can be seen as a leakage-resilient adaptation of the model of forward-secure NIKE (FS-NIKE) of Pointcheval and Sanders ($\mathcal{PS}$ model) [40]. We call our model of NIKE as the $\mathcal{ECL}$-$\mathcal{PS}$ model to emphasize that our model is entropic leakage-resilient version of the original $\mathcal{PS}$ model.

4.2 Syntax.

A FS-ECLR-NIKE scheme $\mathsf{FS\text{-}ECLR\text{-}NIKE}$, consists of the tuple of algorithms ($\mathsf{NIKE.Setup}$, $\mathsf{NIKE.Gen}$, $\mathsf{NIKE.Upd}$, $\mathsf{NIKE.Key}$). We associate to $\mathsf{FS\text{-}ECLR\text{-}NIKE}$ a public key space $\mathcal{PK}$, a secret key space $\mathcal{SK}$, a 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.

- $\mathsf{NIKE.Setup}(1^\kappa, (\alpha, \lambda))$: On input the security parameter κ (expressed in unary), and leakage parameters (α, λ) of the ECL model, the setup algorithm outputs a set of global parameters of the system denoted by *params*. The current time period t is initially set to 1. In case, the system supports a maximum of T time periods, the setup additionally takes T as argument to generate *params*.
- $\mathsf{NIKE.Gen}(1^\kappa, ID_A)$: On input an identity $ID_A \in \mathcal{IDS}$, the key generation outputs a public-secret key pair (pk^A, sk_1^A) for ID_A corresponding to the base time period. We assume that the secret keys implicitly contain the description of the appropriate time periods.
- $\mathsf{NIKE.Upd}(sk_t^A)$: The update algorithm takes as input the secret key sk_t^A of some user ID_A corresponding to time period $t \geq 1$ and outputs the updated secret key sk_{t+1}^A for the next time period $t+1$. They key sk_t^A is then securely erased from memory.

- $\mathsf{NIKE.Key}(ID_A, pk^A, ID_B, sk_t^B)$: 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 $shk_t^{AB} \in \mathcal{SHK}$ 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 secret keys sk_t^A and sk_t^B respectively, it holds that:

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

4.3 On the leakage-resilience of NIKE protocols.

A closer inspection into the model of NIKE will suggest that achieving leakage-resilience for NIKE protocols in its most generic form is *impossible*. To illustrate this, let us consider two parties ID_A (with key pair (pk^A, sk_t^A)) and ID_B (with key pair (pk^B, sk_t^B)) who wants to establish a shared key for some time period, say t. Since the leakage functions can be arbitrary PPT functions, they can, in particular, be the session-key derivation function also! The adversary $\mathcal{A}$, while leaking from the secret key of ID_B (resp. ID_A), can set the leakage function to be $f = \mathsf{NIKE.Key}(ID_A, pk^A, ID_B, \cdot)$ (resp. $f = \mathsf{NIKE.Key}(ID_B, pk^B, ID_A, \cdot)$). This allows the adversary to leak directly from the shared key shk^{AB} established between ID_A and ID_B. Hence, with very high probability shk_{AB} will no longer be indistinguishable from a random session key.

Admissible Adversaries. The above observation suggests that, if the leakage function (while leaking from the secret key of any one of the parties involved in the shared key derivation in the same time period) is allowed to simultaneously access the keys of both the parties (involved in the shared key derivation), then constructing a leakage-resilient NIKE protocol is *impossible*. Hence, we need to enforce some meaningful restrictions on the class of leakage functions. We bypass the above impossibility result by considering security of NIKE protocols with respect to an *admissible* class of adversary $\mathcal{A}_{\mathsf{NIKE}}$. Before defining the class of admissible adversaries for NIKE, we explain the restrictions that we impose on the class of leakage functions that the adversary $\mathcal{A}_{\mathsf{nike}}$ is allowed to query. These are:

- We assume that the leakage functions (queried by $\mathcal{A}_{\mathsf{nike}}$), in any of their invocations, cannot embed as input the keys corresponding to both the (distinct) parties participating in a NIKE protocol in the same time period (say t). In other words, invoking the leakage function f_i as $f_i(pk^A, sk_t^A, params)$ or $f_i(pk^B, sk_t^B, params)$ is permitted; while invoking f_i as $f_i(sk_t^A, pk^B, params)$ or $f_i(sk_t^B, pk^A, params)$ is *not allowed* (since the arguments inside the leakage function f_i involves the keys corresponding to both ID_A and ID_B). That is, the leakage functions f_i are allowed to work only with keys associated with a *single user* at a time. Let us denote the class of adversaries which respect the above constraint to be $\mathcal{A}$.

- The leakage queries asked by the adversary $\mathcal{A}_{\mathsf{nike}}$ must satisfy the FS+ECL requirements, i.e., $\mathcal{A}_{\mathsf{nike}}$ must belong to the class $\mathcal{A}_{\mathsf{FS+ECL}}$ of adversaries (see Section 3.1 for the definition of the class $\mathcal{A}_{\mathsf{FS+ECL}}$).

We define the admissible class $\mathcal{A}_{\mathsf{NIKE}}$ of adversary for the $\mathcal{ECL}$-$\mathcal{PS}$ model as $\mathcal{A}_{\mathsf{NIKE}} = \mathcal{A}_{\mathsf{FS+ECL}} \cap \mathcal{A}$, where $\mathcal{A}_{\mathsf{FS+ECL}}$ and $\mathcal{A}$ are defined as above. In defining the security model for leakage-resilient NIKE protocols, we only consider security against adversaries that belong to this admissible class $\mathcal{A}_{\mathsf{NIKE}}$.

Remark 5. At first glance, it may not be clear why this leakage model for NIKE should be useful. Interestingly, we show that this leakage model for NIKE is already strong enough to construct a FS+ECL PKE scheme (where the leakage functions can be adaptively chosen by the adversary). Indeed, constructing such a PKE scheme was the main objective of our work, and hence this leakage model of NIKE suffices for our purpose!

Remark 6. Lastly, one may note that the above impossibility result (of constructing leakage-resilient NIKE protocols) does not carry forward to the setting of (interactive) key exchange protocols. This is because, the session key established between two parties in a key exchange protocol depends not only on the long term key pairs of both the parties, but also on the ephemeral key pairs (randomnesses) of both of them.

4.4 Security Model for FS+ECL NIKE.

Our security model for NIKE generalizes the model of forward-secure NIKE of [40] (called the $\mathcal{PS}$ model). We refer to our model as the $\mathcal{ECL}$-$\mathcal{PS}$ model. The $\mathcal{ECL}$-$\mathcal{PS}$ model allows the adversary $\mathcal{A}_{\mathsf{nike}} = (\mathcal{A}_1, \mathcal{A}_2)$ to do the following:

- Similar to [23, 40], our $\mathcal{ECL}$-$\mathcal{PS}$ model also allows for both "honest key registration" (HKR) and "dishonest key registration" (DKR) queries (modeled by giving $\mathcal{A}_{\mathsf{nike}}$ access to the RegHon, RegCor oracles respectively). The DKR query allows the adversary to register public keys of his choice in the system. This models real-life situations where minimal trust must be placed on the certification authority (CA). In particular, the CA is not assumed to check whether a registered public key already belongs to the system, and also does not check if the party registering the public key knows the corresponding private key.
- The adversary $\mathcal{A}_{\mathsf{nike}}$ may also ask to reveal shared keys between these "corrupt" users and the honest (non-adversarially controlled) users. This is modeled by giving the adversary access to the oracle Reveal.
- It may also break into the system and obtain the secret key of an honest party at any particular time period t^*. This is modeled by giving the adversary access to the oracle Exp.

Game ECL-PS$^{\mathcal{A}_{\mathsf{nike}}}_{\mathsf{NIKE}}(\kappa, \alpha, \lambda, T)$	$\mathsf{RegHon}(ID)$
$params \leftarrow \mathsf{NIKE.Setup}(1^\kappa, (\alpha, \lambda), T)$; $S, Q \leftarrow \emptyset$	$(pk, sk_1) \leftarrow \mathsf{NIKE.Gen}(1^\kappa, ID)$
$(ID_A, ID_B, \tilde{t}) \leftarrow \mathcal{A}_1^{\mathsf{RegHon},\mathsf{RegCor},\mathsf{Reveal},\mathsf{Lk},\mathsf{Exp}}(params)$	Add $(ID, sk_1, pk, \mathsf{honest})$ to S
$b \leftarrow_{\$} \{0, 1\}$	Return pk
If $b = 0$ then	
$\quad shk_{\tilde{t}}^{AB} \leftarrow \mathsf{NIKE.Key}(ID_A, pk_A, ID_B, sk_{\tilde{t}}^B)$	$\mathsf{RegCor}(ID, pk)$
$\quad$ Return $shk_{\tilde{t}}^{AB}$	Add $(ID, --, pk, \mathsf{corrupt})$ to S
Else Return $shk_{\tilde{t}}^{AB} \leftarrow_{\$} \mathcal{SHK}$	
$b' \leftarrow_{\$} \mathcal{A}_2^{\mathsf{RegHon},\mathsf{RegCor},\mathsf{Reveal},\mathsf{Lk},\mathsf{Exp}}(shk_{\tilde{t}}^{AB})$	
If $(ID_A, -, -, \mathsf{corrupt}) \in S$ or	$\mathsf{Lk}(L, ID, t)$
$\quad (ID_B, -, -, , \mathsf{corrupt}) \in S$, then return $\perp$	Return $L(sk_t)$
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 $\perp$	
If $(ID_A, ID_B, \tilde{t})$ was made, then return $\perp$	$\mathsf{Exp}(ID, t^*)$
Return $(b' = b)$	If $(ID, sk_1, pk, \mathsf{honest}) \in S$
$\mathsf{Reveal}(ID_A, ID_B, t)$	$\quad$ Add (ID, t^*) to Q
If $(ID_A, -, -, \mathsf{corrupt}) \in S$ and	$\quad$ Return sk_{t^*}
$\quad (ID_B, -, -, , \mathsf{corrupt}) \in S$, then return $\perp$	Else Return $\perp$
If $(ID_B, -, -, \mathsf{corrupt}) \notin S$	
$\quad shk_t^{AB} \leftarrow \mathsf{NIKE.Key}(ID_A, pk_A, ID_B, sk_t^B)$	
$\quad$ Return shk_t^{AB}	
Else If $(ID_B, -, -, \mathsf{corrupt}) \in S$	
$\quad shk_t^{AB} \leftarrow \mathsf{NIKE.Key}(ID_A, sk_t^A, ID_B, pk_B)$	
$\quad$ Return shk_t^{AB}	

Fig. 2. Game defining ECL-PS of NIKE scheme NIKE under entropic continual leakage.

- The adversary $\mathcal{A}_{\mathsf{nike}}$ may also query the oracle Lk to *leak* from the secret keys of honest parties, subject to constraint that $\mathcal{A}_{\mathsf{nike}}$ belongs to the class $\mathcal{A}_{\mathsf{NIKE}}$ of admissible adversaries (defined above).
- Finally, in the challenge phase, the adversary $\mathcal{A}_{\mathsf{nike}}$ has to distinguish the shared key established between two honest users from a random key for any time period prior to the period of exposure (break-in) t^*. Note that, in the ECL-PS game the adversary is also allowed to leak from the secret keys of the challenge parties for all the time periods prior to t^*.

The formal details of our ECL-PS game is given in Figure 2.

Definition 17 (Forward Secure Entropic Continual Leakage Resistant NIKE). *A* NIKE *protocol is* (λ, α)-entropic forward-secure *and resilient to* entropic continual leakage $((\lambda, \alpha)$-FS-ECLR) *with respect to the class* $\mathcal{A}_{\mathsf{nike}}$ *of adversaries, if the advantage defined as* $\mathsf{Adv}^{\mathsf{fs\text{-}ecl}}_{\mathcal{A}_{\mathsf{ECL}}}(\kappa) = |\mathsf{Pr}[\text{ECL-PS}^{\mathcal{A}}_{\mathsf{ECL}}(\kappa, \alpha, \lambda, T) = 1] - 1/2|$ *(see Figure 2) is negligible in* κ *for all "valid" PPT adversaries* $\mathcal{A}_{\mathsf{nike}}$ *(as defined above).*

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 users needs to be honestly registered (ii) the adversary $\mathcal{A}_{\mathsf{nike}}$ is not allowed to corrupt any of the targeted users prior to the challenge time period $\tilde{t}$, (iii) $\mathcal{A}_{\mathsf{nike}}$ is not allowed to obtain the shared key between the targeted users at the time period $\tilde{t}$, (iv) $\mathcal{A}_{\mathsf{nike}}$ is allowed to leak on the secret keys of both the targeted users, as long as it satisfies all the restrictions given above (namely it belongs to the class $\mathcal{A}_{\mathsf{NIKE}}$ of admissible adversaries). We emphasize that the adversary is allowed to corrupt any user other than the targeted users before the time period $\tilde{t}$, and it can even corrupt the targeted users after $\tilde{t}$.

4.5 Construction of FS + ECL NIKE scheme

In this section, we present our construction of forward-secure NIKE protocol resilient to entropic continual leakage. Our construction of FS-ECLR NIKE can only support an a-priori bounded (yet any polynomial) number of time periods, say T. Let $\mathsf{PRF} : \mathcal{K} \times \{0,1\}^* \to \mathcal{SHK}$ be a puncturable PRF, $i\mathcal{O}$ be an indistinguishability obfuscator for circuits, and $\mathsf{PRG}_2 : \{0,1\}^\kappa \to \{0,1\}^{2\kappa}$ be a length-doubling PRG. We build our FS-ECLR-NIKE as below.

- $\mathsf{NIKE.Setup}(1^\kappa, T)$: Choose a random key K to obtain an instance of a pseudorandom function PRF. Set $params = (\mathcal{H}, \mathsf{PRG}_2, i\mathcal{O})$.
- $\mathsf{NIKE.Gen}(1^\kappa, params, ID_i)$: To compute the key pair of an user ID_i, sample $sk_1^i \xleftarrow{\$} \{0,1\}^\kappa$. Create the public key as $pk^i = i\mathcal{O}(P_i)$, where the program P_i is defined in Figure 3.
- $\mathsf{NIKE.Upd}(1^\kappa, sk_t^i)$: 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.
- $\mathsf{NIKE.Key}(ID_i, pk^i = i\mathcal{O}(P_i), ID_j, sk_t^j)$: The user ID_j runs the program $i\mathcal{O}(P_i)$ on inputs the secret key sk_t^j and the 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 non-local 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 ℓ-entropic leakage-resilient one-way function (ℓ-ELR-OWF).

Constants: sk_1^i, K, T.

Inputs: sk_t^j, t.

1. Check if $t \leq T$. If not, output $\perp$.
2. Update $sk_t^i = \mathsf{NIKE.Upd}^{t-1}(sk_1^i)$.
3. Compute: $X_t^i = \mathsf{PRG}_2(sk_t^i)$ and $X_t^j = \mathsf{PRG}_2(sk_t^j)$.

Output the shared key $shk_t^{ij} = \mathsf{pPRF}(K, (X_t^i, X_t^j))$.

Fig. 3. The program P_i. This program is appropriately padded to the maximum of the size of itself and the programs P_i^*, P_i^{**} and P_i^{final} defined in Figures 4, 5 and 6 respectively. . The programs P_i^*, P_i^{**} and P_i^{final} are only used in the security proof.

It is not hard to see that both the parties ID_i and ID_j end up with the *same* shared key. The correctness of the computation of the shared key shk_t^{ij} by both the parties is shown below:

$$\mathsf{NIKE.Key}(ID_j, pk^j = i\mathcal{O}(P_j), ID_i, sk_t^i) = \mathsf{pPRF}(K, (X_t^i, X_t^j, \mathcal{T}))$$

$$= \mathsf{pPRF}\big(K, \big(\mathsf{PRG}_2(sk_t^i), \mathsf{PRG}_2(\mathsf{NIKE.Upd}^{t-1}(sk_1^j)), \mathsf{PRG}_2(t)\big)\big)$$
$$= \mathsf{pPRF}\big(K, \big(\mathsf{PRG}_2(\mathsf{NIKE.Upd}^{t-1}(sk_1^i)), \mathsf{PRG}_2(\mathsf{NIKE.Upd}^{t-1}(sk_1^j)), \mathsf{PRG}_2(t)\big)\big).$$

$$\mathsf{NIKE.Key}(ID_i, pk^i = i\mathcal{O}(P_i), ID_j, sk_t^j) = \mathsf{pPRF}(K, (X_t^i, X_t^j, \mathcal{T}))$$

$$= \mathsf{pPRF}\big(K, \big(\mathsf{PRG}_2(\mathsf{NIKE.Upd}^{t-1}(sk_1^i)), \mathsf{PRG}_2(sk_t^j), \mathsf{PRG}_2(t)\big)\big)$$
$$= \mathsf{pPRF}\big(K, \big(\mathsf{PRG}_2(\mathsf{NIKE.Upd}^{t-1}(sk_1^i)), \mathsf{PRG}_2(\mathsf{NIKE.Upd}^{t-1}(sk_1^j)), \mathsf{PRG}_2(t)\big)\big).$$

Remark 7. One nice feature of our construction of FS+ECL NIKE is that, it does not require any trusted setup assumptions, like a common reference string (CRS). This is in contrast to the Boneh-Zhandry NIKE protocol [8] which required a CRS. Besides, the public keys of each user in our construction are succinct, in the sense that it scales only logarithmically in the number of time periods T supported by the scheme (since the obfuscated programs only need to remember the counter till T, which takes only up to $\log T$ bits of information).

4.6 Security Proof of our FS-ECLR-NIKE construction.

Theorem 4. *Let κ be the security parameter. Let,* PRG_2 *be a secure length-doubling pseudo-random generator,* pPRF *be a secure punctured PRF, the F_K be a family of secure ℓ-ELR-OWF, and $i\mathcal{O}$ be a secure indistinguishability obfuscator for circuits, then the construction shown in section 4.5 is a (α, λ)-forward-secure entropic leakage-resilient NIKE in the* ECL-PS *model, where $\alpha = \big((1 - \frac{1}{\log \kappa}) \cdot \kappa\big) \in (1 - o(1))\kappa$ and $\lambda = \alpha - \ell$ denotes the leakage per time period.*

Proof sketch. The proof of our NIKE protocol uses the punctured programming paradigm of Sahai and Waters [41]. In the proof, the values X_t^i and X_t^j are chosen uniformly at random, instead of being computed by a length doubling PRG PRG_2. Note that, at this point, we cannot directly invoke the security of PRG_2 to argue that these values are indistinguishable from the output of the PRG. This is because, the inputs to the PRG are not random, and are *only* entropic. However, as we show, the output of the PRG still has sufficient HILL entropy (actually the outputs have Metric^* entropy, which can be converted to HILL entropy with some loss in quality and quantity), and in particular has much larger HILL entropy than the entropy of its input. Now, it can be shown that, with overwhelming probability these values will not have a pre-image under PRG_2. At this point, we change the functionality of our program as follows: If there exists an input that maps to any of these values, we abort. The indistinguishability of the obfuscator ensures that this change in the functionality of the program is not detected by a PPT adversary. Finally, we can puncture the PRF pPRF at the point (X_t^i, X_t^j), and include only the punctured key inside the obfuscated program (i.e., the public key). The indistinguishability of the obfuscator again ensures that this modification is undetectable to an adversary, and we can simulate the view of the adversary using the punctured key. The security of pPRF finally allows us to switch the real shared key to a random key. In our case, the adversary also gets leakage from each of the (updated) NIKE secret keys. However, the entropic continual leakage model ensures that each of the NIKE keys retain some min-entropy, even given all the leakage information. Hence, as discussed above, we can replace the shared key with a random key. We now give the detailed proof of Theorem 4.

Proof. We prove the security of Theorem 4 via a sequence of hybrid experiments. Let S_i be the probability that $\mathcal{A}_{\mathsf{ECL}}$ wins in Game i.

Hybrid$_0$: This corresponds to the original security game in the experiment $\mathsf{Exp}^{\mathsf{fs\text{-}ecl}}_{\mathcal{A}_{\mathsf{ECL}}}(\kappa)$. The challenger first chooses a time period $\tilde{t}$ as a guess for the break-in period of the adversary $\mathcal{A}_{\mathsf{ECL}}$. If $\mathcal{A}_{\mathsf{ECL}}$ chooses any period other than $\tilde{t}$, the challenger aborts. It then chooses two random seeds sk_1^i and sk_1^j and computes $X_t^i = \mathsf{PRG}_2(sk_t^i)$, $X_t^j = \mathsf{PRG}_2(sk_t^j)$. It also creates the obfuscated programs $pk_i = i\mathcal{O}(P_i)$ and $pk_j = i\mathcal{O}(P_j)$ respectively, and gives the challenge key as $shk_t^{ij} = \mathsf{pPRF}(K, (X_t^i, X_t^j))$ to $\mathcal{A}_{\mathsf{ECL}}$. All the leakage information that $\mathcal{A}_{\mathsf{ECL}}$ asks can also be answered by the challenger with the knowledge of the secret keys sk_1^i and sk_1^j. According to the definition of $\mathcal{A}_{\mathsf{ECL}}$, we have:

$$\mathsf{Adv}^{\mathsf{fs\text{-}ecl}}_{\mathcal{A}_{\mathsf{ECL}}}(\kappa) = \mathsf{Pr}[S_0].$$

Hybrid$_1$: This is similar to Hybrid$_0$, except that the challenger chooses the values X_t^i, and X_t^j uniformly at random from $\{0,1\}^{2\kappa}$. The leakage from all the secret keys till time period $\tilde{t}$ are still simulated using the original secret keys $sk_1^k, \cdots, sk_{\tilde{t}}^k$, where $k \in \{i, j\}$.

Claim. $|\mathsf{Pr}[S_1] - \mathsf{Pr}[S_0]| \leq \mathsf{negl}(\kappa)$.

Proof. Note that, the above proof does not follow in a straightforward way from the security of PRG_2, since the inputs to PRG_2 are not uniformly random. Instead, each of the secret keys are $(\alpha - \lambda)$-entropic. This follows from our adversarial modeling because we require each key to have at least α entropy conditioned on the non-local leakage and the local leakage (of output length λ) can at most reduce the entropy by λ. We now estimate the computational entropy left in the output of PRG_2, given the leakage. Let $\mathcal{F}$ be a class of functions that leaks at most $(\alpha - \lambda)$ bits from any secret key. Hence, for any function $f \in \mathcal{F}$, the chain rule for Metric^* entropy gives us the following:

$$\begin{aligned} \mathrm{H}^{\mathsf{Metric}^*}_{\epsilon \cdot 2^{(\kappa-\alpha)}, s'}\big(\mathsf{PRG}_2(sk^i_t) | f(sk^i_t) \in \mathcal{F}\big) &\geq \mathrm{H}^{\mathsf{Metric}^*}_{\epsilon, s}(\mathsf{PRG}_2(sk^i_t)) - (\alpha - \lambda) \\ &\geq 2\kappa - (\alpha - \lambda) \\ &\geq 2\kappa - \alpha + \lambda \\ &\geq 2\kappa - \alpha \end{aligned}$$

The second inequality comes from the fact that the output of a PRG has full HILL entropy (and hence Metric^* entropy) when the input is random. Hence, the output of PRG_2 has 2κ bits of Metric^* entropy.

This implies the following: For every $\epsilon_{\mathsf{HILL}} > 0$, $\epsilon' \geq \epsilon \cdot 2^{(2\kappa-\alpha)} + \epsilon_{\mathsf{HILL}}$, we have:

$$H^{\mathsf{HILL}^*}_{\epsilon', s_{\mathsf{HILL}}}\big((\mathsf{PRG}_2(sk^i_t) | f(sk^i_t) \in \mathcal{F})\big) \geq 2\kappa - \alpha,$$

where $s_{\mathsf{HILL}} = \Omega(\frac{\epsilon^2_{\mathsf{HILL}} s}{2\kappa})$.

The above result implies that the output of PRG_2 is indistinguishable from a random variable that has $(2\kappa - \alpha)$ bits of min-entropy. Hence, if we sample a random 2κ bit string X^i_t, the probability that any input of PRG_2 maps to the random string X^i_t is at most $\frac{2^\kappa}{2^{2\kappa-\alpha}} = 2^{\alpha-\kappa}$. Now, if we set, $\alpha = \big((1 - \frac{1}{\log \kappa}) \cdot \kappa\big) \in (1 - o(1))\kappa$, we get that $2^{\alpha-\kappa}$ is negligible. The claim follows. □

Hybrid$_2$: Similar to **Hybrid**$_1$, except that the challenger now changes the obfuscation of the program P_i to an obfuscation of a related program P^*_i, as shown in Figure 4. The program P^*_i additionally checks if there exists an input $sk^j_{t^*}$ for which it holds that $\mathsf{PRG}_2(sk^j_{t^*}) = X^j_t$. If this is the case, the challenger aborts. Similar modification is also done for the other party ID_j.

Claim. $|\mathsf{Pr}[S_2] - \mathsf{Pr}[S_1]| \leq \mathsf{negl}(\kappa)$.

Proof. Note that, at the end of **Hybrid**$_1$, with overwhelming probability, none of the values X^i_t or X^j_t will have a pre-image under PRG_2, since the size of the image of PRG_2 is much larger than its domain. Therefore, with overwhelming probability, there is no input to the program P_i that will cause pPRF to be evaluated at the point (X^i_t, X^j_t). The indistinguishability of the obfuscator ensures that the difference of the advantage of the adversary $\mathcal{A}_{\mathsf{ECL}}$ between Hybrid$_1$ and Hybrid$_2$ is negligible. Hence, the claim follows.

Hybrid$_3$: Similar to **Hybrid**$_2$, except that in this hybrid, we modify the program P^*_i to obtain the program P^{**}_i, as shown in Figure 5. In particular, we puncture the PRF key K at the point (X^i_t, X^j_t), and use the punctured key in the obfuscated program.

Constants: K, T.

Inputs: sk_t^j, t.

Compute the following:

1. Check if $t \leq T$. If not, output $\perp$.
2. Sample $X_t^i, X_t^j \xleftarrow{\$} \{0,1\}^{2\kappa}$.
3. $\boxed{\text{If there exists an input } sk_{t^*}^j\text{, for which } \mathsf{PRG}_2(sk_{t^*}^j) = X_t^j\text{, output } \perp.}$

Output the shared key $shk_t^{ij} = \mathsf{pPRF}(K, (X_t^i, X_t^j))$.

Fig. 4. The program P_i^*.

Claim. $|\mathsf{Pr}[S_3] - \mathsf{Pr}[S_2]| \leq \mathsf{negl}(\kappa)$.

Proof. At the end of Hybrid$_2$, we know that the PRF will never be evaluated on the point (X_t^i, X_t^j). Hence the functionality of both the programs P_i^* and P_i^{**} are identical. The indistinguishability of the obfuscation thereby guarantees that an obfuscation of P_i^{**} is indistinguishable from the obfuscation of the program P_i^*. The claim follows.

Constants: $\boxed{K_{(X_t^i, X_t^j)}}$, T.

Inputs: sk_t^j, t.

Compute the following:

1. Check if $t \leq T$. If not, output $\perp$.
2. Sample $X_t^i, X_t^j \xleftarrow{\$} \{0,1\}^{2\kappa}$.
3. If there exists an input $sk_{t^*}^j$, for which $\mathsf{PRG}_2(sk_{t^*}^j) = X_t^j$, output $\perp$.

Output the shared key $shk_t^{ij} = \mathsf{pPRF}(K, (X_t^i, X_t^j))$.

Fig. 5. The program P_i^{**}.

Hybrid_4 : Similar to **Hybrid**$_2$, except that we modify the program P_i^{**} to obtain the program P_i^{final}, as shown in Figure 6. Here, we sample the shared key shk_t^{ij} uniformly at random from the shared key space.

Claim. $|\mathsf{Pr}[S_4] - \mathsf{Pr}[S_3]| = \mathsf{negl}(\kappa)$.

Proof. We will show that if there exists an adversary $\mathcal{A}_{nike}$ that can distinguish between Hybrid$_3$ and Hybrid$_4$, we can build an adversary $\mathcal{A}_{PRF}$ that can break the security of the punctured PRF. The adversary $\mathcal{A}_{PRF}$ acts as a challenger for $\mathcal{A}_{nike}$. Whenever $\mathcal{A}_{nike}$ makes a register honest user query to $\mathsf{RegHon}(ID_A)$, $\mathcal{A}_{PRF}$ sample sk_1^A uniformly at random, and return the obfuscated program as the public key pk^A to $\mathcal{A}_{nike}$. On a register corrupt user query to $\mathsf{RegCor}(ID_A, pk^A)$ with public key pk_A, $\mathcal{A}_{PRF}$ records

$(ID_A, -, pk^A, \mathsf{corrupt})$. When $\mathcal{A}_{nike}$ queries the oracle $\mathsf{Exp}(ID_A, t)$, the challenger $\mathcal{A}_{PRF}$ computes sk_t^A and returns it to $\mathcal{A}_{nike}$. For a reveal query $\mathsf{Reveal}(ID_A, ID_B, t)$, the challenger $\mathcal{A}_{PRF}$ asks the pPRF oracle for the correct shared key and thus always reveals the correct key. For the challenge query corresponding to the tuple $(ID_i, ID_j, \tilde{t})$, $\mathcal{A}_{PRF}$ asks its challenger for a real or random value, and returns the resulting output to the adversary $\mathcal{A}_{nike}$. The leakage queries of the $\mathcal{A}_{nike}$ are also answered using the knowledge of the appropriate secret keys . Thus, $\mathcal{A}_{PRF}$ perfectly simulates the view of $\mathcal{A}_{nike}$. If the returned value is the actual PRF value, we are in Hybrid_3, else if it a random value we are in Hybrid_4. Hence, the claim follows.

Constants: $K_{(X_t^i, X_t^j)}$, T.

Inputs: sk_t^j, t.

Compute the following:

1. Check if $t \leq T$. If not, output $\perp$.
2. Sample $X_t^i, X_t^j \xleftarrow{\$} \{0,1\}^{2\kappa}$.
3. If there exists an input sk_{t*}^j, for which $\mathsf{PRG}_2(sk_{t*}^j) = X_t^j$, output $\perp$.

$\boxed{\text{Output the shared key } shk_t^{ij} \xleftarrow{\$} \mathcal{SHK}.}$

Fig. 6. The program P_i^{final}.

Finally, note that, the probability $\mathsf{Pr}[S_4] = 0$, since the shared key is sampled randomly. Putting all the above together, the proof of Theorem 4 immediately follows. □

4.7 FINDECL secure PKE from FS-ECLR-NIKE scheme

As a central application of forward-secure entropic leakage-resilient NIKE (FS-ECLR-NIKE), we show how to construct a IND-CCA secure key-evolving key encapsulation mechanism (KEM) resilient to ECL attacks, generically starting from any FS-ECLR-NIKE. From such a FINDCL-secure KEM it is easy to construct a FINDECL-secure PKE scheme using standard techniques. Hence, we focus on the construction of the FINDECL-secure KEM scheme. Our generic transformation essentially adapts the ideas of Freire et al. [24] to deal with forward security and entropic continual leakage.

The main idea of our transformation is as follows: the base public-secret key pair (pk, sk_1) of the FINDECL-secure KEM scheme is sampled using the key generation algorithm of the underlying FS-ECLR-NIKE scheme. The secret keys of the KEM scheme is updated using the key update algorithm of the FS-ECLR-NIKE scheme. To encrypt a message m for time period t, independently sample another base key pair (pk', sk'_1) of the FS-ECLR-NIKE scheme and the secret key sk'_1 is updated (using the key-update algorithm of FS-ECLR-NIKE) to the time period t, resulting in the key sk'_t. The public key pk' is also signed using a one-time signature scheme that binds the public key to its identity. The encapsulation key K is then generated by running the shared key generation algorithm of the FS-ECLR-NIKE with input (pk, sk'_t). The ciphertext consists of the randomly sampled public key pk' and the signature σ. The receiver can compute the same encapsulated key by running the shared key generation algorithm of the FS-ECLR-NIKE with the inputs (pk', sk_t) (where sk_t is the updated version of sk_1 to time period t), assuming the one-time signature verifies.

We now show the detailed construction of our key-evolving KEM scheme $\mathsf{KEM} = (\mathsf{KEM.Kg}, \mathsf{KEM.Upd}, \mathsf{KEM.Enc}, \mathsf{KEM.Dec})$. Before proceeding with the construction we present the syntax and security model for a forward-secure entropic leakage-resilient KEM scheme.

Modelling KEM in the FS+ECL Model. A key-evolving KEM scheme consists of the following algorithms $\mathsf{KEM} = (\mathsf{KEM.Kg}, \mathsf{KEM.Upd}, \mathsf{KEM.Enc}, \mathsf{KEM.Dec})$. The key generation algorithm $\mathsf{KEM.Kg}$ takes as input the security parameter 1^κ (in unary) and optionally the parameters of the scheme (e.g., the maximum number of time periods supported by the scheme), and output a public-secret key pair (pk, sk_1), where sk_1 is secret key corresponding to the base time period. The key update algorithm $\mathsf{KEM.Upd}$ takes as input $(1^\kappa, pk, i)$ and secret key sk_i for time period i, and outputs a secret key sk_{i+1} for the next time period. Assume that the total number of time period supported by the scheme is $T \in \mathbb{N}$. The encapsulation algorithm $\mathsf{KEM.Enc}$ takes as input $(1^\kappa, pk, i)$ to return (c_i, K_i), where K_i is the encapsulated key and c_i is the ciphertext encrypting K_i under pk for time period $i \in [T]$. The decapsulation algorithm $\mathsf{KEM.Dec}$ takes $(1^\kappa, pk, i, sk_i, c_i)$ as input to return an output in $K_i \cup \{\perp\}$. Let us denote the encpasulated key sapce by $\mathcal{K}$.

Correctness: The correctness requirement for KEM states that: for all $\kappa \in \mathbb{N}$, for all $(pk, sk_1) \leftarrow \mathsf{KEM.Kg}(1^\kappa, T)$, all $sk_2, \cdots, sk_i$ satisfying $sk_j \leftarrow \mathsf{KEM.Upd}(1^\kappa, pk, j-1, sk_{j-1})$, $\forall\, 2 \le j \le i$, and all $c_i, K_i \leftarrow \mathsf{KEM.Enc}(1^\kappa, pk, i)$, it holds that $\mathsf{KEM.Dec}(1^\kappa, pk, i, sk_i, c_i) = K_i$.

Forward Indistinguishability under entropic continual leakage (FINDECL)**.** Given a KEM scheme, consider the experiment $\mathrm{FINDECL}^{\mathcal{A}_{\mathsf{kem}}}_{\mathsf{KEM}}(\kappa, \alpha, \lambda, T)$ as defined in Figure 7, running between a challenger $\mathcal{C}$ and a PPT adversary $\mathcal{A}_{\mathsf{kem}} = (\mathcal{A}_1, \mathcal{A}_2)$, parametrized by the security parameter κ, the maximum leakage bound λ in a single time-period, an entropy parameter α, and the total number of time periods T that can be supported by the scheme. Similar to Definition 15, we define what it means for an adversary $\mathcal{A}_{\mathsf{kem}}$ to be "valid".

Definition 18 (Valid Adversary against the FINDECL Game of KEM). *We call an adversary $\mathcal{A}_{\mathsf{kem}} = (\mathcal{A}_1, \mathcal{A}_2)$ "*valid*" if the following holds true: (i) it makes at most one query to its* Exp *oracle and this is the last query, (ii) the adversary $\mathcal{A}_{\mathsf{kem}}$ belongs to the class $\mathcal{A}_{\mathsf{FS+ECL}}$ of admissible adversaries (see Section 3.1 for the definition of the class $\mathcal{A}_{\mathsf{FS+ECL}}$.).*

Definition 19. (Forward indistinguishability under entropic continual leakage). *We say that* $\mathsf{KEM} = (\mathsf{KEM.Kg}, \mathsf{KEM.Upd}, \mathsf{KEM.Enc}, \mathsf{KEM.Dec})$ *is* (λ, α)-forward

Game $\text{FINDECL}_{\mathsf{KEM}}^{\mathcal{A}_{\mathsf{kem}}}(\kappa, \alpha, \lambda, T)$	$\underline{\mathsf{Up}()}$
$t \leftarrow 1$; $t^* \leftarrow T + 1$	If $t < T$ then
$(pk, sk_1) \leftarrow_{\$} \mathsf{KEM.Kg}(1^\kappa)$	$sk_{t+1} := \mathsf{KEM.Upd}(1^\kappa, pk, t, sk_t)$
$(i, state) \leftarrow_{\$} \mathcal{A}_1^{\mathsf{Up},\mathsf{Lk},\mathsf{Exp},\mathsf{Dec}}(1^\kappa, pk)$	$t \leftarrow t + 1$
$b \leftarrow_{\$} \{0, 1\}$; $(i, c, K_0) \leftarrow_{\$} \mathsf{KEM.Enc}(1^\kappa, pk, i)$;	$\underline{\mathsf{Lk}(L)}$
$K_1 \leftarrow_{\$} \mathcal{K}$;	Return $L(sk_t)$
$b' \leftarrow_{\$} \mathcal{A}_2^{\mathsf{Up},\mathsf{Exp},\mathsf{Dec}}(1^\kappa, state, (i, c, K_b))$	$\underline{\mathsf{Exp}()}$
If not $(1 \le i < t^*)$ then return false	$t^* \leftarrow t$; Return sk_t
Return $(b' = b)$	$\underline{\mathsf{Dec}(t, c_t)}$
	If $(t, c_t) \neq (i, c)$
	Return $\mathsf{KEM.Dec}(1^\kappa, pk, (t, sk_t), c_t)$

Fig. 7. Game defining forward indistinguishability of key-evolving KEM scheme KEM under entropic continual leakage.

indistinguishable under entropic continual leakage ((λ, α)-FINDECL) *with respect to the class* $\mathcal{A}_{\mathsf{FS+ECL}}$ *of adversaries, if the advantage defined as* $\mathsf{Adv}^{\mathsf{findecl}}_{\mathsf{KEM},\mathcal{A}_{\mathsf{kem}}}(\kappa) = \Pr[\text{FINDECL}^{\mathcal{A}_{\mathsf{pke}}}_{\mathsf{KEM}}(\kappa, \alpha, \lambda, T) = 1]$ *(see Figure 7) is negligible for all "*valid*" PPT adversaries* $\mathcal{A}_{\mathsf{kem}} = (\mathcal{A}_1, \mathcal{A}_2)$ *(as defined above).*

The Construction. Let FS-ECLR-NIKE = (NIKE.Setup, NIKE.Gen, NIKE.Upd, NIKE.Key) be a *forward-secure entropic leakage-resilient* NIKE secure in the $\mathcal{ECL}$-$\mathcal{PS}$ model, with shared key space $\mathcal{SHK}$, and let OTS = (OTS.Gen, OTS.Sign, OTS.Vfy) be a *strong* existentially unforgeable *one-time* signature scheme. The construction of FINDECL KEM is shown in Figure 8.

4.8 Security proof.

Theorem 5. *Let* FS-ECLR-NIKE *be a* (α, λ)*-entropic continual leakage-resilient forward-secure NIKE* ((α, λ)-FS-ECLR-NIKE) *secure in the* $\mathcal{ECL}$-$\mathcal{PS}$ *model, and* OTS *be a strong existentially unforgeable one-time signature scheme. Then* KEM = (KEM.Kg, KEM.Upd, KEM.Enc, KEM.Dec) *is a* (α, λ)-FINDECL-secure KEM *scheme.*

Proof. Let $\mathcal{A}_{\mathsf{kem}}$ be an adversary against the FINDECL-secure KEM scheme KEM. We now show how to use $\mathcal{A}_{\mathsf{kem}}$ to construct another adversary $\mathcal{A}_{\mathsf{nike}}$ against FS-ECLR-NIKE, thereby contradicting its $\mathcal{ECL}$-$\mathcal{PS}$ security. $\mathcal{A}_{\mathsf{nike}}$ simulates the environment to $\mathcal{A}_{\mathsf{kem}}$ in the following way:

$\mathcal{A}_{\mathsf{nike}}$ on input *params* picks one identity ID_i uniformly at random and runs OTS.Gen to obtain a key pair $(signk, vk)$. It then sets $ID_j = vk$ and makes two queries to its RegHon oracle, namely $\mathsf{RegHon}(ID_i)$ and $\mathsf{RegHon}(ID_j)$ queries to receive the public keys pk^i and pk^j respectively. $\mathcal{A}_{\mathsf{nike}}$ then returns $pk_{\mathsf{KEM}} = (params, ID_i, pk^i)$ to $\mathcal{A}_{\mathsf{kem}}$. When $\mathcal{A}_{\mathsf{kem}}$ makes an update query, $\mathcal{A}_{\mathsf{nike}}$ forwards the query to its own challenger and answers

1. $\mathsf{KEM.Kg}(1^\kappa, (\lambda, \alpha), T)$: Run $\mathsf{NIKE.Setup}(1^\kappa, (\lambda, \alpha), T)$ algorithm to obtain the system parameters $params$. It then chooses $ID \in \mathcal{IDS}$ uniformly at random, and runs $\mathsf{NIKE.Gen}(1^\kappa, ID)$ to obtain a (base) key pair (pk, sk_1). It then sets $pk_{\mathsf{KEM}} = (params, ID, pk)$, and $sk_{\mathsf{KEM}} = (ID, sk_1)$.
2. $\mathsf{KEM.Upd}(1^\kappa, pk, t, sk_t)$: The key update algorithm of KEM runs $\mathsf{NIKE.Upd}(sk_t)$ (where sk_t is the secret key for the current time period t) to obtain the updated key sk_{t+1} for the next time period $t+1$.
3. $\mathsf{KEM.Enc}(1^\kappa, pk_{\mathsf{KEM}}, t)$: This algorithm does the following:
   - Parses pk_{KEM} as $(params, ID, pk)$.
   - Runs $(signk, vk) \leftarrow \mathsf{OTS.Gen}(1^\kappa)$, and repeats this until $vk \neq ID$.
   - Runs $\mathsf{NIKE.Gen}(1^\kappa, vk = ID')$ to obtain another key pair (pk', sk'_1), and computes $\sigma \leftarrow \mathsf{OTS.Sign}(signk, pk')$ to obtain a signature σ on pk'.
   - Runs $sk'_t = \mathsf{NIKE.Upd}^{t-1}(sk'_1)$ to obtain the updated key corresponding to time period t.
   - It then runs $\mathsf{NIKE.Key}(ID, pk, ID', sk'_t)$ to obtain a shared key $K \in \mathcal{SHK}$.

   The output is the tuple $\big(K, C_{\mathsf{KEM}} = (pk', vk, \sigma)\big)$.
4. $\mathsf{KEM.Dec}(1^\kappa, pk_{\mathsf{KEM}}, t, sk_1, C_{\mathsf{KEM}})$: This algorithm does the following:
   - Parse pk_{KEM} as $(params, ID, pk)$ and C_{KEM} as (pk', vk, σ).
   - Run $\mathsf{OTS.Vfy}(vk, pk', \sigma)$. If the signature does not verify, output $\perp$. Also, if $vk = ID$, output $\perp$.
   - Run $sk_t = \mathsf{NIKE.Upd}^{t-1}(sk_1)$ to the time period t.
   - Finally, run $\mathsf{NIKE.Key}(vk = ID', pk', ID, sk_t)$ to obtain a shared key $K' \in \mathcal{SHK} \cup \{\perp\}$. (note that K' can also be $\perp$)

Fig. 8. Construction of FINDECL-secure KEM scheme.

$\mathcal{A}_{\mathsf{kem}}$. On input a leakage query $f(\cdot, t)$ on ID_1 from $\mathcal{A}_{\mathsf{kem}}$, the adversary $\mathcal{A}_{\mathsf{nike}}$ queries it leakage oracle $\mathsf{Lk}(f, ID_i, t)$. It then forwards the answer to $\mathcal{A}_{\mathsf{kem}}$. When the adversary $\mathcal{A}_{\mathsf{kem}}$ enters into the the challenge phase for some time period $\tilde{t}$ (before the break-in period t^*), the adversary $\mathcal{A}_{\mathsf{nike}}$ also enters its own challenge phase and queries the tuple $(ID_i, ID_j, \tilde{t})$ to its challenger to receive a shared key $shk^{ij}_{\tilde{t}}$ for time period $\tilde{t}$. The key $shk^{ij}_{\tilde{t}}$ can either be a real key or a random key. $\mathcal{A}_{\mathsf{nike}}$ then sets the encapsulated key $K^* = shk^{ij}_{\tilde{t}}$, and computes the signature $\sigma^* \leftarrow \mathsf{OTS.Sign}(signk, pk_j)$. Finally, $\mathcal{A}_{\mathsf{nike}}$ sets $C_{\mathsf{KEM}} = (ID_j, pk_j, \sigma^*)$, and returns the challenge tuple (K^*, C_{KEM}) to $\mathcal{A}_{\mathsf{kem}}$. When the adversary $\mathcal{A}_{\mathsf{kem}}$ decides to break-in to some time period say t^*, $\mathcal{A}_{\mathsf{nike}}$ makes a query to its oracle Exp for the same time period t^* and returns the answer to $\mathcal{A}_{\mathsf{kem}}$.

$\mathcal{A}_{\mathsf{kem}}$ also makes Dec queries which can are handled by $\mathcal{A}_{\mathsf{nike}}$ as follows: For each decryption query of the form (t, C_t), parse C_t as (ID', pk', σ'). If $ID' = ID_j$, and $(pk', \sigma') \neq (pk_j, \sigma^*)$, then it is easy to see that we can build another adversary $\mathcal{A}_{\mathsf{OTS}}$ that breaks the strong existential unforgeability of the OTS scheme. If $ID' = ID_i$, it returns $\perp$. If $ID' \notin \{ID_i, ID_j\}$, $\mathcal{A}_{\mathsf{nike}}$ makes a call to its oracle $\mathsf{RegCor}(ID', pk')$ and then makes a call to the oracle $\mathsf{Reveal}(ID_i, ID', t)$ to get a shared key $K \in \mathcal{SHK}$ or $\perp$. It then returns the key K to $\mathcal{A}_{\mathsf{kem}}$.

This completes the description of $\mathcal{A}_{\mathsf{nike}}$'s simulation. Note that, the view of $\mathcal{A}_{\mathsf{kem}}$ is identical when playing either against $\mathcal{A}_{\mathsf{nike}}$ in this simulation or against a real IND-CCA secure FINDECL KEM challenger. When $\mathcal{A}_{\mathsf{kem}}$ outputs a guess b', $\mathcal{A}_{\mathsf{nike}}$ also outputs the same bit b' as guess for the shared key (whether it is real or random). Also, note that, the leakage tolerated by the KEM scheme is exactly the *same* as the amount of leakage tolerated by the underlying FS-ECLR-NIKE NIKE scheme. This concludes the proof. □

5 Signatures Schemes in the FS+ECL Model

In this section, we present our construction of a key-evolving signature scheme in the FS+ECL model. To this end, we first define and construct a new notion of *forward-secure entropic continual leakage-resilient identification* (FS-ECLR-ID) scheme in Sections 5.1 and 5.2 respectively. Our notion of FS-ECLR-ID schemes generalizes the prior definitions of ID schemes, which were either leakage-resilient or forward-secure, but not both. Later, in Section 5.3, we show how to transform such a FS-ECLR-ID scheme into a FUFECL-secure signature scheme using (generalized) Fiat-Shamir (FS) transform. This shows the applicability of FS transform even in the FS+ECL setting. The FUFECL signature scheme obtained via the FS-transform is secure in the RO model and can tolerate a leakage rate of $1/2 - o(1)$. However, one drawback of our construction is that, the resulting signature scheme can support an a-priori bounded (but arbitrary polynomial) number of time periods.

5.1 Forward-secure Entropic Leakage-resilient Identification schemes

An identification scheme is an interactive protocol that enables a client (or prover) to prove its identity to a server (or verifier). In a forward-secure identification scheme, the time is divided into discrete time periods, such that the secret key for period $i + 1$ can be computed from the secret key of period i. The public key remains the same in every time period. More formally, a forward-secure identification scheme consists of the five algorithms $(\mathsf{ParamGen}_{\mathsf{ID}}, \mathsf{Gen}_{\mathsf{ID}}, \mathsf{Update}_{\mathsf{ID}}, \mathcal{P}, \mathcal{V})$ as described below:

1. $\mathsf{ParamGen}_{\mathsf{ID}}(1^\kappa)$: The parameter generation algorithm takes as input the security parameter κ (in unary) and outputs a set of system parameters params, and the maximum number of time periods T supported by the system. The parameters params are taken as implicit input by all the algorithms.
2. $\mathsf{Gen}_{\mathsf{ID}}(1^\kappa, T)$: The key generation algorithm outputs a pair (pk, sk_1) containing the public key pk and a base secret key sk_1 for the first time period.
3. $\mathsf{Update}_{\mathsf{ID}}(pk, sk_t)$: The *deterministic* key update algorithm takes the secret key sk_t of the current time period t and outputs the secret key sk_{t+1} for the next time period $t + 1$, if sk_t is a secret key for time period $t < T$. We assume that the secret keys implicitly contain the information about the time periods they are associated with.
4. $\mathcal{P}(pk, sk_t)$: The prover algorithm takes as input the secret key sk_t for the current time period t, the current conversation transcript and the associated state and outputs the next message (if any) to be sent to the verifier.

5. $\mathcal{V}(pk, t)$: The deterministic verification algorithm takes the public key and the period t and outputs a decision in $\{accept, reject\}$ at the end of the protocol execution. We denote the interaction between $\mathcal{P}$ and $\mathcal{V}$ corresponding to time period t as $\{\mathcal{P}(pk, sk_t) \rightleftharpoons \mathcal{V}(pk, t)\}$.

$\underline{\mathsf{FS\text{-}IDPRE}^{\kappa,\lambda,\alpha,\mathsf{T}}_{\mathsf{ent}}(\mathcal{A}_{\mathsf{id}}), \quad \mathsf{FS\text{-}IDANY}^{\kappa,\lambda,\alpha\mathsf{T}}_{\mathsf{ent}}(\mathcal{A}_{\mathsf{id}})}$

1. **Key Stage.** Let $(\mathsf{params}, T) \leftarrow \mathsf{ParamGen}_{\mathsf{ID}}(1^\kappa)$, $(pk, sk_1) \leftarrow \mathsf{Gen}_{\mathsf{ID}}(1^\kappa)$; give (params, pk) to the adversary $\mathcal{A}_{\mathsf{id}}$.
2. **Learning Stage.** The adversary $\mathcal{A}_{\mathsf{id}}^{\mathcal{O}^{\lambda,\alpha}_{sk_t}, \mathcal{P}(pk, sk_t), \mathcal{O}\mathsf{Upd}(pk, sk_t)}$ gets access to (a) leakage oracle provided that $\mathcal{A}_{\mathsf{id}}$ belongs to the admissible class $\mathcal{A}_{\mathsf{FS+ECL}}$ of adversaries (see Section 3.1 for the definition of the class $\mathcal{A}_{\mathsf{FS+ECL}}$), (b) an honest prover $\mathcal{P}(pk, sk_t)$ modeled as an oracle that runs (arbitrarily many) proofs upon request for each time period $t \in [T]$, and (c) update oracle $\mathcal{O}\mathsf{Upd}(pk, sk_t)$.
3. **Break-in Stage.** The adversary $\mathcal{A}_{\mathsf{id}}$ may provide the description of a time period t^* in which it wants to break into the system. Set $t^* \leftarrow t$, and return sk_t to $\mathcal{A}_{\mathsf{id}}$.
4. **Impersonation Stage.** This stage is defined separately for the two games:

   - The adversary $\mathcal{A}_{\mathsf{id}}$ looses access to all the oracles and runs a protocol $\{\mathcal{A}_{\mathsf{id}} \rightleftharpoons \mathcal{V}(pk, t)\}$ with an honest verifier with respect to some time period $t < t^*$. This notion is called *forward-secure pre-impersonation entropic leakage* security game, and is denoted by $\mathsf{FS\text{-}IDPRE}^{\kappa,\lambda,\alpha,\mathsf{T}}_{\mathsf{ent}}$.
   - The adversary $\mathcal{A}_{\mathsf{id}}^{\mathcal{O}^{\lambda,\alpha}_{sk_t}}$ maintains access to only the leakage oracle $\mathcal{O}^{\lambda,\alpha}_{sk_t}$, but not the prover oracle, and runs a protocol $\{\mathcal{A}_{\mathsf{id}}^{\mathcal{O}^{\lambda,\alpha}_{sk_t}} \rightleftharpoons \mathcal{V}(pk, t)\}$ with an honest verifier with respect to some time period $t < t^*$. This notion is called *forward-secure anytime entropic leakage game, and is denoted by* $\mathsf{FS\text{-}IDANY}^{\kappa,\lambda,\alpha\mathsf{T}}_{\mathsf{ent}}$.

Fig. 9. Attack games for FS-ECLR-ID scheme.

An ID scheme should satisfy the standard *completeness* property, i.e., if the prover is honest the (honest) verifier will accept the transcript generated by the interaction $\{\mathcal{P}(pk, sk_t) \rightleftharpoons \mathcal{V}(pk, t)\}$. We now define the security of forward-secure entropic leakage-resilient ID schemes. Informally, in the *learning stage* the adversary is given access to polynomially many "copies of the prover", and also access to the leakage and update oracles. The adversary can obtain leakage on each of the secret keys corresponding to each time period $i \in [T]$, provided the leakage functions satisfy the constraints of the FS+ECL model. The adversary may also break into the system and obtain the secret key sk_{t^*} for any time period t^*. After the learning stage, the adversary enters into an *impersonation stage* in which it either looses access to all the oracles (called *forward-secure pre-impersonation entropic leakage* security) or retains access to only the leakage oracle (*forward-secure anytime entropic leakage* security). In this stage, the adversary tries to impersonate the prover to the honest verifier with respect to a time period t prior to the break-in period t^*, and wins the game if it succeeds. The attack game is defined in Figure 9. The advantage of an adversary $\mathcal{A}_{\mathsf{id}}$ in the games $\mathsf{FS\text{-}IDPRE}^{\kappa,\lambda,\mathsf{T}}_{\mathsf{ent}}(\mathcal{A}_{\mathsf{id}})$ and $\mathsf{FS\text{-}IDANY}^{\kappa,\lambda,\mathsf{T}}_{\mathsf{ent}}(\mathcal{A}_{\mathsf{id}})$ is the probability that the verifier $\mathcal{V}$ accepts in the impersonation stage.

Definition 20. *Let* $\mathsf{FS\text{-}ECLR\text{-}ID} = (\mathsf{ParamGen}_{\mathsf{ID}}, \mathsf{Gen}_{\mathsf{ID}}, \mathsf{Update}_{\mathsf{ID}}, \mathcal{P}, \mathcal{V})$ *be a forward secure identification scheme parametrized with the security parameter* κ*, leakage parameter* (α, λ)*, number of time periods* T*, and satisfying perfect completeness. We say that the scheme is secure with* forward-secure pre-impersonation entropic leakage-resilient *with respect to the class* $\mathcal{A}_{\mathsf{FS+ECL}}$ *of adversaries, if the advantage of any PPT adversary* $\mathcal{A}_{\mathsf{id}}$ *in the game* $\mathsf{FS\text{-}IDPRE}_{\mathsf{ent}}^{\kappa,\lambda,\alpha,\mathsf{T}}(\mathcal{A}_{\mathsf{id}})$ *is negligible in* κ*. Similarly, we say that the scheme is secure with* forward-secure anytime entropic leakage-resilient *with respect to the class* $\mathcal{A}_{\mathsf{FS+ECL}}$ *of adversaries, if the advantage* $\mathcal{A}_{\mathsf{id}}$ *in the game* $\mathsf{FS\text{-}IDANY}_{\mathsf{ent}}^{\kappa,\lambda,\alpha,\mathsf{T}}(\mathcal{A}_{\mathsf{id}})$ *is negligible in* κ*.*

5.2 Construction of FS-ECLR-ID scheme.

In this section, we present our construction of forward-secure entropic continual leakage-resilient ID scheme.

The Scheme. We show that a forward-secure version of the generalized GQ identification scheme is secure against entropic continual key leakage attacks (see Figure 10). To analyze the scheme, we use the relation $\mathcal{R} = \{(pk, sk) : sk = (\rho, \omega_1, \cdots, \omega_\ell), pk = (g_1, \cdots, g_\ell, h)$, s.t. $h = \prod_{i=1}^{\ell} g_i^{\omega_i} \cdot \rho^e \mod N\}$, where $N = p \cdot q$ (p and q are prime numbers), e and d are chosen such that $e \cdot d = 1 \mod \phi(N)$, and e is a prime number.

Remark 8. Note that, in our construction the sizes of the keys are *linear* in the number of time periods. This is because we store the exponents $e_1, \cdots, e_T$ in the public and the secret key. However, it is possible to have *constant* sized keys by computing the exponents using a *random oracle*, using techniques similar to [2, Sec. 5.1]. We refer the reader to [2] for the details.

Theorem 6. *Assuming the hardness of the* RSA representation problem*, the construction of our identification scheme* $\mathsf{FS\text{-}ECLR\text{-}ID}$ *as shown in* Figure 10 *is* $\big((\ell \log e_i + \log \phi(N) - \lambda), \lambda\big)$-forward-secure pre-impersonation entropic leakage-resilient*, where* $\lambda = \ell \log e - \kappa$*. It is* forward-secure anytime entropic leakage-resilient *tolerating leakage up to* $\lambda' = \frac{1}{2}\lambda$ *bits.*

Proof. To prove the above theorem, we first prove the following lemma:

Lemma 3. *The following three properties hold for our* $\mathsf{FS\text{-}ECLR\text{-}ID}$ *construction:*

1. *It is difficult to find a public key* pk *and two different secret keys* sk_i *and* sk_i' *for* pk *for any time period* $i \in [T]$*. In particular,*

$$\Pr[sk_i' \neq sk_i \text{ and } (pk, sk_i), (pk, sk_i') \in \mathcal{R} \,|\, (pk, sk_i, sk_i') \leftarrow \mathcal{A}(\mathsf{params}); \\ \mathsf{params} \leftarrow \mathsf{ParamGen}_{\mathsf{ID}}(1^\kappa))] \leq \mathsf{negl}(\kappa).$$

2. *The protocol* $\mathcal{P}$, $\mathcal{V}$ *is a* Σ *protocol for* $\mathcal{R}$*.*

$\underline{\mathsf{ParamGen}_{\mathsf{ID}}(1^\kappa, 1^T)}$: Let $N = p{\cdot}q$, where p and q are two distinct ℓ_N-bit primes, and let $e_1, \cdots, e_T$ be distinct ℓ_e-bit primes, co-prime to $\phi(N) = (p-1)(q-1)$, chosen uniformly at random. Here, T denote the number of time periods supported by the system. Also, let $g_1, \cdots, g_\ell \xleftarrow{\$} (\mathbb{Z}_N^*)^\ell$ be generators of a prime-order cylic subgroup of $\mathbb{Z}_N^*$. Set $\mathsf{params} := \big(N, (e_1, \cdots, e_T), (g_1, \cdots, g_\ell), T\big)$. Also, let us denote $f_i = e_{i+1} \cdots e_T$, $f_T = 1$.

$\underline{\mathsf{Gen}_{\mathsf{ID}}(\mathsf{params}, T)}$: This is the initial key generation algorithm. Choose $\rho, \omega_1, \cdots, \omega_\ell \xleftarrow{\$} \mathbb{Z}_N^* \times \mathbb{Z}_{e_1}^\ell$, and set $pk = \big(N, (e_1, \cdots, e_T), (g_1, \cdots, g_\ell), h\big)$, where $h = \prod_{j=1}^\ell g_j^{\omega_j} {\cdot} \rho^E \mod N$, and $E = \prod_{j=1}^T e_j$. Let $\rho_1 = \rho^{E/e_1}$ and $\rho_1' = \rho^{E/f_1}$. The secret key for the base time period is $sk_1 = \big(N, e_1, \rho_1, \rho_1'\big)$.

$\underline{\mathsf{Update}_{\mathsf{ID}}(\mathsf{pk}, \mathsf{sk_i})}$: Parse $pk = \big(N, (e_1, \cdots, e_T), (g_1, \cdots, g_\ell), h\big)$ and $sk_i = \big(N, e_i, \rho_i, \rho_i'\big)$, where $\rho_i = \rho^{E/e_i}$ and $\rho_i' = \rho^{E/f_i}$. Compute, $\rho_{i+1} = \rho_i'^{f_{i+1}}$ and $\rho_{i+1}' = \rho_i'^{e_{i+1}}$. Set $sk_{i+1} = \big(N, e_{i+1}, \rho_{i+1}, \rho_{i+1}'\big)$.

$\underline{\mathcal{P}(pk, sk_i), \mathcal{V}(pk, i)}$: The prover and the verifier run the following protocol corresponding to some time period i. The public key is $pk = (N, e_i, (g_1, \cdots, g_\ell), h)$, and the secret key $sk_i = \big(N, e_i, \rho_i, \rho_i'\big)$. The goal of the prover $\mathcal{P}$ is to prove that h is a e_i-residue. Observe that, $\prod_{j=1}^\ell g_j^{\omega_j} {\cdot} \rho_i^{e_i} \mod N = h$.

$\mathcal{P}$: Randomly chooses $\vec{a_i} = (a_i^{(1)} \cdots, a_i^{(\ell)}) \leftarrow \mathbb{Z}_{e_i}^\ell$, and $s_i \leftarrow \mathbb{Z}_N^*$.Then compute $\mathsf{com}_i := \prod_{j=1}^\ell g_j^{a_i^{(j)}} \cdot s_i^{e_i} \mod N$. Output $(\vec{a}, \mathsf{com}_i)$, and send com_i to $\mathcal{V}$.
$\mathcal{V}$: Chooses a random $\mathsf{ch}_i \leftarrow \mathbb{Z}_{e_i}$ and send ch_i to $\mathcal{P}$.
$\mathcal{P}$: Compute $\vec{z_i} = \big(\mathsf{ch}_i \cdot w_1 + a_i^{(1)}, \cdots, \mathsf{ch}_i \cdot w_\ell + a_i^{(\ell)}\big)$, and $u_i = (s_i \cdot \rho_i^{\mathsf{ch}_i}) \mod N$. Output $\mathsf{resp}_i = \big(\vec{z_i} = (z_i^{(1)}, \cdots, z_i^{(\ell)}), u_i\big)$ and send resp_i to $\mathcal{V}$.

The verifier $\mathcal{V}$ *accepts* if and only if $u_i^{e_i} \cdot \prod_{j=1}^\ell g_j^{z_i^{(j)}} = h^{\mathsf{ch}_i} \cdot \mathsf{com}_i \mod N$.

Fig. 10. Construction of forward-secure entropic continuous leakage-resilient ID scheme FS-ECLR-ID.

3. *Let* $\mathsf{PK}, \mathsf{SK}_i$ *are random variables defined over the key pairs* (pk, sk_i) *for any time period* i*. Then it holds that* $\widetilde{\mathrm{H}}_\infty(\mathsf{SK}_i|\mathsf{PK}) \geq \ell \log \min\{e_1, \cdots, e_T\}$.

Using the properties of Lemma 3, we will complete the proof of Theorem 6.

Proof of Lemma 3. We now prove the three properties stated in Lemma 3.

`Proof of property 1:` The proof of property 1 follows in a straightforward manner from the RSA assumption. The secret key in our construction of the FS-ECLR-ID scheme for a particular time period is actually a RSA ℓ-representation of the public key pk. Property 1 then follows immediately from the hardness of finding two distinct representations for the same public key.

`Proof of property 2:` We now show that FS-ECLR-ID is a Σ protocol. In particular, we need to show that FS-ECLR-ID satisfies completeness, special-soundness and HVZK properties. The completeness is trivial to see.

Special soundness: We will show that given two accepting transcripts $(\mathsf{com}_i, \mathsf{ch}_i, \mathsf{resp}_i)$ and $(\mathsf{com}_i, \mathsf{ch}_i', \mathsf{resp}_i')$ with $\mathsf{ch}_i \neq \mathsf{ch}_i'$ for some time period $i \in [T]$, we can find another representation sk_i' corresponding to pk as follows.

Parse $\mathsf{com}_i = \prod_{j=1}^{\ell} g_j^{a_i^{(j)}} \cdot s_i^{e_i} \mod N$, $\mathsf{resp}_i = \big(\vec{z_i} = (z_i^{(1)}, \cdots, z_i^{(\ell)}), u_i\big)$ and $\mathsf{resp}'_i = \big(\vec{z'_i} = (z_i'^{(1)}, \cdots, z_i'^{(\ell)}), u'_i\big)$, where $z_i^{(j)} = \mathsf{ch}_i \cdot w_i + a_i^{(j)}$, $z_i'^{(j)} = \mathsf{ch}'_i \cdot w_i + a_i^{(j)}$, $u_i = (s_i \cdot \rho_i^{\mathsf{ch}_i}) \mod N$ and $u'_i = (s_i \cdot \rho_i^{\mathsf{ch}'_i}) \mod N$. Also, assume that $\mathsf{ch}_i < \mathsf{ch}'_i$. Then it is possible to extract all the values $(w_1, \cdots, w_\ell)$ by solving the above systems of linear equations, namely by computing $w_j = \frac{(z_i'^{(j)} - z_i^{(j)})}{\Delta\mathsf{ch}_i}$, where $\Delta\mathsf{ch}_i = (\mathsf{ch}'_i - \mathsf{ch}_i)$. Similarly, another e_i-th residue can be extracted by computing $\rho'_i = \big(\frac{u'_i}{u_i}\big)^{1/(\Delta\mathsf{ch}_i)}$. Note that, the inverse exists since $\Delta\mathsf{ch}_i > 0$.

Honest Verifier Zero Knowledge: To prove this, we need to design a simulator Sim who is only given the public key and the challenges corresponding to each time period and has to produce transcripts that are identically distributed to the original transcripts for all the time periods. Without loss of generality, let us show the simulation for any time period $i \in [T]$. The simulator Sim is first given the public key $pk = \big(N, (e_1, \cdots, e_T), (g_1, \cdots, g_\ell), h\big)$, and has to produce a simulated transcript identical to $(\mathsf{com}_i, \mathsf{ch}_i, \mathsf{resp}_i)$. Sim then samples $\vec{z_i} = (z_i^{(1)}, \cdots, z_i^{(\ell)}) \xleftarrow{\$} \mathbb{Z}_{e_i}^{\ell}$ and $u_i \xleftarrow{\$} \mathbb{Z}_N^*$. The simulator then receives the challenge $\mathsf{ch}_i \in \mathbb{Z}_{e_i}$, and computes $\mathsf{com}_i = \frac{u_i^{e_i} \cdot \prod_{j=1}^{\ell} g_j^{z_i^{(j)}}}{h^{\mathsf{ch}_i}} \mod N$. It is trivial to see that the simulated transcript is identically distributed to the original transcript.

`Proof of property 3:` The length of a secret key sk_i is $|sk_i| = \ell \log e_i + \log \phi(N)$, and the public key $pk \in \mathbb{Z}_N^*$. Hence we have:

$$\widetilde{\mathrm{H}}_\infty(\mathsf{sk}_i|\mathsf{pk}) \geq \widetilde{\mathrm{H}}_\infty(\mathsf{sk}_i) - \log \phi(N) = \ell \log e_i \geq \ell \log \min\{e_1, \cdots, e_T\}.$$

The proofs of these properties proves Lemma 3. □

We now continue with the proof of Theorem 6. The proof of this part is similar to the proof of the leakage-resilient ID scheme, as shown in [4]. Suppose there is an adversary $\mathcal{A}_{\mathsf{id}}$ running in time t and having advantage ε in the game $\mathsf{FS\text{-}IDPRE}_{\mathsf{ent}}^{\kappa,\lambda,\mathsf{T}}(\mathcal{A})$. Then, we can construct another adversary $\mathcal{B}$ that runs in time $\approx 2t$ and

$$\begin{aligned}\mathsf{Pr}[sk'_i \neq sk_i \; and \; (pk, sk_i), (pk, sk'_i) \in \mathcal{R} \,|\, (pk, sk_i, sk'_i) \leftarrow \mathcal{B}(\mathsf{params});\\ \mathsf{params} \leftarrow \mathsf{ParamGen}_{\mathsf{ID}}(1^\kappa))] \leq \varepsilon^2 - \frac{1}{\log \phi(N)} - 2^{-\kappa}\end{aligned}$$

The adversary $\mathcal{B}$ chooses a random base key pair $(pk, sk_1) \leftarrow \mathsf{Gen}_{\mathsf{ID}}(\mathsf{params})$ and simulates the environment for the adversary $\mathcal{A}_{\mathsf{id}}$. $\mathcal{B}$ then gives pk to $\mathcal{A}_{\mathsf{id}}$ and uses sk_1 to simulate the leakage queries from the secret keys corresponding to each time period and also the prover oracle $\mathcal{P}(pk, sk_i)$. When $\mathcal{A}_{\mathsf{id}}$ reaches the impersonation stage corresponding to some time period t, $\mathcal{B}$ choose a fresh random challenge $\mathsf{ch}_t \leftarrow \mathbb{Z}_{e_t}$, and receives the transcript $(\mathsf{com}_t, \mathsf{ch}_t, \mathsf{resp}_t)$. Then $\mathcal{B}$ rewinds $\mathcal{A}_{\mathsf{id}}$ and sends a fresh random challenge $\mathsf{ch}'_t \leftarrow \mathbb{Z}_{e_t}$, which results in the transcript $(\mathsf{com}_t, \mathsf{ch}'_t, \mathsf{resp}'_t)$. If both the conversations are accepting and $\mathsf{ch}'_t \neq \mathsf{ch}_t$, then the special soundness property guarantees the existence of an extractor which can find a secret key sk'_t such that $(pk, sk'_t) \in \mathcal{R}$. Let E_1^t be the event that the above happens and E_2^t denote the event that $\mathsf{ch}'_t = \mathsf{ch}_t$.

Claim. $\Pr[E_1^t] \geq \varepsilon^2 - \frac{1}{\log \phi(N)}$

Proof. This follows from a rather straightforward probabilistic argument.

Claim. $\Pr[E_2^t] \leq 2^{-\kappa}$

Proof. Let us think of an experiment $\mathcal{E}_0$ where the adversary $\mathcal{A}_{\mathsf{id}}$ gets access to all the oracles, namely $\mathcal{O}_{sk_t}^{\lambda(\kappa)}$, $\mathcal{P}(pk, sk_t)$ and $\mathcal{O}\mathsf{Upd}(pk, sk_t)$. Let $\mathcal{E}_1$ denote the same experiment as $\mathcal{E}_0$, except that the adversary $\mathcal{A}_{\mathsf{id}}$ is not given access to the prover oracle $\mathcal{P}(pk, sk_t)$, and let $\mathcal{E}_2$ be the experiment in which $\mathcal{A}_{\mathsf{id}}$ has access to only the public key pk. so, we have:

$$\widetilde{\mathrm{H}}_\infty(\mathsf{SK}_t|\mathcal{E}_0) \geq \widetilde{\mathrm{H}}_\infty(\mathsf{SK}_t|\mathcal{E}_1) - \lambda \geq \widetilde{\mathrm{H}}_\infty(\mathsf{SK}_t|\mathcal{E}_2) - \lambda = \widetilde{\mathrm{H}}_\infty(\mathsf{SK}_t|\mathsf{PK}) - \lambda \\ \geq \ell \log e_t - \lambda \geq \kappa.$$

where the first inequality follows from chain rule for min-entropy, the second inequality follows from Lemma 2, and the last inequality follows from property (3) of Lemma 3. The final inequality holds since $\lambda \leq \ell \log \min\{e_1, \cdots, e_T\} - \kappa$.

Note that, the secret keys for each of the time periods retain enough *min-entropy* in them, even given the leakage in that particular time period. Now, by the entropic continual leakage assumption, all the keys are unpredictable, even given the leakage across all the time periods.

Finally, observe that $\Pr[E_2^t] \leq 2^{\widetilde{\mathrm{H}}_\infty(\mathsf{SK}|\mathcal{E}_0)}$

Combining the above two claims, the first part of the theorem follows as:

$$\Pr[E_1^t \wedge \neg E_2^t] \geq \varepsilon^2 - \frac{1}{\log \phi(N)} - 2^{-\kappa} \geq \varepsilon^2 + \mathsf{negl}(\kappa.)$$

The proof for forward-secure anytime entropic leakage security is similar to above, with a subtle difference. In this setting, the leakage functions of the adversary $\mathcal{A}_{\mathsf{id}}$ may also depend on the challenge ch_t. Hence, while rewinding the adversary to fresh random challenge ch'_t the leakage functions may also depend on the new challenge ch'_t. Hence, for anytime leakage security the allowed leakage is half of the original leakage tolerated for pre-impersonation security. □

5.3 FUFECL signatures from FS-ECLR-ID schemes

In this section, we show how to transform any public-coin FS-ECLR-ID scheme into a FUFECL signature scheme using a generalized version of Fiat-Shamir (FS) transform [2]. More precisely, the signature in period i is just the signature obtained via FS transform using the secret key of the i^{th} period $sk_i = \mathsf{Update_{ID}}^{i-1}(pk, sk_1)$ (with period i included in the random oracle input). The amount of leakage tolerated by the signature scheme is exactly the *same* as the leakage tolerated by the underlying identification scheme. We now present the details of the construction.

THE CONSTRUCTION. Let $\mathsf{FS\text{-}ECLR\text{-}ID} = \big(\mathsf{ParamGen_{ID}}, \mathsf{Gen_{ID}}, \mathsf{Update_{ID}}, \mathcal{P}, \mathcal{V}\big)$ be a forward-secure entropic leakage-resilient identification scheme. Let H be a hash function modeled as a random oracle, and let $\mathsf{KES} = (\mathsf{KES.Kg}, \mathsf{KES.Upd}, \mathsf{KES.Sign}, \mathsf{KES.Vfy})$ be the signature scheme obtained via Fiat-Shamir transform applied to $\mathsf{FS\text{-}ECLR\text{-}ID}$ as shown in Fig. 11.

$\mathsf{KES.Kg}(1^\kappa, 1^T)$: Run $\mathsf{params} \leftarrow \mathsf{ParamGen_{ID}}(1^\kappa, 1^T)$; return params.

$\mathsf{KES.Sign}(1^\kappa, vk, sk_i, m)$: Compute: (1) $\mathsf{com}_i \xleftarrow{\$} \mathcal{P}(vk, sk_i)$; (2) $\mathsf{ch}_i \leftarrow \mathsf{H}(\mathsf{com}_i, m, i)$; and (3) $\mathsf{resp}_i \leftarrow \mathcal{P}(vk, sk_i, \mathsf{com}_i, \mathsf{ch}_i)$. Finally, return the signature $\sigma_i \leftarrow (\mathsf{com}_i, \mathsf{resp}_i, i)$

$\mathsf{KES.Upd}(1^\kappa, vk, i, sk_i)$: Run $sk_{i+1} \leftarrow \mathsf{Update_{ID}}(vk, sk_i)$; return sk_{i+1}.

$\mathsf{KES.Vfy}(1^\kappa, vk, m, i, \sigma_i)$: Parse $(\mathsf{com}_i, \mathsf{resp}_i, i) \leftarrow \sigma_i$; compute $\mathsf{ch}_i \leftarrow \mathsf{H}(\mathsf{com}_i, m, i)$, and $d \leftarrow \mathcal{V}(vk, \mathsf{com}_i, \mathsf{ch}_i, \mathsf{resp}_i)$. Return d.

Fig. 11. Generalized Fiat-Shamir transform for forward-secure entropic leakage-resilient signature

Theorem 7. *Let* $\mathsf{FS\text{-}ECLR\text{-}ID} = \big(\mathsf{ParamGen_{ID}}, \mathsf{Gen_{ID}}, \mathsf{Update_{ID}}, \mathcal{P}, \mathcal{V}\big)$ *be a* three-round public-coin (α, λ)-forward-secure anytime entropic leakage-resilient *identification scheme. Then the signature scheme* $\mathsf{KES} = (\mathsf{KES.Kg}, \mathsf{KES.Upd}, \mathsf{KES.Sign}, \mathsf{KES.Vfy})$ *obtained by the generalized Fiat-Shamir transform applied to* $\mathsf{FS\text{-}ECLR\text{-}ID}$ *is* $(\alpha, \frac{\lambda}{2})$-FUFECL-secure.

Proof Sketch. The main idea of the proof follows the original proof of the FS paradigm. In our case, however, we need to guess the time period i of the signature output by the adversary, in order to embed the challenge correctly, hence resulting in a loss of factor T in the security reduction. The leakage queries of the FUFECL adversary can be easily handled by the adversary of the FS-ECLR-ID scheme by querying its own leakage oracle. This concludes the proof.

6 Encryption scheme in the FS+CL model

In the section, we consider the problem of constructing forward-secure encryption scheme in the continual leakage model (dubbed as FS+CL as in [7]). In the FS+CL model, the update process is randomized and the leakage happens according to the CL model (i.e., bounded leakage between two successive invocations). We refer to the reader to [7] for the details of the FS+CL model. To construct such a forward-indistinguishable encryption scheme secure in the CL model (FINDCL-secure PKE), we first introduce a notion of *continual leakage-resilient binary tree encryption* (CLR-BTE) (see below), which can be seen as a restricted version of CLR hierarchical identity-based encryption (CLR-HIBE). The construction of CLR-BTE follows in a straightforward manner from the CLR-HIBE scheme of Lewko et al. [36] (which is based on static assumptions over composite-order bilinear groups). For appropriate choice of parameters, the CLR-HIBE scheme achieves the optimal leakage rate of $1 - o(1)$. Hence, our CLR-BTE scheme also achieves the same leakage rate. Finally, we show how to transform such a CLR-BTE scheme to a FINDCL-secure encryption scheme using the Canetti-Halevi-Katz (CHK) transform. This approach of constructing FINDCL-secure encryption scheme was already suggested in [7]. However, it was intuitively claimed in [7] that the above approach does not work, due to the following reason: "The problem is that FS+CL security of the resulting scheme requires that multiple nodes of the BTE construction can be leaked on jointly, whereas the CL security of HIBE only buys us leakage on each such node individually." Surprisingly, we prove the contrary and show that, indeed, it is possible to simulate the joint leakage by leaking on a single node. This requires a careful analysis of the CHK transform in the FS+CL setting.

6.1 Continual Leakage-resilient Binary Tree Encryption

In this section, we introduce the notion of continual leakage-resilient binary tree encryption (CLR-BTE). Our security model of CLR-BTE generalizes the definition of binary tree encryption (BTE) (as proposed by Canetti et al. [11]) in the setting of continual leakage. A BTE can be seen as a restricted version of HIBE, where the identity tree is represented as a *binary* tree.[5] 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 their 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. Moreover, the secret key of any node can be used to derive the secret keys of its children.

Definition 21. *(Continual leakage-resilient BTE).* A continual leakage-resilient binary tree encryption (CLR-BTE) is a tuple of the PPT algorithms $(\mathsf{Gen}, \mathsf{Der}, \mathsf{Enc}, \mathsf{Dec})$ such that:

1. 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. The key derivation algorithm Der takes as input MPK, the identity of a node $w \in \{0,1\}^{\leq \ell}$, and its secret key SK_w. It returns secret keys SK_{w0}, SK_{w1} for the two children of w.
3. The encryption algorithm Enc takes as input MPK, the identity of a node $w \in \{0,1\}^{\leq \ell}$ and a message M to return a ciphertext C.
4. The decryption algorithm Dec takes as input MPK, the identity of a node $w \in \{0,1\}^{\leq \ell}$, its secret key SK_w, and a ciphertext C. It returns a message M or $\perp$ (to denote decryption failure).

[5] Recall that, in HIBE the tree can have arbitrary degree.

We require the standard correctness requirement, i.e., for all (MPK, SK_ε) output by Gen, any node $w \in \{0,1\}^{\leq \ell}$, and secret key SK_w correctly generated for this node, and any message M, we have $M = \mathsf{Dec}(MPK, w, SK_w, \mathsf{Enc}(MPK, w, M))$.

We now present our security model for CLR-BTE. Our model generalizes the notion of *selection-node chosen-plaintext attacks* (SN-CPA) put forward by Canetti et al. [11] to define the security of BTE. In our model, the adversary first specifies the identity of the target node[6] $w^* \in \{0,1\}^{\leq \ell}$. 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^*[7]. Besides, the adversary is also allowed to leak *continuously* 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 22. A CLR-BTE scheme is secure against *continual 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 $\mathcal{A}$ provides the description of a probabilistic leakage function h with constant output size acting on the set of keys, and an identity of a node w in the path P_{w^*} (that may also include both the root note and the target node w^*). The challenger scans $\mathcal{T}$ to find the tuple with identity w. It should be of the form (w, SK_w, L). The challenger then checks if $L + |h(SK_w)| \leq \lambda(\kappa)$. If this is true, it responds with $h(SK_w)$ and updates the L in the tuple with $L = L + |h(SK_w)|$. If the check fails, it returns $\perp$ to the adversary.

[6] 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 forward-secure CLR encryption scheme.

[7] 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^*.

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^*.[8]

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 essentially follows in a straightforward manner from the continuous leakage-resilient HIBE (CLR-HIBE) construction of Lewko et al. [36], tuned to the setting of a binary tree. The resulting CLR-BTE is *adaptively secure*, since the CLR-HIBE of [36] 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 readers to [36] for the details of the CLR-HIBE construction and its proof. For appropriate choice of parameters, the CLR-HIBE scheme achieves the optimal leakage rate of $1 - o(1)$.

6.2 FINDCL encryption from CLR-BTE scheme

In this section, we show that a generic construction of a FINDCL-secure encryption scheme from any CLR-BTE scheme. The main idea of our construction is very simple: use the Canetti-Halevi-Katz (CHK) transform [11] to the underlying CLR-BTE scheme to construct a FINDCL encryption scheme. In particular, we show the applicability of the CHK transform[9] 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.

Let $(\mathsf{Gen}, \mathsf{Der}, \mathsf{Update}, \mathsf{Enc}, \mathsf{Dec})$ 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 notations: 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 $w^{i+1} = w^i0$ (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.

[8] 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.

[9] The original CHK transform [11] is used to construct a forward-secure PKE scheme starting from a BTE scheme.

1. $\mathsf{KEE.Kg}(1^\kappa, T)$: Run $\mathsf{Gen}(1^\kappa, \ell)$, where $T \le 2^\ell - 1$, 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 organized as a stack of node keys, with the secret key SK_{w^i} on top. We first pop this key off the stack. If w^i is a leaf node, the next node on top of the stack is $SK_{w^{i+1}}$. If w^i is an internal node, compute $(SK_{w^i0}, SK_{w^i1}) \leftarrow \mathsf{Der}(pk, w^i, SK_{w^i})$ and push SK_{w^i1} and then SK_{w^i0} onto the stack. In either case, the node 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 8. *Let $\lambda : \mathbb{N} \to [0,1]$. Let $\Pi = (\mathsf{Gen}, \mathsf{Der}, \mathsf{Update}, \mathsf{Enc}, \mathsf{Dec})$ be a $\lambda(\kappa)$-*CLR-SN-CPA *continual leakage-resilient binary-tree encryption* (CLR-BTE) *scheme. Let $\ell : \mathbb{N} \to \mathbb{N}$ be a polynomial such that $T \le 2^{\ell-1}$. 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 Sketch. 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. In particular, we show that if there exists a $\lambda(\kappa)$-bounded valid adversary $\mathcal{A}_{\mathsf{kee}}$ that breaks the $\lambda(\kappa)$-FINDCL security of Π', we can build another $\lambda(\kappa)$-bounded valid adversary $\mathcal{A}_{\mathsf{clr\text{-}bte}}$ breaking the $\lambda(\kappa)$-CLR-SN-CPA security of Π. The adversary $\mathcal{A}_{\mathsf{clr\text{-}bte}}$ uses $\mathcal{A}_{\mathsf{kee}}$ in a black-box manner. It is very easy for $\mathcal{A}_{\mathsf{clr\text{-}bte}}$ to simulate key generation, update and encryption queries asked by $\mathcal{A}_{\mathsf{kee}}$. The adversary $\mathcal{A}_{\mathsf{clr\text{-}bte}}$ knows the secret keys of all the nodes that are right siblings of the nodes that lie in the path P_{w^*} from the root node to w^* (the target node). Besides, it also knows the secret keys of both the children of w^*[10]. Hence, $\mathcal{A}_{\mathsf{clr\text{-}bte}}$ can itself simulate the update queries asked by $\mathcal{A}_{\mathsf{kee}}$.

However, $\mathcal{A}_{\mathsf{kee}}$ may also ask leakage queries. We partition the nodes of the binary tree into two disjoint sets and simulate the leakage queries on each of these two sets differently. In particular, $\mathcal{A}_{\mathsf{kee}}$ may ask a leakage query on a node w such that w does not lie in the path P_{w^*}, i.e, $w \notin P_{w^*}$, or it may ask leakage query on a node w that lie on the path P_{w^*} (including the root node and w^*), i.e, $w \in P_{w^*}$. Simulation of leakage queries on all the nodes $w \notin P_{w^*}$ is trivial, since $\mathcal{A}_{\mathsf{clr\text{-}bte}}$ already knows the secret keys of all such nodes. Hence, for these nodes it can simulate the leakage queries by itself. However, for all nodes $w \in P_{w^*}$, $\mathcal{A}_{\mathsf{clr\text{-}bte}}$ does not know their secret keys. Let us denote the path from the root to node w as P_w, which is certainly a prefix of the path P_{w^*}. One may think that the leakage on such nodes $w \in P_{w^*}$ can be simulated by $\mathcal{A}_{\mathsf{clr\text{-}bte}}$ by simply querying the leakage oracle of the challenger of the CLR-BTE scheme. However, this is not true, since the challenger of the CLR-BTE scheme expects a function that leaks on each node of the tree *individually*, rather than leaking *jointly* on multiple nodes of the tree. In particular, the secret key sk_w of any node w in the PKE scheme is a

[10] Recall, in the CLR-SN-CPA security game (please see Def. 22) the adversary gets all these secret keys.

tuple of key components, namely $sk_w = (SK_w, \{SK_{rs(P_w)}\})$, rather than a single key component[11], where $\{SK_{rs(P_w)}\}$ denote the secret keys corresponding to all the right siblings of the nodes that lie on the path P_w. However, at this point, the key observation is that the adversary $\mathcal{A}_{\mathsf{clr\text{-}bte}}$ already knows all the secret key components $\{SK_{rs(P_w)}\}$, except the key component SK_w. This is because it knows the secret keys of all the right siblings of the nodes that lie in the path P_{w^*}, and hence also the secret keys of all the right siblings of the nodes that lie in the path P_w. To simulate the leakage f on sk_w, the adversary $\mathcal{A}_{\mathsf{clr\text{-}bte}}$ now modifies the function f into a related leakage function f' that acts only on the secret key component SK_w of sk_w, and at the same time is consistent with the output of f. Thus, the joint leakage on all the nodes is transformed to a leakage on the single node w. The way that we accomplish this is that: when $\mathcal{A}_{\mathsf{clr\text{-}bte}}$ receives the leakage function f from $\mathcal{A}_{\mathsf{kee}}$, it hardwires the secret keys $\{SK_{rs(P_w)}\}$ into the function f.[12] $\mathcal{A}_{\mathsf{clr\text{-}bte}}$ then sends this modified function f' to its own challenger. Hence, with this leakage information and the full knowledge of the other keys of sk_w, $\mathcal{A}_{\mathsf{clr\text{-}bte}}$ can consistently simulate the joint leakage by just leaking on a single node. The formal proof follows.

Proof. 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 $\Pi = (\mathsf{Gen}, \mathsf{Der}, \mathsf{Update}, \mathsf{Enc}, \mathsf{Dec})$. 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 $i^* \in [T]$ and outputs w^{i^*} (the identity of the node corresponding to i^*). $\mathcal{A}_{\mathsf{clr\text{-}bte}}$ then obtains MPK and $\{SK_w\}$ for all the appropriate nodes w[13] 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 $j \leq i^*$, 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 of any node w in the tree.[14] This node can either be of any one of the following types: (1) $w \notin P_{w^{i^*}}$ or (2) $w \in P_{w^{i^*}}$, where $P_{w^{i^*}}$

[11] Recall that, the secret key of any node w in our construction contains the secret key of w, i.e., SK_w, and also the keys corresponding to all right siblings of the nodes on the path P_w.

[12] Note that this is possible to do since $\mathcal{A}_{\mathsf{clr\text{-}bte}}$ has full knowledge of the secret key component $\{SK_{rs(P_w)}\}$ of sk_w.

[13] 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^*}.

[14] Practically, it will only ask for leakage on sk_i for any $i < j$, where j is the period of break-in. This is because, for any $i > j$, the adversary can itself compute the secret key. However, we consider the general case, where all the node are prone to leakage.

is the path containing the nodes from the root node to the target node w^{i^*} (including both).

For the first case, $\mathcal{A}_{\mathsf{clr\text{-}bte}}$ repeatedly runs the algorithm Der with the appropriate secret keys in the set $\{SK_w\}$ to derive the secret key sk_i (corresponding to node w^i). Hence, any leakage query asked on sk_i for $i > i^*$ can be simulated by $\mathcal{A}_{\mathsf{clr\text{-}bte}}$ by simply computing the corresponding secret key and answering the leakage function. For the second case, i.e, when the leakage function is asked on a node $w^i \in P_{w^{i^*}}$ [15], the adversary $\mathcal{A}_{\mathsf{clr\text{-}bte}}$ does not know sk_i or the secret key of any of the ancestors of w^i. The secret key sk_i can be seen 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. Note that, the adversary $\mathcal{A}_{\mathsf{clr\text{-}bte}}$ already knows all of these node keys, since the path P_{w^i} is a prefix of $P_{w^{i^*}}$. Let us denote $sk_i = (SK_{w^i}, \{SK\}_{rs(P_{w^i})})$, where $\{SK\}_{rs(P_{w^i})}$ denote the secret keys of the right siblings of all nodes in path P_{w^i}. The adversary now does the following:

- Receive as input the leakage function f from $\mathcal{A}_{\mathsf{kee}}$. Modify 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 $\mathcal{A}_{\mathsf{kee}}$.

It is clear that $\mathcal{A}_{\mathsf{clr\text{-}bte}}$ perfectly simulates the answers to the leakage queries of the adversary $\mathcal{A}_{\mathsf{kee}}$, regardless of which node w the leakage query is asked.

4. 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 $\mathcal{A}_{\mathsf{kee}}$.
5. 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 $\epsilon(\kappa)/T$. $\square$

7 Conclusion

In this work, we propose the ECL and FS+ECL models as means to construct cryptographic primitives in the continual leakage model with *deterministic* key update procedures. Some of the key open problems left by our work are:

- Construct FS+ECL-secure NIKE and ID schemes that can support *unbounded* number of time periods (currently our constructions can handle only bounded (but an arbitrary polynomial) number of time periods).

[15] Note that, in this case $i < i^*$, since we follow a pre-order traversal.

- Construct efficient FS+ECL-secure NIKE schemes from standard assumptions. Currently, our FS+ECL NIKE scheme relies on $i\mathcal{O}$ and one-way functions. The reliance on $i\mathcal{O}$ does not seem to be inherent; yet there does not seem to be straightforward way to get a construction without using it.
- Finally, one could also propose alternative security models that capture deterministic key updates and at the same time enable secure and efficient constructions of different cryptographic primitives in these models.

References

1. M. Abdalla, J. H. An, M. Bellare, and C. Namprempre. From identification to signatures via the Fiat-Shamir transform: Minimizing assumptions for security and forward-security. In L. R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 418–433. Springer, Heidelberg, Apr. / May 2002.
2. M. Abdalla, F. Benhamouda, and D. Pointcheval. On the tightness of forward-secure signature reductions. *Journal of Cryptology*, pages 1–67, 2018.
3. 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.
4. 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.
5. B. Barak, R. Shaltiel, and A. Wigderson. Computational analogues of entropy. In *In 11th International Conference on Random Structures and Algorithms*, pages 200–215, 2003.
6. 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.
7. M. Bellare, A. O'Neill, and I. Stepanovs. Forward-security under continual leakage. *In 16th International Conference on Cryptology And Network Security*, 2017.
8. D. Boneh and M. Zhandry. Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In J. A. Garay and R. Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 480–499. Springer, Heidelberg, Aug. 2014.
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. 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.
12. J.-S. Coron, C. Gentry, S. Halevi, T. Lepoint, H. K. Maji, E. Miles, M. Raykova, A. Sahai, and M. Tibouchi. Zeroizing without low-level zeroes: New MMAP attacks and their limitations. In R. Gennaro and M. J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 247–266. Springer, Heidelberg, Aug. 2015.
13. J.-S. Coron, M. S. Lee, T. Lepoint, and M. Tibouchi. Cryptanalysis of GGH15 multilinear maps. In M. Robshaw and J. Katz, editors, *CRYPTO 2016, Part II*, volume 9815 of *LNCS*, pages 607–628. Springer, Heidelberg, Aug. 2016.
14. R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In Y. Desmedt, editor, *CRYPTO'94*, volume 839 of *LNCS*, pages 174–187. Springer, Heidelberg, Aug. 1994.
15. D. Dachman-Soled, S. D. Gordon, F.-H. Liu, A. O'Neill, and H.-S. Zhou. Leakage-resilient public-key encryption from obfuscation. In C.-M. Cheng, K.-M. Chung, G. Persiano, and B.-Y. Yang, editors, *PKC 2016, Part II*, volume 9615 of *LNCS*, pages 101–128. Springer, Heidelberg, Mar. 2016.

16. Ö. Dagdelen and D. Venturi. A second look at Fischlin's transformation. In D. Pointcheval and D. Vergnaud, editors, *AFRICACRYPT 14*, volume 8469 of *LNCS*, pages 356–376. Springer, Heidelberg, May 2014.
17. 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.
18. 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.
19. Y. Dodis, A. B. Lewko, B. Waters, and D. Wichs. Storing secrets on continually leaky devices. In R. Ostrovsky, editor, *52nd FOCS*, pages 688–697. IEEE Computer Society Press, Oct. 2011.
20. 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.
21. 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.
22. S. Dziembowski and K. Pietrzak. Leakage-resilient cryptography. In *49th FOCS*, pages 293–302. IEEE Computer Society Press, Oct. 2008.
23. E. S. Freire, D. Hofheinz, E. Kiltz, and K. G. Paterson. Non-interactive key exchange. Cryptology ePrint Archive, Report 2012/732, 2012. http://eprint.iacr.org/2012/732.
24. 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.
25. B. Fuller and L. Reyzin. Computational entropy and information leakage. Cryptology ePrint Archive, Report 2012/466, 2012. http://eprint.iacr.org/2012/466.
26. S. Garg, C. Gentry, S. Halevi, and D. Wichs. On the implausibility of differing-inputs obfuscation and extractable witness encryption with auxiliary input. In J. A. Garay and R. Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 518–535. Springer, Heidelberg, Aug. 2014.
27. L. C. Guillou and J.-J. Quisquater. A "paradoxical" indentity-based signature scheme resulting from zero-knowledge. In S. Goldwasser, editor, *CRYPTO'88*, volume 403 of *LNCS*, pages 216–231. Springer, Heidelberg, Aug. 1990.
28. 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.
29. 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.
30. S. Hohenberger, V. Koppula, and B. Waters. Adaptively secure puncturable pseudorandom functions in the standard model. In T. Iwata and J. H. Cheon, editors, *ASIACRYPT 2015, Part I*, volume 9452 of *LNCS*, pages 79–102. Springer, Heidelberg, Nov. / Dec. 2015.
31. 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.
32. 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.
33. 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.
34. 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.
35. A. B. Lewko, M. Lewko, and B. Waters. How to leak on key updates. In L. Fortnow and S. P. Vadhan, editors, *43rd ACM STOC*, pages 725–734. ACM Press, June 2011.
36. 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.

37. E. Miles, A. Sahai, and M. Zhandry. Annihilation attacks for multilinear maps: Cryptanalysis of indistinguishability obfuscation over GGH13. In M. Robshaw and J. Katz, editors, *CRYPTO 2016, Part II*, volume 9815 of *LNCS*, pages 629–658. Springer, Heidelberg, Aug. 2016.
38. 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.
39. T. Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In E. F. Brickell, editor, *CRYPTO'92*, volume 740 of *LNCS*, pages 31–53. Springer, Heidelberg, Aug. 1993.
40. 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.
41. 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.
42. Y. Yu, F.-X. Standaert, O. Pereira, and M. Yung. Practical leakage-resilient pseudorandom generators. In E. Al-Shaer, A. D. Keromytis, and V. Shmatikov, editors, *ACM CCS 10*, pages 141–151. ACM Press, Oct. 2010.