Tamper Resilient Cryptography Without Self-Destruct

Ivan Damgård, Sebastian Faust, Pratyay Mukherjee, and Daniele Venturi

*Aarhus University**

Abstract

We initiate a general study of schemes resilient to both tampering and leakage attacks. Tampering attacks are powerful cryptanalytic attacks where an adversary can change the secret state and observes the effect of such changes at the output. Our contributions are outlined below:

1. We propose a general construction showing that *any* cryptographic primitive where the secret key can be chosen as a uniformly random string can be made secure against *bounded* tampering and leakage. This holds in a restricted model where the tampering functions must be chosen from a set of bounded size after the public parameters have been sampled. Our result covers pseudorandom functions, and many encryption and signature schemes.
2. We show that standard ID and signature schemes constructed from a large class of Σ-protocols (including the Okamoto scheme, for instance) are secure even if the adversary can *arbitrarily* tamper with the prover's state a *bounded* number of times and/or obtain some bounded amount of leakage. Interestingly, for the Okamoto scheme we can allow also independent tampering with the public parameters.
3. We show a *bounded* tamper and leakage resilient CCA secure public key cryptosystem based on the DDH assumption. We first define a weaker CPA-like security notion that we can instantiate based on DDH, and then we give a general compiler that yields CCA-security with tamper and leakage resilience. This requires a public tamper-proof common reference string.
4. Finally, we explain how to boost bounded tampering and leakage resilience (as in 2. and 3. above) to *continuous* tampering and leakage resilience, in the so-called *floppy model* where each user has a personal floppy (containing leak- and tamper-free information) which can be used to refresh the secret key (note that if the key is not updated, continuous tamper resilience is known to be impossible). For the case of ID schemes, we also show that if the underlying protocol is secure in the bounded retrieval model, then our compiler remains secure, even if the adversary can tamper with the *computation* performed by the device.

In some earlier work, the implementation of the tamper resilient primitive was assumed to be aware of the possibility of tampering, in that it would switch to a special mode and, e.g., self-destruct if tampering was detected. None of our results require this assumption.

*The authors acknowledge support from the Danish National Research Foundation and The National Science Foundation of China (under the grant 61061130540) for the Sino-Danish Center for the Theory of Interactive Computation, and also from the CFEM research center (supported by the Danish Strategic Research Council) within which part of this work was performed. Pratyay Mukherjee was partly supported by a European Research Commission Starting Grant (no. 279447).

Contents

1 Introduction

Side-channel attacks against cryptographic implementations can have devastating consequences for the security of a cryptographic scheme. Such attacks often exploit the physical nature of an implementation by, e.g., measuring the power consumption of a cryptographic device or inducing faults into the computation. In the last years a large and continuously growing body of work has successfully strengthened cryptographic security models to incorporate such powerful attacks, and designed new schemes that can be proven secure within these models. While most such models consider *passive* attacks, where the adversary observes leakage emitting from the cryptographic computation, only few works (some examples are mentioned below) study the *active* counterpart of a leakage attack – the so-called *tamper* or *fault* attack. In contrast to the passive leakage adversary, a tampering adversary can change the secret state or the cryptographic computation and observes the effect of such changes at the output. In practice, tampering attacks can be carried out, e.g., by heating up the device or altering the internal power supply or clock [3, 6], and may have severe consequences for the security of a cryptographic implementation.

An important type of a tampering attack considers an adversary that changes the secret key into some *related* key and observes the effect of such changes at the output [22, 18, 26, 5, 4]. To illustrate such key tampering, consider a digital signature scheme Sign with public/secret key pair (pk, sk). The tampering adversary obtains pk and can replace sk with $T(sk)$ where T is some arbitrary tampering function. Then, the adversary gets access to an oracle $\mathsf{Sign}(T(sk), \cdot)$, i.e., to a signing oracle running with the tampered key $T(sk)$. As usual the adversary wins the game by outputting a valid forgery with respect to the original public key pk. Notice that T may be the identity function, in which case we get the standard security notion of digital signature scheme as a special case.

A common method to protect against key tampering attacks is to integrate into the implementation a mechanism to detect faulty keys. In case of detection the cryptographic scheme is assumed to switch to a "tampered" mode that protects the confidentiality of the original key by, e.g., outputting a special symbol or self-destructing its state. For illustration consider again the example of a digital signature scheme. For many common signature schemes the validity of a secret/public key pair can be efficiently checked. Hence, a possibility to detect tampering uses the current (potentially tampered) secret key and checks its validity against the public key. Of course, this requires that the adversary cannot tamper with the public key.

While tamper detection is an effective countermeasure against key tampering attacks, it unfortunately suffers from two drawbacks. First, it puts additional overhead on the implementation. Notice that the detection procedure needs to be executed each time the scheme is run, and hence such checks may result into significant slow-downs of the cryptographic computation. This matters in particular for lightweight cryptographic schemes as in such settings the detection mechanism will be at least as costly as the cryptographic algorithm itself.[1] Second, and more importantly, standard cryptographic schemes and implementations do not come with built-in tamper detection. This makes the design of cryptographic implementations more cumbersome as the standard cryptographic scheme has to carefully be combined with a tamper detection mechanism.

The above drawbacks raise the question whether there exist *standard* cryptographic schemes

[1] The simplest detection procedure we can think of is checking the current key against its hash value. Notice that releasing the hash of the secret key may violate the security of the underlying cryptographic scheme, and hence does not work in general.

where neither the construction, nor the implementation need to be specially engineered to achieve tamper resilience. In this work we answer this question affirmatively. We show that *any* cryptographic scheme can be made tamper resilient at a relatively low price and *without* relying on tamper detection mechanisms. Our techniques can be used, for instance, to protect the AES block-cipher against key tampering attacks. We also show that certain common cryptographic schemes naturally provide good level of tamper resilience, or can be made so by putting additional (non-standard) assumptions on the security of the underlying cryptographic scheme. We elaborate on our results in more detail in Section 1.2.

We note that the model of key tampering does not include an adversary that tampers with the computation. In other words for most of our results we assume that the circuitry that computes the cryptographic algorithm using the potentially tampered key runs correctly and is not subject to tampering attacks. While this is a clear restriction of our results, security against key tampering is important in its own right:

1. It is the natural first step towards techniques to protect the entire computation against tampering attacks. Clearly, any scheme that achieves security against tampering with the computation must also protect against key tampering.

2. Key tampering attacks can be devastating. Many cryptographic schemes are prone to related key attacks or suffer from "weak keys". It is indeed easy to make (albeit contrived) examples where already a simple change of the key totally breaks the security guarantees.

3. Our model can be motivated by practice in cases where the secret key is stored in one physical device, but is used elsewhere. This could be the case, for instance, when a key is transported over an insecure channel from one secure device to another.

Finally, we notice that the important question of how to protect the circuitry against tampering attacks has been addressed in several recent works [25, 19, 11]. Current techniques consider limited tampering attacks (e.g., setting individual bits of the computation while large parts of the computation remain tamper free) and make strong assumptions on the way in which the computation is carried out (typically the computation must be described as a Boolean circuit). In our work we do not make these assumptions.

Below, we first discuss some previous work and then present an outline of our results and techniques.

1.1 Previous Work

Below we review different approaches that have been proposed to counteract tampering with the secret key.

Tamper resilient encodings. A generic method to protect a cryptographic primitive against tampering with the state has been put forward by Gennaro *et al.* [22]. The authors propose a general "compiler" that transforms any cryptographic device CS with secret state st, e.g., a block cipher, into a "transformed" cryptoscheme CS' running with state st' that is resilient to arbitrary tampering with st'. The compiler essentially signs the secret state st and stores as part of the transformed st' the signature together with the original state. When $\mathsf{CS}'(\mathsf{st}', \cdot)$ is run on some input X, then CS' first checks the validity of the signature and in case of failure self-destructs

and overwrites the secret state. In [22] the authors also show that self-destruct and additionally tamper proof public parameters (in [22] the public key for the signature scheme is tamper proof) are necessary to construct such generic compilers.

While the above works for any tampering function, it is limited to settings where CS does not change its state as it would need access to the secret signing key to authenticate the new state. Dziembowski *et al.* [18] overcome this drawback with an un-keyed solution by introducing the concept of *non-malleable codes*. Inspired by techniques from error detecting codes, they propose an information-theoretic encoding that protects against bit-wise independent tampering. That is, for each bit of the encoded state, the tampering function can decide whether to keep it, to flip it, or to set it to 0/1. A recent work of Liu and Lysyanskaya [29] significantly broadens the class of tampering attacks and shows non-malleable codes in the split-state model. Here, the encoding consists of two parts A and B and the adversary can *independently* apply tampering functions to A and B. The later construction heavily relies on NIZKs and requires a tamper proof CRS. We point out that constructions based on non-malleable codes require tamper detection mechanism to guarantee tamper resistance of the underlying primitive. Moreover, the primitive that uses the non-malleable code must be aware of the coding scheme as the key must be decoded before each usage.

The work of Kalai *et al.* [26]. Most relevant for us is the work of Kalai *et al.* [26] who design new involved schemes for public key encryption and digital signatures with resistance to tampering and leakage attacks. The results of [26] are shown in the so-called *continuous leakage and tampering model (CLT)*, where the adversary can continuously apply attacks to the secret state as opposed to a model where the adversary tampers only a single (or bounded number of) times. As in our work the constructions in [26] do not require to detect the presence of tampering attacks, and circumvent the impossibility result of Gennaro *et al.* [22] by refreshing the key after each usage.[2]

The CLT model generalizes the continual leakage model of [12, 8] as it allows the adversary between each update of the secret key to make a leakage query L and additionally a tampering query T. Here, L and T are both efficiently computable functions, where the range of L is smaller than its domain. When the adversary applies a leakage query it obtains the leakage $L(sk)$ while for a tampering query the current secret key sk is replaced with a tampered secret key $T(sk)$. Informally, a scheme is said to be secure in the CLT model if even after a polynomial number of tampering and leakage queries the adversary cannot break the security with respect to the original public key pk. For a digital signature scheme, for instance, that means that the adversary cannot forge a signature with respect to pk. We emphasize that continual tamper resilience does not trivially follow from continual leakage resilience. While the leakage that is given to the adversary before updating the key must be shorter than the length of the secret key, the adversary may learn potentially a large amount of information from running the cryptoscheme on a tampered key. This is in particular true when the scheme is executed a large number of times before the key gets refreshed (and the adversary can apply its next tampering attack). Clearly, such a large amount of information cannot be simulated in general by a short amount of leakage. Based on ideas from earlier construction in the leakage realm, Kalai *et al.* propose a non-standard construction of a signature scheme and public key encryption with CLT security under the bilinear assumption. The constructions require

[2] Gennaro *et al.* show that continuous tamper resistance is impossible to achieve without tamper detection. Indeed, there is an easy attack that, e.g., breaks any signature scheme with a linear (in the length of the secret key) number of tampering queries.

Tampering Model	ID Schemes		IND-CCA PKE	Any Primitive	
	Σ-Protocols	Okamoto	BHHO	RKA	Transformation
Secret Key (semi-adaptive)	✓	✓	✓	✓	(✓)
Public Parameters	n.a.	✓	n.a.	n.a.	n.a.
Continuous Tampering *i*Floppy	✓	✓	✓	n.a.	n.a.
Key Length	$\log \lvert\mathcal{X}\rvert$	$\ell \log p$	$\ell \log p$	n	n
Tampering Queries	$\lfloor \log \lvert\mathcal{X}\rvert / \log \lvert\mathcal{Y}\rvert \rfloor - 2$	$\ell - 2$	$\ell - 3$	$O(\sqrt[3]{n})$	$O(\sqrt[3]{n})$

Table 1: An overview of our results for bounded leakage and tamper resilience. All parameters $|\mathcal{X}|$, $|\mathcal{Y}|$ ℓ, p and n are a function of the security parameter k. For the case of Σ-protocol, the set $\mathcal{X}$ is the set of all possible witnesses and the set $\mathcal{Y}$ is the set of all possible statements for the language; we can achieve a better bound depending on the conditional average min-entropy of the witness given the statement (cf. Section 4).

a public tamper proof common reference string (CRS).

Key tampering vs. related key security. In *related key security* [5] we require, e.g., from a PRF, that the output is pseudorandom even when running on the faulty key. Clearly, this is a stronger requirement than security against key tampering attacks as the scheme has to achieve its security properties even using the faulty key. However, such strong security requirements necessarily limit the type of tampering attack that can be carried out. For instance, no scheme can be related key secure if the adversary changes the key into a weak or low-entropy key. Indeed, most works on related key security consider rather simple tampering functions that, e.g., can be described by an affine function [33].

Related key security is mostly studied in the context of secret key primitives, while so far we mainly discussed key tampering security for public key primitives, e.g., for a signature scheme. We emphasize that also in the secret key setting our security notion is meaningful. In this setting we guarantee that even after seeing the output of, e.g., a PRF, on a faulty key does not violate the security of the PRF with respect to the *original secret key*. To illustrate, consider, for instance, an adversary that injects faults into a smart-card implementing a block-cipher. Such an adversary may try to learn information about the original key K by interacting with the faulty device, and use this information later to decrypt messages (e.g., ciphertexts collected over the internet) encrypted under the original key K.

1.2 Our Results

In this paper we initiate a general study of schemes resilient to both *bounded* tamper and leakage attacks. We call this model the *bounded leakage and tampering model (BLT)* model. In contrast to [26], the concrete schemes we propose are proven secure under standard assumptions (DL or the black-box security of the underlying block cipher) and are efficient and simple. Moreover, we show that some of our constructions can easily be extended to the continual setting by putting an additional simple assumption on the hardware. We elaborate more on our main contributions in the following paragraphs (see also Table 1 for an overview of our results). Importantly, none of our results require any kind of tamper detection mechanism.

Bounded tamper resilience for any scheme. We show that *any* cryptographic primitive where the secret key can be chosen as a uniformly random string can be made secure in the BLT model by a simple and efficient transformation. Our results therefore cover pseudorandom functions (PRF), block ciphers and many encryption and signature schemes. The result holds in a restricted model of tampering: the adversary first selects an arbitrary set of tampering functions of bounded size. As he interacts with the scheme he must choose every tampering function from the set that was specified initially. We call this the *semi-adaptive* BLT model.

We believe this is a meaningful and interesting result despite the limitation on the adaptivity. First, there seems to be a trade-off between the generality of the result and the tampering model. In fact, we can give (albeit contrived) schemes that become insecure in the adaptive model. On the other hand, when we analyze specific constructions we often get adaptivity for free (see below for more details on this). Second and perhaps more importantly, in practice the physical implementation would typically put strong restrictions on the tampering functions the adversary can use, and thus our model of semi-adaptive tampering may not be very far from reality. Finally, the results extends to the standard (fully adaptive) tampering model assuming the underlying primitive has an additional security property, namely a form of related key attacks (RKA) security that would be meaningful to assume for, e.g., certain block ciphers.

While the proof of our result is quite involved, the basic idea and intuition behind the construction is easy to explain. We use a random string X_0 as secret key, and a universal hash function h as public (and tamper proof) parameter. The construction then computes $K_0 = h(X_0)$ and uses K_0 as secret key for the original primitive. The intuitive reason why one might hope this would work is as follows: each tampering query changes the key, so we get a sequence of keys $X_0, X_1, \ldots, X_t$ for t queries, where each X_i is a function of X_{i-1}. If all X_i have high min-entropy, then we can show that with a suitable choice of h, all the hash values $K_0 = h(X_0), K_1 = h(X_1), \ldots$, are statistically close to uniformly and independently chosen keys. Since the adversary only gets to interact with the K_i's, the independence means that the tampering queries are useless to him. On the other hand, if some X_j has small entropy, it seems we should be able to reveal the value of X_j to the adversary as the X_i's with $i < j$ should still have high entropy and hence hash to independent values. On the other hand the X_i's with $i \geq j$ can be computed from X_j, so those tampering queries can be simulated.

However, things get more complicated for at least a couple of reasons. The first issue is that the tampering functions are chosen after the adversary has seen the public hash function. However, known results for universal hash functions typically require that they are chosen independently of the variables they are applied to. We circumvent this problem using the assumption that the tampering functions are chosen from a sufficiently small set. Any polynomial size set works, and for schemes with superpolynomial security even larger sets can be allowed.

Secondly, even if, say, $X_1 = T(X_0)$ has very low min-entropy, it is not true that the average entropy of X_0 is large when given X_1. It is easy to give examples of functions T where, for instance, $\mathbf{H}_\infty(X_1) = 1$ while the entropy of X_0 given X_1 is 0 half the time. We get around this problem by giving a characterization of what such "bad T's" must look like, which allows us to handle them by a rather complicated cases analysis. The by-product of this analysis is a general technical lemma showing that given an arbitrary chain of random variables $X_0, \ldots, X_t$ (such that X_0 is uniform), for an appropriate choice of values t and β there exists a point $j \in [t]$ in the chain such that all random variables before j have average min-entropy at least β even conditioned on $X_{j+1}, \ldots, X_t$. We believe the above "chaining lemma" is interesting in its own right (and can indeed be very

useful in the context of tamper resilience).

Due to its generality the above result suffers from two limitations. First, as already mentioned above (without making non-standard assumptions) the tampering has to satisfy a somewhat limited form of adaptivity. Second, the number of tampering queries one can tolerate is upper bounded by the length n of the secret key. While this is true in general for schemes without key update, for our general result the limitation is rather strong. More concretely, with appropriately chosen parameters our transformation yields schemes that can tolerate up to $O(\sqrt[3]{n})$ tampering queries. We can improve some of the parameters by modeling the hash function as a random oracle. We leave it as an important open question to further improve these parameters without the random oracle assumption.

As a second contribution, we follow Kalai *et al.* and show how some of these limitations can be circumvented by looking at specific cryptographic schemes. We show that for large classes of schemes we can obtain fully adaptive BLT security under a larger number of tampering queries. Moreover, by making an additional simple assumption on the hardware our results easily extend to the continuous tampering and leakage setting and achieve CLT security. We elaborate on this below.

Identification schemes. It is well known that the Generalized Okamoto identification scheme [32] provides security against bounded leakage from the secret key [2, 27]. In Section 4, we show that additionally it provides strong security against tampering attacks. While in general the tampered view may contain a polynomial number of faulty transcripts that may potentially reveal a large amount of information about the secret key, we can show that fortunately this is not the case for the Generalized Okamaoto scheme. More concretely, we are able to identify a short amount of information that for each tampering query allows us to simulate the corresponding faulty transcripts. Hence, BLT security of the Generalized Okamoto scheme is implied by its leakage resilience.

Our results on the Okamoto identification can be further generalized to a large class of identification schemes (and signature schemes based on the Fiat Shamir heuristic). More concretely, we shows that Σ-protocols where the secret key is significantly longer than the public key are BLT secure for a large number of tampering queries. We can instantiate our result with the generalized Guillou-Quisquater ID scheme [24], and its variant based on factoring [21] yielding tamper resilient identification based on factoring. We give more details in Section 4.

Interestingly, for Okamoto identification security still holds in a stronger model where the adversary is allowed to tamper not only with the secret key of the prover, but also with the description of the public parameters (i.e., the generator g of a group $\mathbb{G}$ of prime order p). The only restriction is that tampering with the public parameters is independent from tampering with the secret key. We also show that the latter restriction is necessary, by presenting explicit attacks when the adversary can tamper jointly with the secret key and the public parameters.

Public key encryption. We show how to construct IND-CCA secure public key encryption in the BLT model. To this end, we first introduce a weaker CPA-like security notion, where an adversary is given access to a restricted (faulty) decryption oracle. Instead of decrypting adversarial chosen ciphertexts such an oracle accepts inputs (m, r), encrypts the message m using randomness r under the original public key, and returns the decryption using the faulty secret key. This notion includes IND-CPA security as a special case when the tampering function is the identity function. Our notion allows the adversary to tamper adaptively with the secret key; intuitively this allows him

to learn faulty decryptions of ciphertexts for which he already knows the corresponding plaintext (under the original public key). We show how to instantiate our extended security notion under DDH. More concretely, we prove that the BHHO cryptosystem [7] is BLT and CPA secure. The proof uses similar ideas as in the proof of the Okamoto identification scheme.

We then show how to transform our extended CPA-like notion to CCA security in the BLT model. To this end, we follow the classical paradigm to transform IND-CPA security into IND-CCA security by adding an argument of "plaintext knowledge" π to the ciphertext. Our transformation requires a public tamper-proof common reference string similar to the work of Kalai *et al.* [26]. Intuitively this works because the argument π enforces the adversary to submit to the faulty decryption oracle only ciphertexts for which he knows the corresponding plaintext (and the randomness used to encrypt it). The pairs (m, r) can then be extracted from the argument π, allowing to reduce IND-CCA BLT security to our extended IND-CPA security notion.

Updating the key in the *i*Floppy model. As mentioned earlier, if the key is not updated BLT security is the best we can hope for. We show a generalization of an attack by Gennaro *et al.* [22] extending it to security notions which do not have the "verifiability" property required by the original attack [22]. This allows complete key recovery after $< |sk|$ tampering queries even for primitives such as weak PRFs or symmetric encryption.

To go beyond the bound of $|sk|$ tampering queries we may regularly update the secret key with fresh randomness, which renders information that the adversary has learned about earlier keys useless. The effectiveness of key updates in the context of tampering attacks has first been used in the important work of Kalai *et al.* [26]. We follow this idea but add an additional hardware assumption that allows for much simpler and more efficient key updates. More concretely, we propose the i*Floppy model* which is a variant of the floppy model proposed by Alwen *et al.* [2] and recently studied in depth by Agrawal *et al.* [1]. In the floppy model a user of a cryptodevice possesses a so-called *floppy* that stores an update key. The floppy is leakage and tamper proof and the update key that it holds is solely used to refresh the actual secret key kept on the cryptodevice. One may think of the floppy as a particularly secure device that the user keeps at home, while the cryptodevice, e.g., a smart-card, runs the actual cryptographic task and is used out in the wild prone to leakage and tampering attacks. We consider a variant called the *i*Floppy model (here "i" stands for individual). While in the floppy model of [1, 2] all users can potentially possess an identical floppy, in the *i*Floppy model we require that each user has an individual floppy storing some secret key related data. We note that from a practical point of view the *i*Floppy model is incomparable to the original floppy model. It may be more cumbersome to produce personalized floppies, but on the other hand, in practice one would not want to distribute floppies that all contain the same global update key as this constitutes a single point of failure: the device needs to be secure against attacks by its own user since once the update key is known, all bets are off.

We show in the *i*Floppy model a simple compiler that "boosts" any ID scheme with BLT security into a scheme with *continuous* leakage and tamper resilience (CLT security). Similarly, we show how to extend IND-CCA BLT security to the CLT setting for the BHHO cryptosystem (borrowing ideas from [1]). We emphasize that while the *i*Floppy model puts additional requirements on the way users must behave in order to guarantee security, it greatly simplifies cryptographic schemes, and allows us to base the security proof on standard assumptions in a very strong tampering model.

Tampering with the computation via the BRM. Finally, we make a simple observation showing that if we instantiate the above ID compiler with an ID scheme that is secure in the bounded retrieval model [10, 16, 2] we can provide security in the *i*Floppy model even when the adversary can replace the original cryptoscheme with an arbitrary adversarial chosen functionality, i.e., we can allow arbitrary tampering with the computation. While easy to prove, we believe this is nevertheless noteworthy: it seems to us that results in the BRM naturally provide some form of tamper resilience and leave it as an open question for future research to explore this direction further.

1.3 Roadmap

After recalling some basic notation in Section 2, we describe our main technical lemma and our generic transformation for semi-adaptive BLT security in Section 3. The BLT model and the results for ID schemes and IND-CCA PKE are presented in Section 4 and 5. Section 6 defines security in the *i*Floppy model for both ID schemes and PKE. Our variant of the attack from [22] is described in Appendix A. The results about tampering with the computation in the BRM can be found in Appendix B.

2 Preliminaries

We review the basic terminology used throughout the paper.

2.1 Notation

For $n \in \mathbb{N}$, we write $[n] := \{1, \ldots, n\}$. Given a set $\mathcal{S}$, we write $s \leftarrow \mathcal{S}$ to denote that element s is sampled uniformly from $\mathcal{S}$. If A is an algorithm, $y \leftarrow \mathsf{A}(x)$ denotes an execution of A with input x and output y; if A is randomized, then y is a random variable. Vectors are denoted in bold. Given a vector $\mathbf{x} = (x_1, \ldots, x_\ell)$ and some integer a, we write $a^{\mathbf{x}}$ for the vector $(a^{x_1}, \ldots, a^{x_\ell})$. The inner product of $\mathbf{x} = (x_1, \ldots, x_\ell)$ and $\mathbf{y} = (y_1, \ldots, y_\ell)$ is $\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{i=1}^{\ell} x_i \cdot y_i$.

We denote with k the security parameter. A function $\delta(k)$ is called *negligible* in k (or simply negligible) if it vanishes faster than the inverse of any polynomial in k. A machine A is called *probabilistic polynomial time* (PPT) if for any input $x \in \{0,1\}^*$ the computation of $\mathsf{A}(x)$ terminates in at most $poly(|x|)$ steps and A is probabilistic (i.e., it uses randomness as part of its logic). Random variables are usually denoted by capital letters. We sometimes abuse notation and denote a distribution and the corresponding random variable with the same capital letter, say X. We write $\mathsf{sup}(X)$ for the support of X. Given an event E, we let $X_{|E}$ be the conditional distribution of X conditioned on E happening. The statistical distance of two random variables X and Y, defined over a common set $\mathcal{S}$ is $\Delta(X;Y) = \frac{1}{2}\sum_{s \in \mathcal{S}} |\Pr[X = s] - \Pr[Y = s]|$. Given a random variable Z, the statistical distance of X and Y conditioned on Z is defined as $\Delta(X;Y|Z) = \Delta((X,Z);(Y,Z))$.

A *decision problem* related to a language $\mathfrak{L} \subseteq \{0,1\}^*$ requires to determine if a given string y is in $\mathfrak{L}$ or not. We can associate to any $\mathcal{NP}$-language $\mathfrak{L}$ a polynomial-time recognizable relation $\mathfrak{R} \subseteq \{0,1\}^* \times \{0,1\}^*$ defining $\mathfrak{L}$ itself, i.e. $\mathfrak{L} = \{y : \exists x \text{ s.t. } (y,x) \in \mathfrak{R}\}$ for $|x| \leq poly(|y|)$. The string x is called a *witness* for membership of $y \in \mathfrak{L}$.

2.2 Information Theory Basics

The min-entropy of a random variable X over a set $\mathcal{X}$ is defined as $\mathbf{H}_\infty(X) := -\log \max_x \Pr[X = x]$, and measures how X can be predicted by the best (unbounded) predictor. The conditional average min-entropy [14] of X given a random variable Z (over a set $\mathcal{Z}$) possibly dependent on X, is defined as

$$\widetilde{\mathbf{H}}_\infty(X|Z) := -\log \mathbb{E}_{z \leftarrow Z}[2^{-\mathbf{H}_\infty(X|Z=z)}] = \sum_{z \in \mathcal{Z}} \Pr[Z = z] \cdot 2^{-\mathbf{H}_\infty(X|Z=z)}.$$

Following [2], sometimes we rephrase the notion of conditional min-entropy in terms of predictors A that are given some information Z (presumably correlated with X), so $\widetilde{\mathbf{H}}_\infty(X|Z) = -\log(\max_{\mathsf{A}} \Pr[\mathsf{A}(Z) = X])$. The above notion of conditional min-entropy can be generalized to the case of interactive predictors A, which participate in some randomized experiment $\mathcal{E}$. An experiment is modeled as interaction between A and a challenger oracle $\mathcal{E}(\cdot)$ which can be randomized, stateful and interactive. Now the predictor $A^{\mathcal{E}}(\cdot)$ can act arbitrarily in the experiment with the challenger in order to predict X.

Definition 2.1 ([2])**.** The conditional min-entropy of a random variable X, conditioned on the experiment $\mathcal{E}$ is $\widetilde{\mathbf{H}}_\infty(X|\mathcal{E}) = -\log(\max_{\mathsf{A}} \Pr\left[A^{\mathcal{E}(\cdot)}() = X\right])$. In the special case that $\mathcal{E}$ is a non-interactive experiment which simply outputs a random variable Z, then $\widetilde{\mathbf{H}}_\infty(X|Z)$ can be written to denote $\widetilde{\mathbf{H}}_\infty(X|\mathcal{E})$ abusing the notion.

We will rely on the following basic properties (see [14, Lemma 2.2]).

Lemma 2.1. *For all random variables X, Z and Λ over sets $\mathcal{X}$, $\mathcal{Z}$ and $\{0,1\}^\lambda$ such that $\widetilde{\mathbf{H}}_\infty(X|Z) \geq \alpha$, we have that*

$$\widetilde{\mathbf{H}}_\infty(X|Z, \Lambda) \geq \widetilde{\mathbf{H}}_\infty(X|Z) - \lambda \geq \alpha - \lambda.$$

The above lemma can be easily extended to the case of random variables Λ with bounded support, i.e., $\widetilde{\mathbf{H}}_\infty(X|Z, \Lambda) \geq \widetilde{\mathbf{H}}_\infty(X|Z) - \log|\mathsf{sup}(\Lambda)|$.

Lemma 2.2. *For any $\epsilon > 0$, $\mathbf{H}_\infty(X|Z = z)$ is at least $\widetilde{\mathbf{H}}_\infty(X|Z) - \log(1/\epsilon)$ with probability at least $1 - \epsilon$ over the choice of z.*

2.3 Hard Relations

Let $\mathfrak{R}$ be a relation for some $\mathcal{NP}$-language $\mathfrak{L}$. We assume the existence of a probabilistic polynomial time algorithm Setup, called the setup algorithm, which on input 1^k outputs the description of public parameters pp for the relation $\mathfrak{R}$. Furthermore, we say that the *representation problem* is hard for $\mathfrak{R}$ if for all PPT adversaries A there exists a negligible function $\delta : \mathbb{N} \to [0, 1]$ such that

$$\Pr\left[x^\star \neq x; (y, x), (y, x^\star) \in \mathfrak{R} : (y, x, x^\star) \leftarrow \mathsf{A}(pp); pp \leftarrow \mathsf{Setup}(1^k)\right] \leq \delta(k).$$

Representation problem based on discrete log. Let Setup be a group generation algorithm that upon input 1^k outputs $(\mathbb{G}, g, p)$, where $\mathbb{G}$ is a group of prime order p with generator g. The Discrete Log assumption states that for all PPT adversaries A there exists a negligible function $\delta : \mathbb{N} \to [0,1]$ such that

$$\Pr\left[y = g^x : \ x \leftarrow \mathsf{A}(\mathbb{G}, g, p, y), y \leftarrow \mathbb{G}, (\mathbb{G}, g, p) \leftarrow \mathsf{Setup}(1^k)\right] \leq \delta(k).$$

Let $\ell \in \mathbb{N}$ be a function of the security parameter. Given a vector $\boldsymbol{\alpha} \in \mathbb{Z}_p^\ell$, define $g^{\boldsymbol{\alpha}} = (g_1, \ldots, g_\ell)$ and let $\mathbf{x} = (x_1, \ldots, x_\ell) \leftarrow \mathbb{Z}_p^\ell$. Define $y = \prod_{i=1}^{\ell} g_i^{x_i}$; the vector $\mathbf{x}$ is called a *representation* of y. We let $\mathfrak{R}_{\mathsf{DL}}$ be the relation corresponding to the representation problem, i.e. $(y, \mathbf{x}) \in \mathfrak{R}_{\mathsf{DL}}$ if and only if $\mathbf{x}$ is a representation of y with respect to $(\mathbb{G}, g, g^{\boldsymbol{\alpha}})$. We say that the ℓ-representation problem is hard in $\mathbb{G}$ if for all PPT adversaries A there exists a negligible function $\delta : \mathbb{N} \to [0,1]$ such that

$$\Pr\left[\mathbf{x}^\star \neq \mathbf{x}; (y, \mathbf{x}), (y, \mathbf{x}^\star) \in \mathfrak{R}_{\mathsf{DL}} : \ (y, \mathbf{x}, \mathbf{x}^\star) \leftarrow \mathsf{A}(\mathbb{G}, g, g^{\boldsymbol{\alpha}}); (\mathbb{G}, g, g^{\boldsymbol{\alpha}}) \leftarrow \mathsf{Setup}(1^k)\right] \leq \delta(k).$$

The ℓ-representation problem is equivalent to the Discrete Log problem [2, Lemma 4.1].

Decisional Diffie Hellman. Let Setup be a group generation algorithm that upon input 1^k outputs $(\mathbb{G}, g, p)$, where $\mathbb{G}$ is a group of prime order p with generator g. The Decisional Diffie Hellman (DDH) assumption states that for all PPT adversaries A there exists a negligible function $\delta : \mathbb{N} \to [0,1]$ such that

$$\Big|\Pr\left[\mathsf{A}(g, g^x, g^y, g^{xy}) = 1 : \ x, y \leftarrow \mathbb{Z}_p, (\mathbb{G}, g, p) \leftarrow \mathsf{Setup}(1^k)\right] - \Pr\left[\mathsf{A}(g, g^x, g^y, g^z) = 1 : \ x, y, z \leftarrow \mathbb{Z}_p, (\mathbb{G}, g, p) \leftarrow \mathsf{Setup}(1^k)\right]\Big| \leq \delta(k).$$

2.4 Signature Schemes

A signature scheme is a triple of algorithms $\mathcal{SIG} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vrfy})$ such that: (1) KGen takes the security parameter k as input and outputs a key pair (pk, sk); (2) Sign takes as input a message m and the secret key sk, and outputs a signature σ; (3)Vrfy takes as input a message-signature pair (m, σ) together with the public key pk and outputs a decision bit (indicating whether (m, σ) is a valid signature with respect to pk).

We require that for all messages m and for all keys $(pk, sk) \leftarrow \mathsf{KGen}(1^k)$, algorithm $\mathsf{Vrfy}(pk, m, \mathsf{Sign}(sk, m))$ outputs 1 with all but negligible probability. A signature scheme $\mathcal{SIG}$ is existentially unforgeable against chosen message attacks (EUF-CMA), if for all PPT adversaries A there exists a negligible function $\delta : \mathbb{N} \to [0,1]$ such that $\Pr[\mathsf{A} \text{ wins}] \leq \delta(k)$ in the following game:

1. The challenger samples $(pk, sk) \leftarrow \mathsf{KGen}(1^k)$ and gives pk to A.
2. The adversary is given oracle access to $\mathsf{Sign}(sk, \cdot)$.
3. Eventually A outputs a forgery $(m^\star, \sigma^\star)$ and *wins* if $\mathsf{Vrfy}(pk, (m^\star, \sigma^\star)) = 1$ and $m^\star$ was not asked to the signing oracle before.

2.5 True Simulation Extractibility

We recall the notion of true-simulation extractable (tSE) NIZKs [13]. This notion is similar to the notion of simulation-sound extractable NIZKs [23], with the difference that the adversary has oracle access to simulated proofs only of true statements (and not of arbitrary ones).

Let $\mathfrak{R}$ be an NP relation on pairs (y, x) with corresponding language $\mathfrak{L} = \{y : \exists x \text{ s.t. } (y, x) \in \mathfrak{R}\}$. A tSE NIZK proof system for $\mathfrak{R}$ is a triple of algorithm $(\mathsf{Gen}, \mathsf{Prove}, \mathsf{Verify})$ such that: (1) Algorithm Gen takes as input 1^k and generates a common reference string ω, a trapdoor tk and an extraction key ek; (2) Algorithm Prove^ω takes as input a pair (y, x) and produces an argument π which proves that $(y, x) \in \mathfrak{R}$; (3) Algorithm Verify^ω takes as input a pair (y, π) and checks the correctness of the argument π with respect to the public input y. Moreover, the following properties are satisfied:

Completeness. For all pairs $(y, x) \in \mathfrak{R}$, if $(\omega, \mathsf{tk}, \mathsf{ek}) \leftarrow \mathsf{Gen}(1^k)$ and $\pi \leftarrow \mathsf{Prove}^\omega(y, x)$ then $\mathsf{Verify}^\omega(y, \pi) = 1$.

Soundness. For any PPT adversary A, there exists a negligible function $\delta : \mathbb{N} \to [0, 1]$ such that

$$\Pr\left[\mathsf{Verify}^\omega(y, \pi^\star) = 1 \wedge y \notin \mathfrak{L} : (y, \pi^\star) \leftarrow \mathsf{A}(\omega); (\omega, \mathsf{tk}, \mathsf{ek}) \leftarrow \mathsf{Gen}(1^k)\right] \leq \delta(k).$$

Composable non-interactive zero knowledge. There exists a PPT simulator S such that, for any PPT adversary A, there exists a negligible function $\delta : \mathbb{N} \to [0, 1]$ such that $|\Pr[\mathsf{A} \text{ wins}] - \frac{1}{2}| \leq \delta(k)$ in the following game:

1. The challenger samples $(\omega, \mathsf{tk}, \mathsf{ek}) \leftarrow \mathsf{Gen}(1^k)$ and gives (ω, tk) to A.
2. A chooses $(y, x) \in \mathfrak{R}$ and gives these to the challenger.
3. The challenger samples $\pi_0 \leftarrow \mathsf{Prove}^\omega(y, x)$, $\pi_1 \leftarrow \mathsf{S}(y, \mathsf{tk})$, $b \in \{0, 1\}$ and gives π_b to A.
4. A outputs a bit b' and wins iff $b' = b$.

True simulation extractability. Define a simulation oracle $\mathsf{S}'_{\mathsf{tk}}(\cdot, \cdot)$ that takes as input a pair (y, x), checks if $(y, x) \in \mathfrak{R}$ and then it either outputs a simulated argument $\pi \leftarrow \mathsf{S}(y, \mathsf{tk})$ (ignoring x) in case the check succeeds or it outputs $\perp$ otherwise. There exists a PPT algorithm $\mathsf{Ext}(y, \pi, \mathsf{ek})$ such that, for all PPT adversaries A, there exists a negligible function $\delta : \mathbb{N} \to [0, 1]$ such that $|\Pr[\mathsf{A} \text{ wins}] - \frac{1}{2}| \leq \delta(k)$ in the following game:

1. The challenger samples $(\omega, \mathsf{tk}, \mathsf{ek}) \leftarrow \mathsf{Gen}(1^k)$ and gives ω to A.
2. $\mathsf{A}^{\mathsf{S}'_{\mathsf{tk}}(\cdot)}$ can adaptively access the simulation oracle $\mathsf{S}'_{\mathsf{tk}}(\cdot, \cdot)$.
3. Eventually A outputs a pair $(y^\star, \pi^\star)$.
4. The challenger runs $x^\star \leftarrow \mathsf{Ext}(y^\star, \pi^\star, \mathsf{ek})$.
5. A wins if: (a) $(y^\star, \pi^\star) \neq (y, \pi)$ for all pairs (y, π) returned by the simulation oracle; (b) $\mathsf{Verify}^\omega(y^\star, \pi^\star) = 1$; (c) $(y^\star, x^\star) \notin \mathfrak{R}$.

2.6 A Note on Deterministic vs Probabilistic Tampering

In this paper we assume the tampering functions chosen by the adversary to be deterministic. This is without loss of generality as the adversary can always hard-wire the "best" randomness into the function. Here, the best randomness refers to some specific choice of the random coins which would maximize the adversary's advantage. Moreover, in this work we model tampering functions by polynomial size circuits with an identical input/output domain.

3 Semi-Adaptive BLT Security for General Primitives

We show that "any" primitive (e.g., any PRF or signature scheme) can be made tamper resilient by combining it with a universal hash function (or by assuming that the primitive has some form of RA security). We put forward a notion of semi-adaptive BLT security for general primitives in Section 3.1. In Section 3.2 we describe our transformation based on universal hashing and state our main theorem (Theorem 3.1). Section 3.3 contains an high-level overview of the proof of Theorem 3.1; a formal proof appears in Section 3.4 and 3.5. Finally, in Section 3.6 we discuss a few extensions of our main theorem.

Notation for this section. In this section $n = poly(k)$ denotes the length of the key unless explicitly mentioned otherwise, where k is the security parameter. We say that a distribution X over a set $\mathcal{X}$ of size $|\mathcal{X}| = 2^n$ is (α, n)-good if $\mathbf{H}_\infty(X) \geq \alpha$ and $\Pr[X = x] \geq 2^{-n}$ for all $x \in \mathsf{sup}(X)$. Given some event E we write $\widetilde{\mathbf{H}}_\infty(X|Y_1, Y_2, \ldots, E)$ to denote that every random variable is conditioned on the event E.

3.1 Abstract Games with Tampering

We start by defining a general security definition for abstract security games of cryptographic scheme $\mathcal{CS}$. Similarly to [15], we consider an experiment played between an adversary A and a challenger C that is defined by the security game.

Definition 3.1 (Abstract security game)**.** The security of a cryptographic scheme $\mathcal{CS}$ is defined via an interactive game $\mathsf{Game}^{\mathcal{CS}}_{\mathsf{C},\mathsf{A}}$ between a probabilistic attacker A and a probabilistic challenger $\mathsf{C}(\cdot)$, where $\mathsf{C}(\cdot)$ is fixed by the definition of $\mathcal{CS}$. For security parameter k the security game $\mathsf{Game}^{\mathcal{CS}}_{\mathsf{C},\mathsf{A}}(1^k)$ is given below.

1. At the beginning of the game, we sample public parameter pp and a uniformly chosen secret key $X \leftarrow \mathcal{X}$. Notice that in case of a public key primitive, pp contains the public key that depends on X. The public parameters pp are given to A and (pp, X) is given to C.
2. The game can have an arbitrary structure but at the end $\mathsf{C}(pp, X)$ outputs a bit that specifies the output of the game, where 1 indicates that A has won the game.

For *unpredictability* security games we say that a scheme is $\delta(k)$-secure if for any PPT adversary A the advantage of A is

$$\Pr[\mathsf{Game}^{\mathcal{CS}}_{\mathsf{C},\mathsf{A}}(1^k) = 1] \leq \delta(k).$$

Experiment $\mathsf{Game}^{\mathcal{CS}}_{\mathsf{A},\mathsf{C}_{t,\lambda,v}}$

The experiment features a challenger $\mathsf{C}_{t,\lambda,v}$, defined as follows:

1. $\mathsf{C}_{t,\lambda,v}$ takes inputs (pp, X) from the game and obtains a description of a leakage function $L : \mathcal{X} \to \{0,1\}^\lambda$ and a set of tolerated tampering functions $\mathcal{T}$ from A. It gives $pp, L(X)$ to the adversary.
2. The adversary can interact with the underlying challenger $\mathsf{C}(pp, X)$ in an arbitrary way as specified in Definition 3.1.
3. The adversary submits a set of tampering functions $(T_1, \ldots, T_t) \in \mathcal{T}$ to the challenger $\mathsf{C}_{t,\lambda,v}$.
4. For all $i \in [t]$ let $X^{(i)} = T_i(X^{(i-1)})$ where $X^{(0)} = X$. Challenger $\mathsf{C}_{t,\lambda,v}$ instantiates t challengers $\mathsf{C}(pp, X^{(1)}), \ldots, \mathsf{C}(pp, X^{(t)})$ and allows A to interact with each of these oracles.
5. At the end $\mathsf{C}(pp, X)$ outputs a bit b indicating whether A has won the game against $\mathsf{C}(pp, X)$. Let b denote the output of $\mathsf{C}_{t,\lambda,v}$.

Figure 1: A description of experiment $\mathsf{Game}^{\mathcal{CS}}_{\mathsf{A},\mathsf{C}_{t,\lambda,v}}$, modeling semi-adaptive BLT security

For *indistinguishability* games we say that a scheme is $\delta(k)$-secure if for any PPT adversary A the advantage of A is

$$\Pr[\mathsf{Game}^{\mathcal{CS}}_{\mathsf{C},\mathsf{A}}(1^k) = 1] - 1/2 \leq \delta(k).$$

For indistinguishability games, we assume wlog. that the challenger C internally keeps a bit b and A submits as its last message to C a bit b'. If $b = b'$ then the challenger returns 1; otherwise 0. In the following, we will usually omit the parameter $\delta(k)$ and just say that a scheme is secure if $\delta(k)$ is negligible in k.

We now extend the above definition to model an adversary A that is allowed to leak a certain amount of information (say λ bits) about the original key X and also to tamper with X for a bounded number of times (say some value $t \in \mathbb{N}$). The adversary has to commit to the entire sequence of tampering functions $(T_1, \ldots, T_t)$; however the particular sequence of functions to use can be chosen adaptively (from some set $\mathcal{T}$ of possible sequences) after seeing the public parameters and interacting with the challenger $\mathsf{C}(pp, X)$. We refer to this model as the *semi-adaptive* model.

Let λ, t, v be parameters and let $\mathcal{T} := \{(T_1, \ldots, T_t) : \ T_i : \mathcal{X} \to \mathcal{X}\}$ be some set of t-tuples of functions with $|\mathcal{T}| = v$. We define a special type of challenger, $\mathsf{C}_{t,\lambda,v}$, that models semi-adaptive tampering and leakage attacks on the secret key and can be based on any standard challenger C from Definition 3.1. Challenger $\mathsf{C}_{t,\lambda,v}$ runs in $\mathsf{Game}^{\mathcal{CS}}_{\mathsf{A},\mathsf{C}_{t,\lambda,v}}$, as described in Figure 1. Below we define security of a cryptosystem $\mathcal{CS}$ in the presence of (semi-adaptive) bounded tampering and leakage attacks (semi-adaptive BLT security).

Definition 3.2 (Semi-adaptive BLT security of $\mathcal{CS}$)**.** We say that a cryptographic scheme $\mathcal{CS}$ is $(\lambda(k), t(k), \delta(k), v(k))$-secure in the semi-adaptive BLT model if for all PPT adversaries A we have $\Pr[\mathsf{Game}^{\mathcal{CS}}_{\mathsf{A},\mathsf{C}_{t,\lambda,v}}(1^k) = 1] \leq \mathit{negl}(k)$. Here, $|\mathcal{T}| = v$ and each element of $\mathcal{T}$ is a tuple $(T_1, \ldots, T_t)$ of tampering functions.

Some remarks are in order to explain $\mathsf{Game}^{\mathcal{CS}}_{\mathsf{A},\mathsf{C}_{t,\lambda,v}}$. First, A's view now includes the leakage as well as the view that the adversary obtains by interacting with tampered keys. Second, A wins if he breaks the security notion of $\mathcal{CS}$ with respect to the *original* key. For public key primitives this

implies that the adversary has to break the security of the scheme with respect to the public key. For secret key primitives such as PRFs or symmetric encryption it says that A has to break the security notion with respect to the initial secret key X. Finally, we emphasize that A can interact with the tampered challenge oracles in any given order, i.e., A may interact first with $\mathsf{C}(pp, X^{(i+1)})$ before calling $\mathsf{C}(pp, X^{(i)})$. Moreover, he may jump back and forth between these oracles.

3.2 A General Transformation

We now describe a general transformation to leverage security of a cryptographic scheme $\mathcal{CS}$ (as per Definition 3.1) to semi-adaptive BLT security (as per Definition 3.2). The transformation is based on a family $\mathcal{H} = \{h_S : \mathcal{X} \to \mathcal{Y}\}$ of $(2t+1)$-wise independent hash functions. Recall that $\mathcal{H}$ is called t-wise independent if for any sequence of distinct elements $X^{(1)}, \ldots, X^{(t)} \in \mathcal{X}$ the random variables $h_S(X^{(1)}), \ldots, h_S(X^{(t)})$ are uniform, where $h_S \leftarrow \mathcal{H}$.[3]

The transformation. Consider a cryptographic scheme $\mathcal{CS}$ running a cryptographic algorithm CS that operates on some public parameters pp, a uniformly chosen key $K \leftarrow \mathcal{Y}$ and takes some additional input $M \in \mathcal{M}$ to compute the output $Z \leftarrow \mathsf{CS}(pp, K, M)$. For instance, $\mathcal{CS}$ may be a block cipher and CS the associated encryption algorithm with M being the message and Z the corresponding ciphertext. We transform $\mathcal{CS}$ into $\mathcal{CS}'$ as follows. Let $\mathcal{H} = \{h_S : \mathcal{X} \to \mathcal{Y}\}_{S \in \mathcal{S}}$ be a family of $2(t+1)$-wise independent hash functions. At setup we sample the public parameters pp according to $\mathcal{CS}$, choose a hash function key $S \leftarrow \mathcal{S}$ and sample a key $X \leftarrow \mathcal{X}$ uniformly at random. Notice here the difference between the key space $\mathcal{Y}$ for $\mathcal{CS}$ and $\mathcal{X}$ for the transformed scheme $\mathcal{CS}'$. We will discuss the relation between these two sets in further detail below. To compute the cryptographic algorithm CS' on some input M, we run $Z \leftarrow \mathsf{CS}(pp, h_S(X), M)$. Notice that we first map the key X for $\mathcal{CS}'$ to the key K for the underlying cryptoscheme $\mathcal{CS}$ by applying the hash function.

The theorem below states that the above transformation is secure in the semi-adaptive BLT model whenever $\mathcal{CS}$ is secure in the standard sense.

Theorem 3.1. *If $\mathcal{CS}$ is secure and $\mathcal{H} = \{h_S : \mathcal{X} \to \mathcal{Y}\}_{S \in \mathcal{S}}$ is a family of $2(t+1)$-wise independent hash functions with $|\mathcal{X}| = 2^n$ and $|\mathcal{Y}| = 2^\ell$, then we have that $\mathcal{CS}'$ is (λ, t, δ, v)-secure in the semi-adaptive BLT model, where*

$$\lambda = O(\sqrt[3]{n}) \qquad t = O(\sqrt[3]{n}) \qquad \delta \leq negl(k) \qquad v = O(n^d) \qquad \ell = O(\sqrt[4]{n}),$$

for some constant $d > 0$.

Concretely, we can think of $\mathcal{CS}$ being a PRF (or a signature scheme) with security in the standard sense, i.e., the adversary has negligible advantage when playing against the underlying challenger. The Theorem 3.1 says that, for sufficiently large n, the transformed PRF $\mathcal{CS}'$ achieves semi-adaptive BLT security against adversaries tampering $O(\sqrt[3]{n})$ times and leaking $O(\sqrt[3]{n})$ bits from the original key. Notice that the hash function compresses the n-bit input to $O(\sqrt[4]{n})$ bits and the set of admissible (sequences of) tampering functions has size $O(n^d)$ for some constant $d > 0$. Notice that if the underlying primitive is super-polynomial secure than we can increase the size of

[3] A concrete construction is given by the following function $h_S : \mathbb{Z}_p \to \mathbb{Z}_p$: Sample S by choosing t random elements $s_0, s_1, \ldots, s_{t-1} \leftarrow \mathbb{Z}_p^t$ and define $h_S(X) = s_0 + s_1 \cdot X + \ldots + s_{t-1} \cdot X^{t-1} \bmod p$.

admissible tampering functions. In the extreme case when the underlying primitive has exponential security, the size of $\mathcal{T}$ may be sub-exponentially large.

We emphasize that we can obtain stronger leakage resilience as we inherit the security properties from the underlying cryptoscheme $\mathcal{CS}$. Hence, if $\mathcal{CS}$ is secure against adaptive leakage attacks from the key K, then also $\mathcal{CS}'$ is secure against adaptive leakage attacks from the key $h_S(X)$ used by the actual cryptographic scheme.

3.3 Outline of the Proof

We explain the intuition and the main ideas behind the proof of Theorem 3.1. The proof is by reduction: Given an adversary A with non-negligible advantage in the semi-adaptive BLT game for $\mathcal{CS}'$ (cf. Definition 3.2), we build an adversary B against standard security of $\mathcal{CS}$ (cf. Definition 3.1). The main difficulty is that B has only access to a challenger $\mathsf{C}(pp, K)$ (for some uniform K), so it is not a priori clear how B can answer A's tampering queries and simulate the oracle $\mathsf{C}_{t,\lambda,v}$.

The idea is to let B sample the initial key $X^{(0)}$ independently of the target key K (which is anyway not known to B) and compute the keys $X^{(1)}, \ldots, X^{(t)}$ as specified by the tampering functions T_i in order to simulate the tampered view of A. To give a first hint why this may indeed be a good strategy, consider the simple case where all tampered keys have high min-entropy (say higher than some threshold β). In this case, we can rely on a property of $2(t+1)$-wise independent hashing, namely for a uniformly sampled hash function h_S the tuple $(h_S(X^{(0)}), h_S(X^{(1)}), \ldots, h_S(X^{(t)}))$ is statistically close to uniform and thus B's simulation of A's view is indistinguishable from the real view. The exact property we need from $\mathcal{H}$ is formalized in the following lemma. The proof is a straightforward extension of [28, Lemma 3.2] and is deferred to Appendix C.

Lemma 3.1. *Let $(X_1, X_2, \ldots, X_t) \in \mathcal{X}^t$ be t (possibly dependent) random variables such that $\mathbf{H}_\infty(X_i) \geq \beta$ and $(X_1, \ldots, X_t)$ are pairwise different. Let $\mathcal{H} = \{h_S : \mathcal{X} \to \mathcal{Y}\}$ be a family of $2t$-wise independent hash functions, with $|\mathcal{Y}| = 2^\ell$. Then for random $h_S \leftarrow \mathcal{H}$ we have that*

$$\Delta((h_S, h_S(X_1), h_S(X_2), \ldots, h_S(X_t)); (h_S, \underbrace{U_\mathcal{Y}, \ldots, U_\mathcal{Y}}_{t \text{ times}})) \leq \frac{t}{2} \cdot 2^{(t \cdot \ell - \beta)/2}.$$

Of course, in our general tampering model nothing guarantees that all keys have high min-entropy, and hence we cannot immediately apply Lemma 3.1. At this point, a careful reader may object that at the end this does not matter too much: if the compression of the hash function is high enough (as it is the case for our choice of $\ell \approx \sqrt[4]{n}$ in Theorem 3.1) the hashed keys are short anyway, and thus the entropy of $X^{(0)}$ given the hash of the tampered keys remains high. At this point it looks tempting to apply the leftover hash lemma, and argue that $h_S(X^{(0)})$ is statistically close to uniform even given the hashed tampered keys. The leftover hash lemma, however, requires that the key S can be sampled uniformly and independently from the distribution of $X^{(0)}$. Unfortunately, the conditional distribution of $X^{(0)}$ (given the tampered hashed keys) may now depend on S, and we cannot apply the leftover hash lemma directly.

To deal with low min-entropy keys we prove a technical lemma stating that given an arbitrary chain of random variables $X^{(0)} \xrightarrow{T_1} X^{(1)} \xrightarrow{T_2} \ldots \xrightarrow{T_t} X^{(t)}$ (such that $X^{(0)}$ has high min-entropy) for any t arbitrary functions $T_i : \mathcal{X} \to \mathcal{X}$, for an appropriate choice of values t and β there exists a point $i \in [t]$ in the chain such that all random variables before i have average min-entropy at least β even conditioned on $X^{(i+1)}, \ldots, X^{(t)}$. We state the lemma below. The proof is quite involved and is given in Appendix D.

Experiment Real **vs.** Sim

1. Experiment $\mathsf{Real}(X, (T_1, \ldots, T_t), L)$*:* Let $X^{(0)}$ be a random variable with distribution X and $h_S \leftarrow \mathcal{H}$ a uniformly sampled hash function. For $i \in [t]$ denote by $X^{(i)} = T_i(X^{(i-1)})$, then

$$\mathsf{Real} := (D_0, \ldots, D_{t+2}) = \Big(h_S(X^{(0)}), \ldots, h_S(X^{(t)}), L(X^{(0)}), S\Big)$$

2. Experiment $\mathsf{Sim}(X, (T_1, \ldots, T_t), L)$*:* Let $X^{(0)}$ be a random variable with distribution X and $h_S \leftarrow \mathcal{H}$ a uniformly sampled hash function. For $i \in [t]$, denote by $X^{(i)} = T_i(X^{(i-1)})$, and proceed as follows

   Sample $D_0 \leftarrow U_{\mathcal{Y}}$.
   For $i \in [t]$ compute:
   If $X^{(i)} \neq X^{(0)}$ then $D_i = h_S(X^{(i)})$
   Else $D_i = D_0$
   Output $\mathsf{Sim} = (D_0, \ldots, D_t, L(X^{(0)}), S)$.

Figure 2: Experiment Real denotes the real tampering experiment and Sim our simulation.

Lemma 3.2. *For $n \in \mathbb{N}_{>1}$ let $\alpha, \beta, t, \epsilon$ be some parameters where $t \in \mathbb{N}$, $0 < \alpha \leq n$, $\beta > 0$, $\epsilon \in (0, 1]$ and $t \leq \frac{\alpha-\beta}{\beta+2\sqrt{n}}$. Let $\mathcal{X}$ be some set of size $|\mathcal{X}| = 2^n$ and let $X^{(0)}$ be a (α, n)-good distribution over $\mathcal{X}$. For $i \in [t]$ let $T_i : \mathcal{X} \to \mathcal{X}$ be arbitrary functions and $X^{(i)} = T_i(X^{(i-1)})$. There exists an event E such that:*

(i) If $\Pr[E] > 0$, for all $i \in [t]$, $\mathbf{H}_\infty(X^{(i)}_{|E}) \geq \beta$.

(ii) If $\Pr\left[\overline{E}\right] \geq \epsilon$ there exists an index $j \in [t]$ such that

$$\widetilde{\mathbf{H}}_\infty(X^{(j-1)}_{|\overline{E}} | X^{(j)}_{|\overline{E}}) \geq \beta - \log \frac{t}{\epsilon}.$$

Instead of the real experiment we can now turn to a mental experiment where at some point in the chain we reveal an entire source $X^{(i)}$. By Lemma 3.2 we are guaranteed that $X^{(0)}, X^{(1)}, \ldots, X^{(i-1)}$ individually all have high min-entropy even given $X^{(i)}$, which allows us to apply Lemma 3.1 and conclude that $h_S(X^{(0)}), \ldots, h_S(X^{(i-1)})$ are jointly close to uniform. Notice that in the mental experiment clearly the remaining sources $X^{(0)}, X^{(1)}, \ldots, X^{(i-1)}$ remain independent from S even given $X^{(i)}$. At this point we are almost done except for two technical difficulties: (1) Lemma 3.2 requires that all $X^{(j)}$ (for $j < i$) are pairwise distinct, and (2) the adversary picks its tampering choice *adaptively* from a fixed set $\mathcal{T}$ after seeing the key for the hash function S and after interacting with the original challenger (the so-called semi-adaptive model). We solve the first by changing the above mental experiment and eliminate all sources that appear multiple times in the source chain. We then show that given a short advice we can re-sample the complete $X^{(0)}, \ldots, X^{(t)}$ from the reduced chain. To complete the proof, we address the semi-adaptivity mentioned in (2) by a counting argument as the size of the set of potential tampering queries $\mathcal{T}$ is not too big (polynomial in the security parameter).

We conclude the above outline by defining two experiments that describe how the keys $X^{(0)}, X^{(1)}, \ldots, X^{(t)}$ are sampled in the real game and in the simulation. For $t, \lambda \in \mathbb{N}$ and any set of functions $T_1, \ldots, T_t : \mathcal{X} \to \mathcal{X}$, $L : \mathcal{X} \to \{0,1\}^\lambda$ consider the two experiments as given in Figure 2.
In the lemma below we show that for a distribution X with a sufficient amount of min-entropy

and certain set of carefully chosen parameters the distance between $\mathsf{Real}(X, (T_1, \ldots, T_t), L)$ and $\mathsf{Sim}(X, (T_1, \ldots, T_t), L)$ is statistically close. We will in the following omit to explicitly mention the inputs to the experiments.

Lemma 3.3. *Let $k \in \mathbb{N}$ be the security parameter and $n, t, \lambda, \epsilon, \ell, \alpha$ be functions in k such that $\lambda, t < \alpha \leq n$ and $\epsilon \in (0, 1]$. Let $\mathcal{H} = \{h_S : \mathcal{X} \to \mathcal{Y}\}$ be a family of $2(t+1)$-wise independent hash functions. Let $|\mathcal{X}| = 2^n$, $|\mathcal{Y}| = 2^\ell$ and X be an (α, n)-good distribution over $\mathcal{X}$. For any set of functions $T_1, \ldots, T_t : \mathcal{X} \to \mathcal{X}$, $L : \mathcal{X} \to \{0,1\}^\lambda$ as specified above:*

$$\Delta(\mathsf{Real}; \mathsf{Sim}) \leq t \cdot 2^{((t+1)\cdot\ell - c)/2} + 4\epsilon,$$

where $c := \beta - 2\log t/\epsilon - \lambda - 2t\log(t)$ and $\beta := \frac{\alpha - 2t\sqrt{n}}{t+1}$.

3.4 Proof of Lemma 3.3

We start by describing a distribution D^1 that together with a short advice Z allows to sample Real. Distribution D^1 is defined exactly as Real except that it only contains distinct values (in particular notice that D^1 can contain less values than Real). More precisely, D^1 is sampled as follows:

1. Let $X^{(0)}$ be distributed according to X and compute $X^{(i)} = T_i(X^{(i-1)})$ (this is exactly as in Real).
2. For all $i \in [t]$ output $h_S(X^{(i)})$ if for all $j < i$ we have $X^{(i)} \neq X^{(j)}$. Denote these outputs by D^1. We also output $(L(X^{(0)}), S)$.

The advice Z (depending on $X^{(0)}$ and the functions $T_1, \ldots, T_t$) describes where the values from D^1 appear in Real. An easy way to describe such an advice Z requires $t^2/2$ bits. A more thorough analysis shows that one can encode the information necessary to map from D^1 to Real by $2t\log(t)$ bits.[4] In the following we denote the mapping algorithm that maps (D^1, Z) to Real as $\mathsf{Samp}(D^1, Z)$. Clearly, $\mathsf{Samp}(D^1, Z)$ and Real are identically distributed and all values in D^1 are distinct. For ease of notation we will reuse the parameter t to denote the number of elements in D^1.

Claim 1. *Let $\beta = \frac{\alpha - 2t\sqrt{n}}{t+1}$. There exists an $i \in [t]$ and an event Good such that $\Pr[Good] \geq 1 - \epsilon$ and*

$$\widetilde{\mathbf{H}}_\infty(X^{(i)} | X^{(i+1)}, L(X^{(0)}), S, Z, Good) \geq \beta - \log t/\epsilon - \lambda - 2t\log(t). \tag{1}$$

In the above $X^{(t+1)}$ denotes a random variable that is chosen uniformly and independently from $\mathcal{X}$.

Proof. Recall that by putting *Good* in the condition of (1) we denote that all random variables are conditioned on the fact that *Good* happens. We prove this statement by relying on Lemma 3.2 which shows that each $X^{(i)}$ has average min-entropy at least $\beta - \log t/\epsilon$. Lemma 3.2 puts a constraint on β:

$$\beta \leq \frac{\alpha - 2t\sqrt{n}}{t+1}$$

Clearly, our choice of β satisfies the above constraint. As $X^{(0)}$ is (α, n)-good, we can now apply Lemma 3.2:

[4] This can be done by first describing for each element in D^1 how often it appears in Real and then by defining a mapping that maps each element to its position in Real. Each of these steps require at most $t\log(t)$ bits.

1. If $\Pr[E] > 0$, for all $i \in [t]$: $\mathbf{H}_\infty(X^{(i)}_{|E}) \geq \beta$,

2. If $\Pr[\overline{E}] \geq \epsilon$ then there exists $i \in [t]$ such that : $\widetilde{\mathbf{H}}_\infty(X^{(i-1)}_{|\overline{E}}|X^{(i)}_{|\overline{E}}) \geq \beta - \log \frac{t}{\epsilon}$.

Consider now the setting when $\Pr[E] > 0$. Hence we know by Step (1) from above that for all $i \in [t]$: $\mathbf{H}_\infty(X^{(i)}_{|E}) \geq \beta$, and in particular $\mathbf{H}_\infty(X^{(t)}_{|E}) \geq \beta$. As $X^{(t+1)}$ is uniformly and independently chosen from all other variables, we get in this case that

$$\widetilde{\mathbf{H}}_\infty(X^{(t)}_{|E}|X^{(t+1)}_{|E}) \geq \beta \geq \beta - \log t/\epsilon. \tag{2}$$

Again if $\Pr[\overline{E}] \geq \epsilon$ then by Step (2) from above there exists an $i \in [t]$ such that

$$\widetilde{\mathbf{H}}_\infty(X^{(i-1)}_{|\overline{E}}|X^{(i)}_{|\overline{E}}) \geq \beta - \log t/\epsilon. \tag{3}$$

We define *Good* as follows: $\mathit{Good} = E$ if $\Pr[\overline{E}] < \epsilon$ and $\mathit{Good} = \Omega$ if $\Pr[\overline{E}] \geq \epsilon$ where Ω denotes the whole probability space. We can bound the probability of the event *Good* considering two cases:

- When $\Pr\left[\overline{E}\right] \geq \epsilon$, then $\Pr[\mathit{Good}] = 1$.
- When $\Pr\left[\overline{E}\right] < \epsilon$ then, $\Pr[\mathit{Good}] = \Pr[E] > 1 - \epsilon$.

So clearly $\Pr[\mathit{Good}] > 1 - \epsilon$.

We conclude that there must exist an $i \in [t]$ such that

$$\begin{aligned}
\widetilde{\mathbf{H}}_\infty(X^{(i)}|X^{(i+1)}, L(X^{(0)}), S, Z, \mathit{Good}) &\geq \widetilde{\mathbf{H}}_\infty(X^{(i)}|X^{(i+1)}, S, \mathit{Good}) - \lambda - 2t\log(t) && (4)\\
&= \widetilde{\mathbf{H}}_\infty(X^{(i)}|X^{(i+1)}, \mathit{Good}) - \lambda - 2t\log(t) && (5)\\
&\geq \beta - \log t/\epsilon - \lambda - 2t\log(t). && (6)
\end{aligned}$$

Eq. (4) follows from the chain rule for conditional average min entropy (cf. Lemma 2.1). Eq. (5) holds because S is chosen uniformly and independently from all other variables. Finally, as either E or $\overline{E}$ must happen and we condition on *Good*, we get from Eq. (2) and Eq. (3) that Eq. (6) holds. This concludes the proof of the claim. □

By using the union bound and Lemma 2.1, we can now restate Claim 1 in terms of min-entropy and condition all random variables on event *Good* happening. Thus we get that there exists an $i \in [t]$ such that with probability at least $1 - 2\epsilon$ the following holds:

$$\mathbf{H}_\infty(X^{(i)}|(X^{(i+1)}, L(X^{(0)}), S, Z) = r) \geq \beta - 2\log t/\epsilon - \lambda - 2t\log(t).$$

Recall that $X^{(i)}$ can be computed as a (deterministic) function from $X^{(j)}$ where $j < i$. Hence, the above holds for all $X^{(j)}$ where $j \leq i$, i.e.,

$$\mathbf{H}_\infty(X^{(j)}|(X^{(i+1)}, L(X^{(0)}), S, Z) = r) \geq \beta - 2\log t/\epsilon - \lambda - 2t\log(t) =: c.$$

As with probability at least $1 - 2\epsilon$ all $X^{(j)}$ individually have min-entropy c and by assumption all $X^{(j)}$ are distinct, we can apply Lemma 3.1:

$$\Delta((D^1, Z); (\underbrace{\overbrace{U_{\mathcal{Y}}, \ldots, U_{\mathcal{Y}}}^{i+1 \text{ times}}, X^{(i+1)}, L(X^{(0)}), S}_{D^2}, Z)) \leq t2^{((t+1)\cdot\ell - c)/2 - 1} + 2\epsilon =: \epsilon'.$$

As Samp is a deterministic algorithm, the above implies:

$$\Delta(\mathsf{Samp}(D^1, Z); \mathsf{Samp}(D^2, Z)) \leq \epsilon'.$$

Notice that in $\mathsf{Samp}(D^2, Z)$ the first $i+1$ values are now sampled uniformly and independently from $U_{\mathcal{Y}}$. Consider now a distribution D^3 where only the first element is replaced by $U_{\mathcal{Y}}$ and the following i elements are computed correctly as the output of the hash function h_S. By a standard argument, we get

$$\Delta(\mathsf{Samp}(D^1, Z); \mathsf{Samp}(D^3, Z)) \leq 2\epsilon'.$$

To conclude the proof notice that Real and Sim are identically distributed except for the effect that the first element has on the two distributions.[5] Hence, $\mathsf{Samp}(D^3, Z)$ and Sim are identically distributed. As moreover $\mathsf{Samp}(D^1, Z)$ and Real are identically distributed this concludes the proof. □

Some comments are in order to explain the mechanics of the parameters defining the statistical distance between Real and Sim in Lemma 3.3. To obtain a negligible quantity (as we will need for the proof of Theorem 3.1 in Section 3.5), the value c must be chosen to be sufficiently larger than the value $(t+1) \cdot \ell$; this shows a clear trade-off between the value of t and the value of ℓ. We instantiate Lemma 3.3 with concrete values in the following corollary. It shows a setting where we try to maximize the number of tampering queries we can tolerate by using a very high compression factor.

Corollary 1. *For sufficiently large n, if we set $\ell = O(\sqrt[4]{n})$, $\lambda = O(\sqrt[3]{n})$, $\alpha = n - O(\sqrt[3]{n})$ and $\epsilon = \exp(-\Theta(\sqrt[3]{n}))$ in Lemma 3.3 we get $t = O(\sqrt[3]{n})$ for which the distance $\Delta(\mathsf{Real}; \mathsf{Sim}) \leq \exp(-\Omega(\sqrt[3]{n}))$ in n.*

3.5 Proof of Theorem 3.1

We now turn to the proof of Theorem 3.1.

Proof of Theorem 3.1. Suppose there exists an adversary A and a polynomial $p(.)$ such that A breaks the (λ, t, v) semi-adaptive BLT security of $\mathcal{CS}'$ with advantage at least $1/p(k)$ for infinitely many k. Then, we construct an adversary B that breaks the security of $\mathcal{CS}$ according to the challenge oracle $\mathsf{C}(pp, K)$ for some uniform key $K \leftarrow \mathcal{Y}$ with advantage at least $1/p'(k)$ for some polynomial $p'(.)$. To this end, adversary B needs to simulate the environment specified by $\mathsf{C}_{t,\lambda,v}$ to A given only access to its target oracle $\mathsf{C}(pp, K)$. At a high-level this simulation is carried out as follows: B uses its target oracle $\mathsf{C}(pp, K)$ to simulate the interaction of A with the cryptoscheme running on the original key $h_S(X^{(0)})$, and simulates the tampered view with keys that are sampled uniformly and independently. The simulation closely follows the structure of the tampering challenger as specified in Section 1 and is given below:

1. In the first step B receives a leakage function $L : \mathcal{X} \to \{0,1\}^\lambda$ and a set of tolerated tampering functions $\mathcal{T}$ from A. It also receives the public parameters pp from its own target game. B chooses uniformly at random an index $j^* \in [v]$. Recall that $v = |\mathcal{T}| = O(n^d)$ for some constant d.

[5] In both cases we start with an element sampled from X and apply the functions $T_1, \ldots, T_t$ to it.

2. B samples a random key S for the hash function and uniformly at random an initial key $X^{(0)}$ from $\mathcal{X}$. It forwards $L(X^{(0)})$ and $pp' = (pp, S)$ as the public parameters to A.

3. B uses its challenge oracle $\mathsf{C}(pp, K)$ to simulate A's interaction with the original (un-tampered) key.

4. B receives a tuple of tampering functions $V_j = (T_1, \ldots, T_t) \in \mathcal{T}$ from A. If $j \neq j^*$ then we proceed as follows:

   (a) If B runs an unpredictability game, then it aborts.

   (b) If B runs an indistinguishability game, then it samples a random bit b and submits b as its last message to its challenge oracle C.

5. B computes $X^{(i)} = T_i(X^{(i-1)})$ and simulates interactions with the oracles as follows:

   (a) For all $i \geq 1$ with $X^{(i)} \neq X^{(0)}$, it uses $X^{(i)}$ to simulate A's interaction with the challengers $\mathsf{C}(pp', h_S(X^{(i)}))$. As $X^{(i)}$ is known to B this can be done efficiently.

   (b) For all $i \geq 0$ with $X^{(i)} = X^{(0)}$, it uses its target oracle to simulate A's view. Notice that this includes the case when A interacts with the scheme running on the original key $X^{(0)}$.

We argue that when A wins the tampering game with advantage at least $1/p(k)$, then B wins the underlying game against challenger C with advantage $1/p'(k)$. To this end, we first show that conditioned on $j = j^*$ the view of the adversary in the simulation and the adversary's view in the real experiment are statistically close. If $j = j^*$ the simulation above is identically distributed to the simulation given in Sim from Figure 2. This follows from the following observations:

1. In the simulation A is committed to the tampering option j^* *before* he starts to interact with the challenge oracles as otherwise B will abort the simulation. Notice that this commitment is in particular before seeing the hash key S, and the view with the original key. Hence, B's simulation corresponds to the non-adaptive case as given in Sim.

2. B uses its own challenge oracle running with a uniform key to simulate A's un-tampered view. This is exactly as in the simulation Sim from Figure 2, where we replace the first output of the hash function with a uniformly and independently sampled value (independently of S and the first input to the hash function).

3. B simulates the tampering queries by using an initial input $X^{(0)}$ for the hash function that is chosen independently from the un-tampered view. That is exactly what happens in Sim.

The above concludes that the simulation of B and the simulation given in Sim are identical if $j = j^*$. By Lemma 3.3 and Corollary 1 we get for the choice of parameters given in the theorem's statement (notice that this choice corresponds to the parameters of Corollary 1) that for $j = j^*$

$$\Delta(\mathsf{Game}^{\mathcal{CS}'}_{\mathsf{C}_{t,\lambda,v},\mathsf{A}}; \overline{\mathsf{Game}}^{\mathcal{CS}'}_{\mathsf{C}_{t,\lambda,v},\mathsf{A}}) \leq \exp(-\Omega(\sqrt[3]{(n)}), \tag{7}$$

where $\overline{\mathsf{Game}}^{\mathcal{CS}'}_{\mathsf{C}_{t,\lambda,v},\mathsf{A}}$ is the game where A runs in the experiment as defined by B. To complete the proof we need to lower bound the advantage of B when running against challenger C. We discuss how to handle unpredictability and indistinguishability games separately.

1. *Unpredictability security notion:* For unpredictability games B aborts in Step 4 if $j \neq j^*$. Hence, we get:

$$\begin{aligned}
\Pr[\mathsf{Game}^{\mathcal{CS}}_{\mathsf{C},\mathsf{B}}(1^k) = 1] &= \Pr[\mathsf{Game}^{\mathcal{CS}}_{\mathsf{C},\mathsf{B}}(1^k) = 1 | j = j^*] \Pr[j = j^*] \\
&\quad + \Pr[\mathsf{Game}^{\mathcal{CS}}_{\mathsf{C},\mathsf{B}}(1^k) = 1 | j \neq j^*] \Pr[j \neq j^*] \\
&\geq \Pr[\mathsf{Game}^{\mathcal{CS}}_{\mathsf{C},\mathsf{B}}(1^k) = 1 | j = j^*] \Pr[j = j^*] \\
&\geq \left(\Pr[\mathsf{Game}^{\mathcal{CS}'}_{\mathsf{C}_{t,\lambda,v},\mathsf{A}}(1^k) = 1] - \exp(-\Omega(\sqrt[3]{n})\right) \frac{1}{v} \qquad (8) \\
&> \frac{1}{p'(k)}. \qquad (9)
\end{aligned}$$

(8) follows from Eq. (7) and the fact that conditioned on $j = j^*$ B wins against challenger C when A wins against challenger $\mathsf{C}_{t,\lambda,v}$. Recall that if A wins then the underlying oracle C running with the original key has to output 1. Finally, (9) holds because $v = O(n^d)$ (for some constant d) and by assumption $\Pr[\mathsf{Game}^{\mathcal{CS}'}_{\mathsf{C}_{t,\lambda,v},\mathsf{A}}(1^k) = 1] \geq 1/p(k)$. Clearly, (9) yields a contradiction.

2. *Indistinguishability security notion:* For indistinguishability games B aborts and sends a random bit b to the challenger. As above we need to lower bound the advantage of B when playing against the challenge oracle C.

$$\begin{aligned}
\Pr[\mathsf{Game}^{\mathcal{CS}}_{\mathsf{C},\mathsf{B}}(1^k) = 1] - \frac{1}{2} &= \Pr[\mathsf{Game}^{\mathcal{CS}}_{\mathsf{C},\mathsf{B}}(1^k) = 1 | j = j^*] \Pr[j = j^*] \\
&\quad + \Pr[\mathsf{Game}^{\mathcal{CS}}_{\mathsf{C},\mathsf{B}}(1^k) = 1 | j \neq j^*] \Pr[j \neq j^*] - \frac{1}{2} \\
&\geq \Pr[\mathsf{Game}^{\mathcal{CS}}_{\mathsf{C},\mathsf{B}}(1^k) = 1 | j = j^*] \Pr[j = j^*] + \frac{\Pr[j \neq j^*]}{2} - \frac{1}{2} \qquad (10) \\
&\geq \frac{\left(\Pr[\mathsf{Game}^{\mathcal{CS}'}_{\mathsf{C}_{t,\lambda,v},\mathsf{A}}(1^k) = 1] - \exp(-\Omega(\sqrt[3]{n}))\right)}{v} + \frac{v-1}{2v} - \frac{1}{2} \qquad (11) \\
&> \frac{1}{p'(k)}. \qquad (12)
\end{aligned}$$

(10) holds because $\Pr[\mathsf{Game}^{\mathcal{CS}}_{\mathsf{C},\mathsf{B}}(1^k) = 1 | j \neq j^*] = 1/2$. (11) follows from Eq. (7) and the fact that conditioned on $j = j^*$ B wins against challenger C when A wins against challenger $\mathsf{C}_{t,\lambda,v}$. Finally, (12) holds because $v = O(n^d)$ (for some constant d) and $\Pr[\mathsf{Game}^{\mathcal{CS}'}_{\mathsf{C}_{t,\lambda,v},\mathsf{A}}(1^k) = 1] \geq 1/2 + 1/p(k)$ for some polynomial $p(.)$.

The above yields a contradiction as for both game types the adversary B has a non-negligible advantage against the underlying challenger C. Hence, we get

$$\Pr[\mathsf{Game}^{\mathcal{CS}'}_{\mathsf{C}_{t,\lambda,v},\mathsf{A}}(1^k) = 1] \leq \mathit{negl}(k)$$

as claimed in the theorem. This concludes the proof. □

3.6 Extensions

We discuss some extensions of the main result from this section.

Beyond semi-adaptivity. Notice that since $\mathcal{T}$ is a set of tuples of functions, Definition 3.2 clearly implies non-adaptive security where the adversary commits to a single chain of tampering functions $(T_1, \ldots, T_t)$. We further notice that we can obtain a stronger form of semi-adaptivity by paying a higher price in the security loss. In this model, after committing to a set of functions $\mathcal{T} = \{T_i : \mathcal{X} \to \mathcal{X}\}$ (in Step 1 of Figure 1), the adversary can adaptively choose individual functions from $\mathcal{T}$ (in Step 4 of Figure 1). The loss in security however increases by a factor v^t (instead of just v as in Theorem 3.1). Finally, observe that we can replace the $2(t+1)$-wise independent hash function with any cryptographic hash function and model it in the security proof of Theorem 3.1 as a random oracle. As long as the tampering function cannot query the random oracle, the tampering choice may now be fully adaptively. Notice also that the random oracle allows us to improve some of the parameters from the theorem – in particular, the compression rate ℓ.

In Section 4 and Section 5 we show that fully adaptive BLT security can be achieved for free if we consider specific constructions. It is an interesting question if also for general primitives stronger adaptivity security notions in the BLT model can be obtained. The following simple example shows, however, that this question may be hard — at least in its most general form.

Example 3.1. Consider a PRF $\psi(K, M)$ that is a function of a d-key K and input M and is secure in the standard sense (without tampering or leakage). We also assume that the function can be broken if one learns a constant fraction of the key bits. We turn this into a new scheme with a public parameter $x_1, \ldots, x_d$ chosen from a large finite field of characteristic 2. The secret key is now a random polynomial f of degree at most $d-1$. To evaluate the function on input M, we first compute $K = (lsb(f(x_1)), \ldots, lsb(f(x_d)))$ where lsb denotes the least significant bit, and output $\psi(K, M)$. If there is no tampering, this is still secure, since K is random, even given the x_i's.

However, a fully adaptive tampering function that has full information on the x_i's can interpolate a polynomial that takes any set of desired values in the x_i's. It can therefore tamper freely with individual bits of K, and use a generic attack to learn t bits of K using t tampering queries and break the function.

On the other hand, a non-adaptive tampering function is not allowed to depend on the public parameters. Assume it replaces the polynomial by $f' \neq f$. Then if $f - f'$ is constant, either K is not changed or all bits of K are flipped. We can reasonably assume that ψ is secure against such a related-key attack. If $f - f'$ is not constant, then $(f - f')(x_i)$ is close to uniform for all i because the degree of $f - f'$ is at most d and this is much smaller than the size of the field. Although the values $(f - f')(x_i)$ are not independent, it is certainly not possible to change only one or a small number of key bits. So assuming ψ has some form of related-key security, non-adaptive tampering cannot break the function.

Avoid hashing by assuming RKA security. We discuss a simple extension of our result from Theorem 3.1 which allows to lift the statement to a fully adaptive setting, in case one is willing to assume the underlying cryptographic scheme has an additional security property (essentially a form of related-key attack security [4]). The scheme $\mathcal{CS}$ should remain secure even against an adversary which is allowed to see outputs Z' produced with keys related to X but that still retain high enough min-entropy. In this case, we can avoid entirely the transformation based on hashing and apply directly this assumption in the proof of Lemma 3.3.

One natural question to ask is whether one can hope to prove that all primitives are secure in the non-adaptive BLT model, without necessarily using our transformation. The question to this answer is negative. Consider for instance the Naor-Reingold construction of a PRF [30]. For a group

$\mathbb{G}$ of prime order p with generator g, let $\mathrm{NR} : (\mathbb{Z}_p^*)^{n+1} \times \{0,1\}^n \to \mathbb{G}$ be defined as $\mathrm{NR}(\mathbf{x}, m) = g^{x_0 \cdot \prod_{i=1}^n x_i^{m_i}}$. The following is a simple non-adaptive attack on NR. Before the public parameters are sampled, commit to tampering function T, such that $T(x_0, \ldots, x_n) = (x_0, x_2, x_1, x_3 \ldots, x_n)$ (i.e., T just swap x_1 and x_2). Query the function on input $m' = (1, 0, \ldots, 0)$; this yields the value $y' = g^{x_0 \cdot x_2}$. Now, run the challenge phase using input $m'' = (0, 1, 0, \ldots, 0)$. This is clearly distinguishable from random, as $y'' = y'$ for NR.

4 ID Schemes with BLT Security

In an identification scheme a prover tries to convince a verifier of its identity (corresponding to its public key pk). Formally, an identification scheme is a tuple of algorithms $\mathcal{ID} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{P}, \mathsf{V})$ defined as follows:

$pp \leftarrow \mathsf{Setup}(1^k)$: Algorithm Setup takes the security parameter as input and outputs public parameters pp. The set of all public parameter is denoted by $\mathcal{PP}$.

$(pk, sk) \leftarrow \mathsf{Gen}(1^k)$: Algorithm Gen outputs the public key and the secret key corresponding to the prover's identity. The set of all possible secret keys is denoted by $\mathcal{SK}$.

(P, V): We let $(\mathsf{P}(pp, sk) \rightleftarrows \mathsf{V}(pp))(pk)$ denote the interaction between prover P (holding sk and using public parameters pp) and verifier V on common input pk. Such interaction outputs a result in $\{accept, reject\}$, where *accept* means P's identity is considered as valid.

Definition 4.1. Let $\lambda = \lambda(k)$, $t = t(k)$ and $\delta = \delta(k)$ be parameters and let $\mathcal{T}$ be some set of functions such that $T \in \mathcal{T}$ has a type $T : \mathcal{SK} \times \mathcal{PP} \to \mathcal{SK} \times \mathcal{PP}$. We say that $\mathcal{ID}$ is (λ, t, δ)-bounded leakage and tamper secure (in short BLT-secure) against impersonation attacks with respect to $\mathcal{T}$ if the following properties are satisfied.

(i) *Correctness.* For all $pp \leftarrow \mathsf{Setup}(1^k)$ and $(pk, sk) \leftarrow \mathsf{Gen}(1^k)$ we have that $(\mathsf{P}(pp, sk) \rightleftarrows \mathsf{V}(pp))(pk)$ outputs *accept*.

(ii) *Security.* For all PPT adversaries A we have that $\Pr[\mathsf{A} \text{ wins}] \leq \delta(k)$ in the following game:

1. The challenger runs $pp \leftarrow \mathsf{Setup}(1^k)$ and $(pk, sk) \leftarrow \mathsf{Gen}(1^k)$, and gives (pp, pk) to A.
2. The adversary is given oracle access to $\mathsf{P}(pp, sk)$, modeled as an oracle that outputs polynomially many proof transcripts with respect to secret key sk.
3. The adversary may adaptively ask t tampering queries. During the ith query, A chooses a function $T_i \in \mathcal{T}$ and gets oracle access to $\mathsf{P}(\widetilde{pp}_i, \widetilde{sk}_i)$, where $(\widetilde{sk}_i, \widetilde{pp}_i) = T_i(\widetilde{sk}_{i-1}, \widetilde{pp}_{i-1})$ and $(\widetilde{sk}_0, \widetilde{pp}_0) = (sk, pp)$. The adversary can interact with the oracle $\mathsf{P}(\widetilde{pp}_i, \widetilde{sk}_i)$ a polynomially number of times, where it uses the tampered secret key $\widetilde{sk}_i$ and the public parameter $\widetilde{pp}_i$.
4. The adversary may adaptively ask leakage queries. In the jth query, A chooses a function $L_j : \{0,1\}^* \to \{0,1\}^{\lambda_j}$ and receives back the output of the function applied to sk.
5. The adversary loses access to all other oracles and interacts with an honest verifier V (holding pk). We say that A *wins* if $(\mathsf{A} \rightleftarrows \mathsf{V}(pp))(pk)$ outputs *accept* and $\sum_j \lambda_j \leq \lambda$.

Notice that in the above definition the leakage is from the original secret key sk. This is without loss of generality as our tampering functions are modeled as deterministic circuits.

In a slightly more general setting, one could allow A to leak on the original secret key also in the last phase where it has to convince the verifier. In the terminology of [2] this is reminiscent of so-called *anytime leakage* attacks. Our results can be generalized with respect to this more general definition, however we stick to Definition 4.1 for simplicity.

The rest of this section is organized as follows. In Section 4.1 we prove that a large class of Σ-protocols are secure in the BLT model, where the tampering function is allowed to modify the secret state of the prover but not the public parameters. In Section 4.2 we look at a concrete instantiation based on the Okamoto ID scheme, and prove that this construction is secure in a stronger model where the tampering function can modify both the secret state of the prover and the public parameters (but independently). Finally, in Section 4.3 we illustrate that the latter assumption is necessary, as otherwise the Okamoto ID scheme can be broken by (albeit contrived) attacks.

4.1 Σ-protocols are Tamper Resilient

We start by considering ID schemes based on Σ-protocols [9]. Σ-protocols are a special class of interactive proof systems for membership in a language $\mathfrak{L}$, where a prover $\mathsf{P} = (\mathsf{P}_0, \mathsf{P}_1)$ wants to convince a verifier $\mathsf{V} = (\mathsf{V}_0, \mathsf{V}_1)$ (both modelled as PPT algorithms) that a shared string y belongs to $\mathfrak{L}$. Denote with x the witness corresponding to y and let pp be public parameters. The protocol proceeds as follows: (1) The prover computes $a \leftarrow \mathsf{P}_0(pp)$ and sends it to the verifier; (2) The verifier chooses $c \leftarrow \mathsf{V}_0(pp, y)$ uniformly at random and sends it to the prover; (3) The prover answers with $z \leftarrow \mathsf{P}_1(pp, (a, c, x))$; (4) The verifier outputs a result $\mathsf{V}_1(pp, y, (a, c, z)) \in \{accept, reject\}$. We call this a *public-coin* three round interactive proof system. A formal definition of Σ-protocols follows.

Definition 4.2 (Σ-protocol)**.** A Σ-protocol (P, V) for a relation $\mathfrak{R}$ is a three round public-coin interactive proof system with the following properties.

Completeness. Whenever P and V follow the protocol on common input y, public parameters pp and private input x to P such that $(y, x) \in \mathfrak{R}$, the verifier V accepts with all but negligible probability.

Special soundness. From any pair of accepting conversations on public input y, namely (a, c, z), (a, c', z') such that $c \neq c'$, one can efficiently compute x such that $(y, x) \in \mathfrak{R}$.

Perfect Honest Verifier Zero Knowledge (HVZK). There exists a PPT simulator M, which on input y and a random c outputs an accepting conversation of the form (a, c, z), with exactly the same probability distribution as conversations between the honest P, V on input y.

Note that Definition 4.2 requires *perfect* HVZK, whereas in general one could ask for a weaker requirement, namely that the HVZK property holds only computationally.

It is well known that Σ-protocols are a natural tool to design ID schemes. The construction is depicted in Figure 3. Consider now the class of tampering functions $\mathcal{T}_{\mathsf{sk}} \subset \mathcal{T}$ such that $T \in \mathcal{T}_{\mathsf{sk}}$ has the following form: $T = (T^{sk}, ID^{pp})$ where $T^{sk} : \mathcal{SK} \to \mathcal{SK}$ is an arbitrary polynomial time computable function and $ID^{pp} : \mathcal{PP} \to \mathcal{PP}$ is the identity function. This models tampering with the secret state of P without changing the public parameters (these must be hard-wired into the prover's code). We prove the following:

ID Scheme from Σ-Protocol

Let $((\mathsf{P}_0, \mathsf{P}_1), (\mathsf{V}_0, \mathsf{V}_1))$ be a Σ-protocol for a relation $\mathfrak{R}$.

$\mathsf{Setup}(1^k)$**:** Sample public parameters $pp \leftarrow \mathcal{PP}$ for the underlying relation $\mathfrak{R}$.

$\mathsf{Gen}(1^k)$**:** Output a pair $(y, x) \in \mathfrak{R}$, where $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ and $|x|$ is polynomially bounded by $|y|$.

$(\mathsf{P}(pp, x) \rightleftarrows \mathsf{V}(pp))(y)$**:** The protocol works as follows.

1. The prover sends $a \leftarrow \mathsf{P}_0(pp)$ to the verifier.
2. The verifier chooses a random challenge $c \leftarrow \mathsf{V}_0(pp, y)$ and sends it to the prover.
3. The prover computes the answer $z \leftarrow \mathsf{P}_1(pp, (a, c, x))$.
4. The verifier accepts iff $\mathsf{V}_1(pp, y, (a, c, z))$ outputs *accept*.

Figure 3: ID scheme based on Σ-protocol for relation $\mathfrak{R}$

Theorem 4.1. *Let $k \in \mathbb{N}$ be the security parameter and let (P, V) be a Σ-protocol for relation $\mathfrak{R}$ with $|\mathcal{Y}| = O(k^{\log k})$, such that the representation problem is hard for $\mathfrak{R}$ (cf. Section 2). Assume that conditioned on the distribution of the public input $y \in \mathcal{Y}$, the witness $x \in \mathcal{X}$ has high average min entropy β, i.e., $\widetilde{\mathbf{H}}_\infty(X|Y) \geq \beta$. Then, the ID scheme of Figure 3 is $(\lambda(k), t(k), negl(k))$-BLT secure against impersonation attacks with respect to $\mathcal{T}_{\mathsf{sk}}$, where*

$$\lambda \leq \beta - t \log |\mathcal{Y}| - k \qquad \textit{and} \qquad t \leq \left\lfloor \frac{\beta}{\log |\mathcal{Y}|} \right\rfloor - 1.$$

Proof. Assume that there exists a polynomial $p(.)$ and an adversary A that succeeds in the BLT experiment (cf. Definition 4.1) with probability at least $\delta(k) := 1/p(k)$, for infinitely many $k \in \mathbb{N}$. Then, we construct an adversary B (using A as a subroutine) such that:

$$\Pr\left[x^\star \neq x; (y, x), (y, x^\star) \in \mathfrak{R} : \ (y, x, x^\star) \leftarrow \mathsf{B}(pp); pp \leftarrow \mathsf{Setup}(1^k)\right] \geq \delta^2 - |\mathcal{Y}|^{-1} - 2^{-k}.$$

Since $|\mathcal{Y}|$ is super-polynomial in k, this contradicts the assumption that the representation problem is hard for $\mathfrak{R}$ (cf. Section 2).

Adversary B works as follows. It first samples $(y, x) \leftarrow \mathsf{Gen}(1^k)$, then it uses these values to simulate the entire experiment for A. This includes answers to the leakage queries, and access to the oracles $\mathsf{P}(pp, \widetilde{x}_i)$ for all $i \in [t]$. During the impersonation stage, B chooses a random challenge c which results in a transcript (a, c, z). At this point B rewinds A to the point after it chose a, and selects a different challenge c' resulting in a transcript (a, c', z'). Whenever the two transcripts are accepting and $c' \neq c$, the special soundness property ensures that adversary B has extracted successfully some value $x^\star$ such that $(y, x^\star) \in \mathfrak{R}$. Let us call the event described above E_1. And the event $x = x^\star$ is denoted by E_2. Clearly,

$$\Pr\left[\mathsf{B} \text{ succeeds}\right] = \Pr\left[x^\star \neq x; (y, x), (y, x^\star) \in \mathfrak{R} : \ (y, x, x^\star) \leftarrow \mathsf{B}(pp); pp \leftarrow \mathsf{Setup}(1^k)\right] = \Pr\left[E_1 \wedge \neg E_2\right]. \tag{13}$$

Claim 2. *The probability of event E_1 is $\Pr[E_1] \geq \delta^2 - |\mathcal{Y}|^{-1}$.*

Proof. The proof is identical to the proof of [2, Claim 4.1] and is therefore omitted. □

Claim 3. *The probability of event E_2 is* $\Pr[E_2] \leq 2^{-k}$.

Proof. We prove the claim holds even in case the adversary is unbounded. Consider an experiment $\mathcal{E}_0$ which is similar to the experiment of Definition 4.1, except that now the adversary does not get access to the leakage oracle. Consider an adversary A trying to predict the value of x given the view in a run of $\mathcal{E}_0$; such view contains polynomially many transcripts (for the initial secret key and for each of the tampering queries) together with the original public input y and the public parameter pp (which are tamper-free), i.e., $view_{\mathsf{A}}^{\mathcal{E}_0} = \{\mathbf{\Psi}, \mathbf{\Psi}_1, \ldots, \mathbf{\Psi}_t\} \cup \{y, pp\}$. The vector $\mathbf{\Psi}$ and each of the vectors $\mathbf{\Psi}_i$ contains polynomially many transcripts of the form (a, c, z), corresponding (respectively) to the original key and to the ith tampering query.

We now move to experiment $\mathcal{E}_1$, which is the same as $\mathcal{E}_0$ with the modification that we add (for each tampering query) the tampered public values $\widetilde{y}_i$ to A's view. Hence, $view_{\mathsf{A}}^{\mathcal{E}_1} = view_{\mathsf{A}}^{\mathcal{E}_0} \cup \{(\widetilde{y}_1, \ldots, \widetilde{y}_t)\}$. Note that we have

$$\widetilde{\mathbf{H}}_\infty(X|\mathcal{E}_0) \geq \widetilde{\mathbf{H}}_\infty(X|\mathcal{E}_1). \tag{14}$$

Next, we consider experiment $\mathcal{E}_2$ where A is given only the tampered public values $(\widetilde{y}_1, \ldots, \widetilde{y}_t)$, i.e., $view_{\mathsf{A}}^{\mathcal{E}_2} = \{\widetilde{y}_1 \ldots, \widetilde{y}_t\} \cup \{y, pp\}$. We claim that conditioning on $\mathcal{E}_1$ or on $\mathcal{E}_2$ has the same effect on the min-entropy of X. This is because the values $\{\mathbf{\Psi}, \mathbf{\Psi}_i\}_{i \in [t]}$ can be computed as a deterministic function of $(y, \widetilde{y}_1, \ldots, \widetilde{y}_t)$ as follows: For all challenges c run the HVZK simulator M upon input $(pp, \widetilde{y}_i, c)$ and append the output (a, c, z) to $\mathbf{\Psi}_i$. (The same can be done to simulate $\mathbf{\Psi}$ using y.) It follows from perfect HVZK that this generates an identical distribution to that of experiment $\mathcal{E}_1$ and thus

$$\widetilde{\mathbf{H}}_\infty(X|\mathcal{E}_1) = \widetilde{\mathbf{H}}_\infty(X|\mathcal{E}_2). \tag{15}$$

Since the public parameters are tamper-free and are chosen independently of X, we can remove them from the view and write

$$\widetilde{\mathbf{H}}_\infty(X|\mathcal{E}_2) = \widetilde{\mathbf{H}}_\infty(X|\widetilde{Y}_1, \ldots, \widetilde{Y}_t, Y) \geq \widetilde{\mathbf{H}}_\infty(X|Y) - |(\widetilde{Y}_1, \ldots, \widetilde{Y}_t)| \geq \beta - t \log |\mathcal{Y}|, \tag{16}$$

where we used Lemma 2.1 together with the fact that the joint distribution $(\widetilde{Y}_1, \ldots, \widetilde{Y}_t)$ can take at most $(|\mathcal{Y}|)^t$ values and our assumption on the conditional min-entropy of X given Y.

Consider now the full experiment described in Definition 4.1 and call it $\mathcal{E}_3$. Note that this experiment is similar to the experiment $\mathcal{E}_0$, with the only addition that here A has also access to the leakage oracle. Hence, we have $view_{\mathsf{A}}^{\mathcal{E}_3} = view_{\mathsf{A}}^{\mathcal{E}_0} \cup view_{\mathsf{A}}^{\mathsf{leak}}$. Denote with $\Lambda \in \{0,1\}^\lambda$ the random variable corresponding to $view_{\mathsf{A}}^{\mathsf{leak}}$. Using Lemma 2.1 and combining Eq. (14)–(16) we get

$$\widetilde{\mathbf{H}}_\infty(X|\mathcal{E}_3) = \widetilde{\mathbf{H}}_\infty(X|\mathcal{E}_0, \Lambda) \geq \widetilde{\mathbf{H}}_\infty(X|\mathcal{E}_0) - \lambda \geq \beta - t \log |\mathcal{Y}| - \lambda \geq k,$$

where the last inequality comes from the value of λ in the theorem statement. We can thus bound the probability of E_2 as $\Pr[E_2] \leq 2^{-\widetilde{\mathbf{H}}_\infty(X|\mathcal{E}_3)} \leq 2^{-k}$. The claim follows. □

Combining Claim 2 and Claim 3 together with Eq. (13) yields

$$\Pr[\mathsf{B} \text{ succeeds}] = \Pr[E_1 \wedge \neg E_2] \geq \Pr[E_1] - \Pr[E_2] \geq \delta^2 - |\mathcal{Y}|^{-1} - 2^{-k},$$

which contradicts our assumption on the hardness of the representation problem for $\mathfrak{R}$. This finishes the proof. □

Generalized Okamoto ID Scheme

Let $\ell = \ell(k)$ be some function of the security parameter. Consider the following identification scheme.

Setup: Choose a group $\mathbb{G}$ of prime order p with generator g and a vector $\boldsymbol{\alpha} \leftarrow \mathbb{Z}_p^\ell$, and output $pp = (\mathbb{G}, g, g^{\boldsymbol{\alpha}})$ where $g^{\boldsymbol{\alpha}} = (g_1, \ldots, g_\ell)$.

$\mathsf{Gen}(1^k)$: Select a vector $\mathbf{x} \leftarrow \mathbb{Z}_p^\ell$ and set $y = pk = \prod_{i=1}^\ell g_i^{x_i}$ and $sk = \mathbf{x}$.

$(\mathsf{P}(pp, sk) \rightleftarrows \mathsf{V}(pp))(pk)$: The protocol works as follows.

1. The prover chooses a random vector $\mathbf{r} \leftarrow \mathbb{Z}_p^\ell$ and sends $a = \prod_{i=1}^\ell g_i^{r_i}$ to the verifier.
2. The verifier chooses a random challenge $c \leftarrow \mathbb{Z}_p$ and sends it to the prover.
3. The prover computes the answer $\mathbf{z} = (r_1 + cx_1, \ldots, r_\ell + cx_\ell)$.
4. The verifier accepts if and only if $\prod_{i=1}^\ell g_i^{z_i} = a \cdot y^c$.

Figure 4: Generalized Okamoto Identification Scheme

4.2 Concrete Instantiation with more Tampering

We extend the power of the adversary by allowing him to tamper not only with the witness, but also with the public parameters (used by the prover to generate the transcripts). However the tampering has to be independent on the two components. This is reminiscent of the so-called split-state model (considered for instance in [29]), with the key difference that in our case the secret state does not need to be split into two parts.

We model this trough the following class of tampering functions $\mathcal{T}_{\mathsf{split}}$: We say that $T \in \mathcal{T}_{\mathsf{split}}$ if we can write $T = (T^{sk}, T^{pp})$ where $T^{sk} : \mathcal{SK} \to \mathcal{SK}$ and $T^{pp} : \mathcal{PP} \to \mathcal{PP}$ are arbitrary polynomial time computable functions. Recall that the input/output domains of T^{sk}, T^{pp} are identical, hence the size of the witness and the public parameters cannot be changed. As we show in the next section, this restriction is necessary. Note also that $\mathcal{T}_{\mathsf{sk}} \subseteq \mathcal{T}_{\mathsf{split}} \subseteq \mathcal{T}$.

Generalized Okamoto. Consider the generalized version of the Okamoto ID scheme [32], depicted in Figure 4. The underlying hard relation here is the relation $\mathfrak{R}_{\mathsf{DL}}$ and the representation problem for $\mathfrak{R}_{\mathsf{DL}}$ is the ℓ-representation problem in a group $\mathbb{G}$ (cf. Section 2). As proven in [2], this problem is equivalent to the Discrete Log problem in $\mathbb{G}$.

Corollary 2. *Let $k \in \mathbb{N}$ be the security parameter and assume the Discrete Log problem is hard in $\mathbb{G}$. Then, the generalized Okamoto ID scheme is $(\lambda(k), t(k), negl(k))$-BLT secure against impersonation attacks with respect to $\mathcal{T}_{\mathsf{split}}$, where*

$$\lambda \leq (\ell - 1 - t)\log(p) - k \qquad \text{and} \qquad t \leq \ell - 2.$$

Proof. We first show that the protocol is BLT-secure against impersonation attacks with respect to $\mathcal{T}_{\mathsf{sk}}$. This follows immediately from Theorem 4.1 as the protocol of Figure 4 is a Σ-protocol which satisfies perfect HVZK; moreover $|\mathcal{Y}| = p$ and the size of prime p is super-polynomial in k to ensure hardness of the Discrete Log problem. The claimed values of λ and t follow by observing that the secret key $\mathbf{x}$ conditioned on the public key y is uniform in a subspace of dimension $\ell - 1$, i.e., $\mathbf{H}_\infty(X|Y) \geq (\ell - 1)\log p = \beta$.

We now turn to prove security with respect to $\mathcal{T}_{\mathsf{split}}$; note that here $\mathcal{PP} = (p, g_1, \ldots, g_\ell)$. To do so, we modify the view of the adversary in the proof of Theorem 4.1 such that it contains also

the tampered public parameters $\widetilde{pp}_i$ for all $i \in [t]$. In particular, the elements $(a, c, \mathbf{z})$ contained in each vector $\mathbf{\Psi}_i$ in the view of experiment $\mathcal{E}_0$ are now sampled from $\mathsf{P}(\widetilde{pp}_i, \widetilde{\mathbf{x}}_i)$. We then modify $\mathcal{E}_1$ and $\mathcal{E}_2$ by appending the values of the tampered public parameters $\{\widetilde{pp}_i\}_{i\in[t]}$.

We claim that also in this case $\widetilde{\mathbf{H}}_\infty(X|\mathcal{E}_1) = \widetilde{\mathbf{H}}_\infty(X|\mathcal{E}_2)$, in particular the view of A in $\mathcal{E}_1$ can be simulated given only $\{\widetilde{pk}_i, \widetilde{pp}_i\}_{i\in[t]}$. This follows from the fact that the generalized Okamoto ID scheme maintains the completeness and perfect HVZK properties even when the transcripts are computed using tampered public parameters $\widetilde{pp}_i = (\widetilde{p}, \widetilde{g}_1, \ldots, \widetilde{g}_\ell)$. (Whereas of course in this case the protocol is not sound.) The HVZK simulator $\mathsf{M}(\widetilde{pp}, \widetilde{y}, c)$ works as follows: Choose $z_1, \ldots, z_\ell$ at random in $\mathbb{Z}_{\widetilde{p}}$ and if $\widetilde{y} \neq 0 \mod \widetilde{p}$, then compute $a = (\prod_{i=1}^{\ell} \widetilde{g}_i^{z_i})/\widetilde{y}^c \bmod \widetilde{p}$. In case $\widetilde{y} = 0 \mod \widetilde{p}$, then just set $a = 0$.[6] For any $(\widetilde{\mathbf{x}}, \widetilde{pp}) = (T^{sk}(\mathbf{x}), T^{pp}(pp))$, the distributions $\mathsf{M}(\widetilde{pp}, \widetilde{y}, c)$ and $(\mathsf{P}(\widetilde{pp}, \widetilde{\mathbf{x}}) \rightleftarrows \mathsf{V}(\widetilde{pp}))(\widetilde{y})$ are both uniformly random over all values $(a, c, \mathbf{z} = (z_1, \ldots, z_\ell))$ such that $\prod_{i=1}^{\ell} \widetilde{g}_i^{z_i} = a\widetilde{y}^c \bmod \widetilde{p}$.

Therefore the simulation perfectly matches the honest conversation. This proves Eq. (15). Now Eq. (16) follows from the fact that the tampering functions T^{pp} cannot depend on sk. □

4.3 Some Attacks

We show that for the Okamoto scheme it is hard to hope for BLT security beyond the class of tampering functions $\mathcal{T}_{\mathsf{split}}$. We illustrate this by concrete attacks which work in case one tries to extend the power of the adversary in two different ways: (1) Allowing A to tamper jointly with the witness and the public parameters; (2) Allowing A to tamper independently with the witness and with the public parameters but increase their size.

Tampering jointly with the public parameters. Consider the class of tampering functions $\mathcal{T}$ introduced in Definition 4.1.

Claim 4. *The generalized Okamoto ID scheme is* not *BLT-secure against impersonation attacks with respect to* $\mathcal{T}$.

Proof. We show an attack using a single tampering query. Define the tampering function $T(\mathbf{x}, pp) = (\widetilde{\mathbf{x}}, \widetilde{pp})$ to be as follows:

- The witness is unchanged, i.e., $\mathbf{x} = \widetilde{\mathbf{x}}$.
- The value $\widetilde{p}$ is some prime of size $|\widetilde{p}| \approx |p|$ such that the Discrete Log problem is easy in the corresponding group $\widetilde{\mathbb{G}}$. (This can be done efficiently by choosing $\widetilde{p} - 1$ to be the product of small prime (power) factors [34].)
- Let $\widetilde{g}$ be a generator of $\widetilde{\mathbb{G}}$ (which must exists since $\widetilde{p}$ is a prime) and define the new generators as $\widetilde{g}_i = \widetilde{g}^{x_i} \bmod \widetilde{p}$.

Consider now a transcript $(a, c, \mathbf{z})$ produced by a run of $\mathsf{P}(\widetilde{pp}, \mathbf{x})$. We have $a = \widetilde{g}^{\sum_{i=1}^{\ell} x_i r_i} \bmod \widetilde{p}$ for random $r_i \in \mathbb{Z}_{\widetilde{p}}$. By computing the Discrete Log of a in base $\widetilde{g}$ (which is easy by our choice of $\widetilde{\mathbb{G}}$), we get one equation $\sum_{i=1}^{\ell} x_i r_i = \log_{\widetilde{g}}(a) \bmod \widetilde{p}$. Asking for polynomially many transcripts, yields ℓ linearly independent equations (with overwhelming probability) and thus allows to solve for $(x_1, \ldots, x_\ell)$. (Note here that with high probability $x_i \bmod p = x_i \bmod \widetilde{p}$ since $|p| \approx |\widetilde{p}|$.) □

[6] Note that $\widetilde{y} = 0 \bmod \widetilde{p}$ implies that for at least one of the generators g_i's we get $\widetilde{g}_i = 0 \bmod \widetilde{p}$, so that $a = \prod_{i=1}^{\ell} \widetilde{g}_i^{r_i} = 0 \bmod \widetilde{p}$.

Tampering by "inflating" the prime p. Consider the following class of tampering functions $\mathcal{T}_{\mathsf{split}} \subseteq \mathcal{T}^*_{\mathsf{split}}$: We say that $T \in \mathcal{T}^*_{\mathsf{split}}$ if $T = (T^{sk}, T^{pp})$, where $T^{sk} : \mathcal{SK} \to \{0,1\}^*$ and $T^{pp} : \mathcal{PP} \to \{0,1\}^*$.

Claim 5. *The generalized Okamoto ID scheme is* not *BLT-secure against impersonation attacks with respect to* $\mathcal{T}^*_{\mathsf{split}}$.

Proof. We show an attack using a single tampering query. Consider the following tampering function $T = (T^{sk}, T^{pp}) \in \mathcal{T}^*_{\mathsf{split}}$:

- Choose $\widetilde{p}$ to be a prime of size $|\widetilde{p}| = \Omega(\ell|p|)$, such that the Discrete Log problem is easy in $\widetilde{\mathbb{G}}$. (This can be done as in the proof of Claim 4.)
- Choose a generator $\widetilde{g}$ of $\widetilde{\mathbb{G}}$; define $\widetilde{g}_1 = \widetilde{g}$ and $\widetilde{g}_j = 1$ for all $j = 2, \ldots, \ell$.
- Define the witness to be $\widetilde{\mathbf{x}}$ such that $\widetilde{x}_1 = x_1 || \ldots || x_\ell$ and $\widetilde{x}_j = 0$ for all $j = 2, \ldots, \ell$.

Given a single transcript $(a, c, \mathbf{z})$ the adversary learns $a = \widetilde{g}^{r_1}$ for some $r_1 \in \mathbb{Z}_{\widetilde{p}}$. Since the Discrete Log is easy in this group, A can find r_1. Now the knowledge of c and $z_1 = r_1 + c\widetilde{x}_1$, allows to recover $\widetilde{x}_1 = (x_1, \ldots, x_\ell)$. □

4.4 BLT-Secure Signatures

It is well known that every Σ-protocol can be turned into a signature scheme via the Fiat-Shamir heuristic [20]. By applying the Fiat-Shamir transformation to the protocol of Figure 3, we get efficient BLT-secure signatures in the random oracle model.

5 IND-CCA PKE with BLT Security

We start by defining IND-CCA public key encryption (PKE) with BLT security. A PKE scheme is a tuple of algorithms $\mathcal{PKE} = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ defined as follows. (1) Algorithm Setup takes as input the security parameter and outputs the description of public parameters pp; the set of all public parameters is denoted by $\mathcal{PP}$. (2) Algorithm KGen takes as input the security parameter and outputs a public/secret key pair (pk, sk); the set of all secret keys is denoted by $\mathcal{SK}$ and the set of all public keys by $\mathcal{PK}$. (3) The randomized algorithm Enc takes as input the public key pk, a message $m \in \mathcal{M}$ and randomness $r \in \mathcal{R}$ and outputs a ciphertext $c = \mathsf{Enc}(pk, m; r)$; the set of all ciphertexts is denoted by $\mathcal{C}$. (4) The deterministic algorithm Dec takes as input the secret key sk and a ciphertext $c \in \mathcal{C}$ and outputs $m = \mathsf{Dec}(sk, c)$ which is either equal to some message $m \in \mathcal{M}$ or to an error symbol $\perp$.

Definition 5.1. Let $\lambda = \lambda(k)$, $t = t(k)$ and $\delta = \delta(k)$ be parameters and let $\mathcal{T}_{\mathsf{sk}}$ be some set of functions such that $T \in \mathcal{T}_{\mathsf{sk}}$ has a type $T : \mathcal{SK} \to \mathcal{SK}$. We say that $\mathcal{PKE}$ is IND-CCA $(\lambda(k), t(k), \delta(k))$-BLT secure with respect to $\mathcal{T}_{\mathsf{sk}}$ if the following properties are satisfied.

(i) *Correctness.* For all $pp \leftarrow \mathsf{Setup}(1^k)$, $(pk, sk) \leftarrow \mathsf{KGen}(1^k)$ we have that $\Pr[\mathsf{Dec}(sk, \mathsf{Enc}(pk, m)) = m] = 1$ (where the randomness is taken over the internal coin tosses of algorithm Enc).

(ii) *Security.* For all PPT adversaries A we have that $\Pr[\mathsf{A} \text{ wins}] \leq \frac{1}{2} + \delta(k)$ in the following game:

1. The challenger runs $pp \leftarrow \mathsf{Setup}(1^k)$, $(pk, sk) \leftarrow \mathsf{KGen}(1^k)$ and gives (pp, pk) to A.
2. The adversary is given oracle access to $\mathsf{Dec}(sk, \cdot)$. This oracle outputs polynomially many decryptions of ciphertexts using secret key sk.
3. The adversary may adaptively ask t tampering queries. During the ith query, A chooses a function $T_i \in \mathcal{T}_{\mathsf{sk}}$ and gets oracle access to $\mathsf{Dec}(\widetilde{sk}_i, \cdot)$, where $\widetilde{sk}_i = T_i(\widetilde{sk}_{i-1})$ and $\widetilde{sk}_0 = sk$. This oracle outputs polynomially many decryptions of ciphertexts using secret key $\widetilde{sk}_i$.
4. The adversary may adaptively ask polynomially many leakage queries. In the jth query, A chooses a function $L_j : \{0,1\}^* \to \{0,1\}^{\lambda_j}$ and receives back the output of the function applied to sk.
5. The adversary outputs two messages of the same length $m_0, m_1 \in \mathcal{M}$ and the challenger computes $c_b \leftarrow \mathsf{Enc}(pk, m_b)$ where b is a uniformly random bit.
6. The adversary outputs a bit b' and wins if $b = b'$ and $\sum_j \lambda_j \leq \lambda$.

In case $t = 0$ we get the notion of leakage resilient IND-CCA from [31] as a special case.

We build an IND-CCA BLT-secure PKE scheme in two steps. In Section 5.1 we define a weaker notion which we call IND-CPA BLT security. In Section 5.2 we show a general transformation from IND-CPA BLT security to IND-CCA BLT security relying on tSE NIZK proofs [12] in the common reference string (CRS) model. The CRS is supposed to be tamper-free and must be hard-wired into the code of the encryption algorithm; however tampering and leakage can depend adaptively on the CRS and the public parameters. Finally, in Section 5.3, we prove that a variant of the BHHO encryption scheme [31] satisfies our notion of IND-CPA BLT security.

5.1 IND-CPA BLT Security

The main idea of our new security notion is as follows. Instead of giving A full access to a tampering oracle (as in Definition 5.1) we restrict his power by allowing him to see the output of the (tampered) decryption oracle only for ciphertexts c for which A already knows both the corresponding plaintext m and the randomness r used to generate c (via the real public key). Essentially this restricts A to submit to the tampering oracle only "well-formed" ciphertexts.

Definition 5.2. Let $\lambda = \lambda(k)$, $t = t(k)$ and $\delta = \delta(k)$ be parameters and let $\mathcal{T}_{\mathsf{sk}}$ be some set of functions such that $T \in \mathcal{T}_{\mathsf{sk}}$ has a type $T : \mathcal{SK} \to \mathcal{SK}$. We say that $\mathcal{PKE}$ is IND-CPA $(\lambda(k), t(k), \delta(k))$-BLT secure with respect to $\mathcal{T}_{\mathsf{sk}}$ if it satisfies property (i) of Definition 5.1 and property (ii) is modified as follows:

(ii) *Security.* For all PPT adversaries A we have that $\Pr[\mathsf{A} \text{ wins}] \leq \frac{1}{2} + \delta(k)$ in the following game:

1. The challenger runs $pp \leftarrow \mathsf{Setup}(1^k)$, $(pk, sk) \leftarrow \mathsf{KGen}(1^k)$ and gives (pp, pk) to A.
2. The adversary may adaptively ask t tampering queries. During the ith query, A chooses a function $T_i \in \mathcal{T}_{\mathsf{sk}}$ and gets oracle access to $\mathsf{Dec}^*(\widetilde{sk}_i, \cdot, \cdot)$, where $\widetilde{sk}_i = T_i(\widetilde{sk}_{i-1})$ and $\widetilde{sk}_0 = sk$. This oracle answers polynomially many queries of the following form: Upon input a pair $(m, r) \in \mathcal{M} \times \mathcal{R}$, compute $c \leftarrow \mathsf{Enc}(pk, m; r)$ and output a plaintext $\widetilde{m} = \mathsf{Dec}(\widetilde{sk}_i, c)$ using the current tampered key.

From IND-CPA BLT Security to IND-CCA BLT Security

Let $\mathcal{PKE} = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ be a PKE scheme and consider a tSE NIZK argument system $(\mathsf{Gen}, \mathsf{Prove}, \mathsf{Verify})$ for the following relation:

$$\mathfrak{R}_{\mathsf{PKE}} = \{(pk, c), (m, r) : \ c = \mathsf{Enc}(pk, m; r)\}\,.$$

Define the following PKE scheme $\mathcal{PKE}' = (\mathsf{Setup}', \mathsf{KGen}', \mathsf{Enc}', \mathsf{Dec}')$.

Setup': Sample $pp \leftarrow \mathsf{Setup}(1^k)$ and $(\omega, \mathsf{tk}, \mathsf{ek}) \leftarrow \mathsf{Gen}(1^k)$ and let $pp' = (pp, \omega)$.

KGen': Run $(pk, sk) \leftarrow \mathsf{KGen}(1^k)$ and set $pk' = pk$ and $sk' = sk$.

Enc': Sample $r \leftarrow \mathcal{R}$ and compute $c \leftarrow \mathsf{Enc}(pk, m; r)$. Output (c, π), where $\pi \leftarrow \mathsf{Prove}^{\omega}((pk, c), (m, r))$.

Dec': Check that $\mathsf{Verify}^{\omega}((pk, c), \pi) = 1$. If not output $\perp$; otherwise, output $m = \mathsf{Dec}(sk, c)$.

Figure 5: How to transform IND-CPA BLT-secure PKE into IND-CCA BLT-secure PKE

3. The adversary may adaptively ask leakage queries. In the jth query, A chooses a function $L_j : \{0,1\}^* \to \{0,1\}^{\lambda_j}$ and receives back the output of the function applied to sk.
4. The adversary outputs two messages of the same length $m_0, m_1 \in \mathcal{M}$ and the challenger computes $c_b \leftarrow \mathsf{Enc}(pk, m_b)$ where b is a uniformly random bit.
5. The adversary outputs a bit b' and wins if $b = b'$ and $\sum_j \lambda_j \leq \lambda$.

We stress that A loses access to the decryption oracle $\mathsf{Dec}(\widetilde{sk}, \cdot)$.

5.2 A General Transformation

We compile an arbitrary IND-CPA BLT-secure encryption scheme into an IND-CCA BLT-secure one by appending to the ciphertext c an argument of "plaintext knowledge" π computed through a tSE NIZK argument system (cf. Section 2). The same construction has been already used by Dodis *et al.* [12] to go from IND-CPA security to IND-CCA security in the context of memory leakage.

The intuition why the transformation works is fairly simple: The argument π enforces the adversary to submit to the tampered decryption oracle only ciphertexts for which he knows the corresponding plaintext (and the randomness used to encrypt it). In the security proof the pair (m, r) can indeed be extracted from such argument, allowing to reduce IND-CCA BLT security to IND-CPA BLT security.

Theorem 5.1. *Let $k \in \mathbb{N}$ be the security parameter. Assume that $\mathcal{PKE}$ is an IND-CPA $(\lambda(k), t(k), \delta(k))$-BLT secure encryption scheme and that $(\mathsf{Gen}, \mathsf{Prove}, \mathsf{Verify})$ is a strong tSE NIZK argument system for relation $\mathfrak{R}_{\mathsf{PKE}}$. Then, the encryption scheme $\mathcal{PKE}'$ of Figure 5 is IND-CCA $(\lambda(k), t(k), \delta'(k))$-BLT secure for $\delta' \leq \delta + negl(k)$.*

Proof. We prove the theorem by a series of games. All games are a variant of the IND-CCA BLT game and in all games the adversary gets correctly generated public parameters (pp, ω, pk). Leakage and tampering queries are answered using the corresponding secret key sk. The games will differ only in the way the challenge ciphertext is computed or in the way the decryption oracles work.

Game G_1. This is the IND-CCA BLT game of Definition 5.1 for the scheme $\mathcal{PKE}'$. Note in particular that all decryption oracles expect to receive as input a ciphertext of the form (c, π)

and proceed to verify the proof π before decrypting the ciphertext (and output $\perp$ if such verification fails). The challenge ciphertext is a pair (c_b, π_b) such that $c_b = \mathsf{Enc}(pk, m_b; r)$ and $\pi_b \leftarrow \mathsf{Prove}^{\omega}((pk, c_b), (m_b, r))$, where $m_b \in \{m_0, m_1\}$ for a uniformly random bit b. By assumption we have that

$$\Pr[\mathsf{A} \text{ wins in } \mathsf{G}_1] \leq \frac{1}{2} + \delta'(k).$$

Game G_2. In this game we change the way the challenge ciphertext is computed by replacing the argument π_b with a simulated argument $\pi_b \leftarrow \mathsf{S}((pk, c_b), \mathsf{tk})$. It follows from the composable NIZK property of the argument system that G_1 and G_2 are computationally close. In particular there exists a negligible function $\delta_1(k)$ such that $|\Pr[\mathsf{A} \text{ wins in } \mathsf{G}_1] - \Pr[\mathsf{A} \text{ wins in } \mathsf{G}_2]| \leq \delta_1(k)$.

Game G_3. In this game we change the way decryption queries are handled. Queries (c, π) to $\mathsf{Dec}(sk, \cdot)$ (such that π accepts) are answered by running the extractor Ext on π, yielding $(m, r) \leftarrow \mathsf{Ext}((pk, c), \pi, \mathsf{ek})$, and returning m.

Queries (c, π) to $\mathsf{Dec}(\widetilde{sk}_i, \cdot)$ (such that π accepts) are answered as follows. We first extract $(m, r) \leftarrow \mathsf{Ext}((pk, c), \pi, \mathsf{ek})$ as above. Then, instead of returning m, we recompute $c = \mathsf{Enc}(pk, m; r)$ and return $\widetilde{m} = \mathsf{Dec}(\widetilde{sk}_i, c)$.

It follows from true simulation extractability that G_2 and G_3 are computationally close. The reason for this is that A gets to see only a single simulated proof for a true statement (i.e., the pair (pk, c_b)) and thus cannot produce a pair $(c, \pi) \neq (c_b, \pi_b)$ such that the proof π accepts and Ext fails to extract the corresponding plaintext m. In particular there exists a negligible function $\delta_2(k)$ such that $|\Pr[\mathsf{A} \text{ wins in } \mathsf{G}_2] - \Pr[\mathsf{A} \text{ wins in } \mathsf{G}_3]| \leq \delta_2(k)$.

Game G_4. In the last game we replace the ciphertext c_b in the challenge with an encryption of $0^{|m_b|}$, whereas we still compute the proof as $\pi_b \leftarrow \mathsf{S}((pk, c_b), \mathsf{tk})$.

We claim that G_3 and G_4 are computationally close. This follows from IND-CPA BLT-security of $\mathcal{PKE}$. Assume there exists a distinguisher D between G_3 and G_4. We build an adversary B breaking IND-CPA BLT security for $\mathcal{PKE}$. The adversary B uses D as a black-box as follows.

Reduction B^{D}:

1. Receive (pp, pk) from the challenger, sample $(\omega, \mathsf{tk}, \mathsf{ek}) \leftarrow \mathsf{Gen}(1^k)$ and give $pp' = (pp, \omega)$ and $pk' = pk$ to A.
2. Upon input a normal decryption query (c, π) from A, run the extractor to compute $(m, r) \leftarrow \mathsf{Ext}((pk, c), \pi, \mathsf{ek})$ and return m.
3. Upon input a tampering query $T_i \in \mathcal{T}_{\mathsf{sk}}$, forward T_i to the tampering oracle for $\mathcal{PKE}$. To answer a query (c, π), run the extractor to compute $(m, r) \leftarrow \mathsf{Ext}((pk, c), \pi, \mathsf{ek})$. Submit (m, r) to oracle $\mathsf{Dec}^*(\widetilde{sk}_i, \cdot, \cdot)$ and receive the answer $\widetilde{m}$. Return $\widetilde{m}$ to A.
4. Upon input a leakage query L_j, forward L_j to the leakage oracle for $\mathcal{PKE}$.
5. When A outputs $m_0, m_1 \in \mathcal{M}$, sample a random bit b' and output $(m_{b'}, 0^{|m_{b'}|})$. Let c_b be the corresponding challenge ciphertext. Compute $\pi_b \leftarrow \mathsf{S}((pk, c_b), \mathsf{tk})$ and forward (c_b, π_b) to A.
6. Output whatever D does.

Notice that the reduction perfectly simulates the environment for A; in particular c_b is either the encryption of randomly chosen message among (m_0, m_1) (as in G_3) or an encryption of zero (as in G_4). Since $\mathcal{PKE}$ is (λ, t, δ)-BLT secure, it must be $|\Pr[\mathsf{A} \text{ wins in } \mathsf{G}_3] - \Pr[\mathsf{A} \text{ wins in } \mathsf{G}_4]| \leq \delta(k)$.

As clearly $\Pr[\mathsf{A} \text{ wins in } \mathsf{G}_4] = 0$, we have obtained

$$\begin{aligned}\delta' &= |\Pr[\mathsf{A} \text{ wins in } \mathsf{G}_1] - \Pr[\mathsf{A} \text{ wins in } \mathsf{G}_4]| \\ &\leq |\Pr[\mathsf{A} \text{ wins in } \mathsf{G}_1] - \Pr[\mathsf{A} \text{ wins in } \mathsf{G}_2]| + |\Pr[\mathsf{A} \text{ wins in } \mathsf{G}_2] \\ &\qquad\qquad - \Pr[\mathsf{A} \text{ wins in } \mathsf{G}_3]| + |\Pr[\mathsf{A} \text{ wins in } \mathsf{G}_3] - \Pr[\mathsf{A} \text{ wins in } \mathsf{G}_4]| \\ &\leq \delta_1(k) + \delta_2(k) + \delta(k) = \delta(k) + negl(k).\end{aligned}$$

This concludes the proof. □

5.3 Instantiation from BHHO

We show that the variant of the encryption scheme introduced by Boneh *et al.* [7] used in [31] is IND-CPA BLT-secure. The proof relies on the simple observation that one can simulate polynomially many decryption queries for a given tampered key by only leaking a bounded amount of information from the secret key. Hence, security follows from leakage resilience.

The BHHO PKE scheme works as follows: (1) Algorithm Setup chooses a group $\mathbb{G}$ of prime order p with generator g and let $pp = (p, g)$; (2) Algorithm KGen samples random vectors $\mathbf{x}, \boldsymbol{\alpha} \in \mathbb{Z}_p^\ell$, computes $g^{\boldsymbol{\alpha}} = (g_1, \ldots, g_\ell)$ and let $sk = \mathbf{x} = (x_1, \ldots, x_\ell)$ and $pk = (h, g^{\boldsymbol{\alpha}})$ where $h = \prod_{i=1}^{\ell} g_i^{x_i}$; (3) Algorithm Enc takes as input pk and a message $m \in \mathcal{M}$, samples a random $r \in \mathbb{Z}_p$ and returns $c = \mathsf{Enc}(pk, m; r) = (g_1^r, \ldots, g_\ell^r, h^r \cdot m)$; (4) Algorithm Dec parses $c = (g^{\mathbf{c}_0}, c_1)$ and outputs $m = c_1 \cdot g^{-\langle \mathbf{c}_0, \mathbf{x} \rangle}$.

Proposition 5.1. *Let $k \in \mathbb{N}$ be the security parameter and assume that the DDH assumption holds in $\mathbb{G}$. Then, the BHHO encryption scheme is IND-CPA $(\lambda(k), t(k), \delta(k))$-BLT secure, where*

$$\lambda \leq (\ell - 2 - t)\log p - \omega(\log k), \qquad t \leq \ell - 3 \qquad \text{and} \qquad \delta = negl(k).$$

Proof. Naor and Segev [31, Section 5.2] showed that BHHO is IND-CPA leakage resilient up to $\lambda' \leq (\ell - 2)\log p - \omega(\log k)$. Assume there exists an adversary A which breaks IND-CPA BLT security, we build an adversary B which breaks IND-CPA leakage resilience of the scheme yielding a contradiction. (We omit a formal definition of IND-CPA security in the presence of leakage and refer the reader to [31, Definition 3.1] for the details.) Adversary B uses A as a black-box and is described below.

Reduction B^{A}:

1. Receive (pp, pk) from the challenger and forward there values to A.
2. Whenever A asks for a leakage query, submit this query to the leakage oracle and return the answer to A.
3. Upon input a tampering query $T_i \in \mathcal{T}_{\mathsf{sk}}$, submit a leakage function L to the leakage oracle such that $\widetilde{h}_j = \prod_{j=1}^{\ell} g_j^{-\widetilde{x}_j}$, where $\widetilde{x}_i = T_i(T_{i-1}(\cdots T_1(\mathbf{x})))$. When A asks for a decryption query (m, r), compute $\widetilde{m} = (h^r \cdot m) \cdot \widetilde{h}_i^r$.

4. Whenever A outputs $m_0, m_1 \in \mathcal{M}$, forward m_0, m_1 to the challenger. Let c_b be the corresponding challenge ciphertext; give c_b to A.
5. Output whatever A does.

Note that for each tampering query B has to leak one element in $\mathbb{Z}_p$. Using the value of λ' above this gives $\lambda = \lambda' - t \log p = (\ell - 2 - t) \log p - \omega(\log k)$. Moreover, B produces the right distribution since

$$\widetilde{m} = (h^r \cdot m) \cdot \widetilde{h}_i^r = c_1 \cdot \left(\prod_{i=1}^{\ell} g_i^{-\widetilde{x}_i} \right)^r = c_1 \cdot \prod_{i=1}^{\ell} g_i^{-r \cdot \widetilde{x}_i} = c_1 \cdot g^{-\sum_{i=1}^{\ell} r\alpha_i \cdot \widetilde{x}_i} = c_1 - g^{\langle \mathbf{c}_0, \widetilde{\mathbf{x}}_i \rangle},$$

where $(g^{\mathbf{c}_0}, c_1) = ((g^{r\alpha_1}, \ldots, g^{r\alpha_\ell}), h^{r \cdot} m)$ is an encryption of m using randomness r and public key h. This simulates perfectly the answer of oracle $\mathsf{Dec}^*(\widetilde{sk}_i, \cdot, \cdot)$. Hence, B has the same advantage as A and we can conclude that the scheme is IND-CPA BLT secure. □

6 Updating the Key in the *i*Floppy Model

We complement the results from the previous two sections by showing how to obtain security against an unbounded number of tampering queries in the floppy model of [2, 1]. Recall that in this model we assume the existence of an external tamper-free and leakage-free storage (the floppy), which is required for refresh operations. An important difference between the floppy model considered in this paper and the model of [1] is that in our case the floppy can contain "user-specific" information (e.g., its secret key), whereas in [1] it contains a *unique* master key which in principle could be equal for all users. To stress this difference, we refer to our model as the *i*Floppy model.

Clearly, the assumption of a unique master key makes production easier but it is also a single point of failure in the system since in case the content of the floppy is published (e.g., by a malicious user) the entire system needs to be re-initialized.[7] A solution for this is to assume that each floppy contains a different master key, resulting in a trade-off between security and production cost.

For simplicity, we consider a model with polynomially many updates where, between each update, the adversary is allowed to leak and tamper only once. However, the schemes in this section can be proven secure in the stronger model where between two key updates the attacker is allowed to leak adaptively λ bits from the current secret key and tamper with it for some bounded number of times.

6.1 ID Schemes in the *i*Floppy Model

An identification scheme $\mathcal{ID} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{P}, \mathsf{V}, \mathsf{Refresh})$ in the *i*Floppy model is defined as follows. (1) Algorithm Setup is defined as in a standard ID scheme. (2) Algorithm Gen outputs an update key uk together with an initial public/secret key pair (pk, sk). (3) Algorithms P and V are defined as in a standard ID scheme. (4) Algorithm $\mathsf{Refresh}$ takes as input the update key uk and outputs a new key sk' for the same public key pk.

Definition 6.1. Let $\lambda = \lambda(k)$ and $\delta = \delta(k)$ be parameters and let $\mathcal{T}_{\mathsf{sk}}$ be some set of functions such that $T \in \mathcal{T}_{\mathsf{sk}}$ has a type $T : \mathcal{SK} \to \mathcal{SK}$. We say that $\mathcal{ID}$ is $(\lambda(k), 1, \delta(k))$-CLT secure against

[7] We stress that in the schemes of [1] making the content of the floppy public does not constitute a total breach of security; however the security proof completely breaks down, leaving no security guarantee for the schemes at hand.

impersonation attacks with respect to $\mathcal{T}_{\mathsf{sk}}$ in the iFloppy model, if the following properties are satisfied.

(i) *Correctness.* For all $pp \leftarrow \mathsf{Setup}(1^k)$, $(pk, sk, uk) \leftarrow \mathsf{Gen}(1^k)$ we have that:

$$(\mathsf{P}(pp, sk) \rightleftarrows \mathsf{V}(pp))(pk) = (\mathsf{P}(pp, \mathsf{Refresh}(uk)) \rightleftarrows \mathsf{V}(pp))(pk) = accept.$$

(ii) *Security.* For all PPT adversaries A we have that $\Pr[\mathsf{A} \text{ wins}] \leq \delta(k)$ in the following game:

1. The challenger runs $pp \leftarrow \mathsf{Setup}(1^k)$ and $(pk, sk, uk) \leftarrow \mathsf{Gen}(1^k)$, and gives (pp, pk) to A; let $sk_1 = sk$.
2. The adversary is given oracle access to $\mathsf{P}(pp, sk_1)$.
3. The adversary may adaptively ask leakage and tampering queries. During the ith query:
   (a) A specifies a function $L_i : \{0,1\}^* \to \{0,1\}^\lambda$ and receives back $L_i(sk_i)$.
   (b) A specifies a function $T_i : \mathcal{SK} \to \mathcal{SK}$ and is given oracle access to $\mathsf{P}(pp, \widetilde{sk}_i)$, where $\widetilde{sk}_i = T_i(sk_i)$.
   (c) The challenger updates the secret key, $sk_{i+1} \leftarrow \mathsf{Refresh}(uk)$.
4. The adversary loses access to all oracles and interacts with an honest verifier V (holding public key pk). We say that A *wins* if $(\mathsf{A} \rightleftarrows \mathsf{V})(pk)$ outputs *accept*.

Remark 1. One could also consider a more general definition where between two key updates A is allowed to ask multiple leakage queries with output size λ_j, as long as $\sum_j \lambda_j \leq \lambda$. Similarly, we could allow A to tamper in each round for t times with the secret key sk_i. The constructions in this section can be proven secure in this extended setting, but we stick to Definition 6.1 for simplicity.

A general compiler. We now describe a compiler to boost any (λ, t)-BLT ID scheme (P, V), to a (λ, t)-CLT ID scheme $(\mathsf{P}', \mathsf{V}')$. The compiler is based upon a standard (not necessarily leakage or tamper resilient) signature scheme $\mathcal{SIG}$, and is described in Figure 6.

The basic idea is as follows. We generate the key pair (mpk, msk) using the key generation algorithm of the underlying signature scheme. We store msk in the floppy and publish mpk as P's identity. We also sample a key pair (pk, sk) for $\mathcal{ID}$ (which we call the *temporary* keys) and we provide the prover with a value help which is a signature of pk under the master secret key msk. Whenever P want to prove its identity, it first sends the temporary pk together with the helper value and V verifies this signature using mpk.[8] If the verification succeeds, P and V run an execution of $\mathcal{ID}$ where P proves it knows the secret key sk corresponding to pk. At the end of each authentication the prover updates its pair of temporary keys using the floppy, using the update key msk to sign the new public key pk' that is freshly generated. We obtain the following result.

Theorem 6.1. *Let $k \in \mathbb{N}$ be the security parameter. If $\mathcal{SIG}$ is EUF-CMA and $\mathcal{ID}$ is $(\lambda, 1, \delta)$-BLT secure against impersonation attacks with respect to $\mathcal{T}_{\mathsf{sk}}$, then the scheme $\mathcal{ID}'$ output by the compiler of Figure 6 is $(\lambda, 1, \delta')$-CLT against impersonation attacks with respect to $\mathcal{T}_{\mathsf{sk}}$ in the* i*Floppy model, where $\delta' \leq \delta + negl(k)$.*

[8] Alternatively P can send (pk, help) together with the first message of the identification scheme, in order to keep the same round complexity as in $\mathcal{ID}$.

iFloppy ID Compiler

Given as input an ID scheme $\mathcal{ID} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{P}, \mathsf{V})$ and a signature scheme $\mathcal{SIG} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vrfy})$ outputs an ID scheme $\mathcal{ID}' = (\mathsf{Setup}', \mathsf{Gen}', \mathsf{P}', \mathsf{V}', \mathsf{Refresh})$, specified below.

Setup'**:** Run $pp \leftarrow \mathsf{Setup}(1^k)$ and publish pp.

Gen'**:** Run the key generation algorithm of the underlying signature scheme, obtaining $(mpk, msk) \leftarrow \mathsf{KGen}(1^k)$. Also, run $(pk, sk) \leftarrow \mathsf{Gen}(1^k)$. The value mpk is the actual public key, whereas we refer to the values (pk, sk) as the *temporary* keys. Compute and publish a helper value $\mathsf{help} \leftarrow \mathsf{Sign}(msk, pk)$. The prover P' holds (pk, sk, help), the verifier V' holds mpk. The master key $uk = msk$ is the update key, which is stored in the floppy.

$\mathsf{P}' \rightleftarrows \mathsf{V}'$**:** The prover P' first sends the pair (pk, help) to V'. The verifier verifies the signature, i.e. it checks that $\mathsf{Vrfy}(mpk, (pk, \mathsf{help}))$ outputs *accept*. If the verification was successful, they run $(\mathsf{P}(pp, sk) \rightleftarrows \mathsf{V}(pp))(pk)$ and V' accepts if and only if the interaction leads to *accept*.

$\mathsf{Refresh}$**:** Sample a fresh pair $(pk', sk') \leftarrow \mathsf{Gen}(1^k)$ and update the helper value as in $\mathsf{help}' \leftarrow \mathsf{Sign}(msk, pk')$. The prover now holds $(pk', sk', \mathsf{help}')$.

Figure 6: Boosting BLT security to CLT security for ID schemes

Proof. We show that if there exists a PPT adversary A who wins the CLT security game against $\mathcal{ID}'$ with non-negligible probability, then we can build either of two reductions B or C violating BLT security of $\mathcal{ID}$ or EUF-CMA of $\mathcal{SIG}$ (respectively) with non-negligible probability. Let us assume that $\Pr[\mathsf{A} \text{ wins}] \geq \delta(k)$ for $\delta(k) = 1/p(k)$ for some polynomial $p(.)$ and infinitely many k. The CLT experiment for $\mathcal{ID}'$ is specified below:

CLT Experiment:

1. The challenger runs $pp \leftarrow \mathsf{Setup}'(1^k)$ and $(mpk, msk) \leftarrow \mathsf{KGen}(1^k)$, and gives (pp, mpk) to A.
2. For each $i = 1, \ldots, q(k)$ (where $q(k)$ is some polynomial in the security parameter), the challenger does the following:
   - During round i sample $(pk_i, sk_i) \leftarrow \mathsf{Gen}(1^k)$ and compute $\mathsf{help}_i \leftarrow \mathsf{Sign}(msk, pk_i)$.
   - Give A oracle access to $\mathsf{P}'((pp, \mathsf{help}_i, pk_i), sk_i)$.
   - Answer the leakage and tampering query from A using key sk_i.
3. During the impersonation stage, the challenger (playing now the role of the verifier V') receives the pair $(pk^\star, \mathsf{help}^\star)$ from A; if $\mathsf{Vrfy}(mpk, (pk^\star, \mathsf{help}^\star))$ outputs 0, the challenger outputs *reject*. Otherwise, it runs $(\mathsf{A} \rightleftarrows \mathsf{V}(pp))(pk^\star)$ and outputs whatever V does.

Let FRESH be the following event: The event becomes true if the pair $(pk^\star, \mathsf{help}^\star)$ used by A during the impersonation stage of the above experiment is equal to one of the pairs A has seen during the learning phase (i.e., one of the pairs (pk_i, help_i)). We have

$$\Pr[\mathsf{A} \text{ wins}] = \Pr[\mathsf{A} \text{ wins} \wedge \text{FRESH}] + \Pr\left[\mathsf{A} \text{ wins} \wedge \overline{\text{FRESH}}\right], \tag{17}$$

where all probabilities are taken over the randomness space of the CLT experiment and over the randomness of A. We now describe a reduction B (using A as a black-box) which breaks BLT security of $\mathcal{ID}$.

Reduction B^{A}:

1. Receive $pp \leftarrow \mathsf{Setup}(1^k)$ from the challenger. Sample $(mpk, msk) \leftarrow \mathsf{KGen}(1^k)$ and forward (pp, mpk) to A.
2. Choose an index $j \leftarrow [q]$ uniformly at random.
3. For all $i = 1, \ldots, q$, simulate the learning stage of A as follows.
   (a) During all rounds i such that $i \neq j$:
   - Sample $(pk_i, sk_i) \leftarrow \mathsf{Gen}(1^k)$ and compute $\mathsf{help}_i \leftarrow \mathsf{Sign}(msk, pk_i)$. Give A oracle access to $\mathsf{P}'((pp, \mathsf{help}_i, pk_i), sk_i)$.
   - Simulate A's leakage and tampering queries by using key sk_i.

   (b) During round j:
   - Receive the public key $\overline{pk}$ from the challenger and use this key as the jth temporary public key. Compute $\overline{\mathsf{help}} \leftarrow \mathsf{Sign}(msk, \overline{pk})$.
   - Simulate oracle $\mathsf{P}'((pp, \overline{\mathsf{help}}, \overline{pk}), \overline{sk})$ by forwarding $(\overline{pk}, \overline{\mathsf{help}})$ to A and using the target oracle $\mathsf{P}(pp, \overline{sk})$.
   - Simulate leakage query L_j and tampering query T_j by submitting the same functions to the target oracle.
4. Simulate the impersonation stage for A as follows:
   (a) Receive $(pk^\star, \mathsf{help}^\star)$ from A. If $pk^\star \neq \overline{pk}$ (i.e., B's guess is wrong) abort the execution. Otherwise, run $\mathsf{Vrfy}(mpk, (pk^\star, \mathsf{help}^\star))$ and output *reject* if verification fails.
   (b) Run $(\mathsf{A} \rightleftarrows \mathsf{V}(pp))(pk^\star)$ and use the messages from A in the impersonation stage, to answer the challenge from the target oracle.

Note that B's simulation is perfect, since it simulates all rounds using random keys whereas round j is simulated using the target oracle which allows for one tampering query and λ bits of leakage from $\overline{sk}$. Denote with GUESS the event that B guesses the index j correctly. Since B wins whenever A is successful and $\overline{\text{FRESH}}$ occurs, and moreover event GUESS is independent of all other events, we get

$$\begin{aligned} \Pr[\mathsf{B} \text{ wins}] &= \Pr[\mathsf{B} \text{ wins} \wedge \text{GUESS}] + \Pr\left[\mathsf{B} \text{ wins} \wedge \overline{\text{GUESS}}\right] \\ &\geq \Pr[\mathsf{B} \text{ wins} \wedge \text{GUESS}] = \frac{1}{q(k)} \Pr\left[\mathsf{A} \text{ wins} \wedge \overline{\text{FRESH}}\right]. \end{aligned} \tag{18}$$

We now describe a second reduction C (using A as a black-box), breaking existential unforgeability of $\mathcal{SIG}$.

Reduction C^{A}:

1. Run $pp \leftarrow \mathsf{Setup}(1^k)$, receive the public key mpk from the challenger and forward (pp, mpk) to A. Denote with msk the secret key corresponding to mpk (which of course is not known to C).
2. For all $i = 1, \ldots, q$, simulate the learning stage of A as follows:
   (a) Sample $(pk_i, sk_i) \leftarrow \mathsf{Gen}(1^k)$. Forward pk_i to the target signing oracle and receive back the corresponding signature $\mathsf{help}_i \leftarrow \mathsf{Sign}(msk, pk_i)$. Simulate oracle access to $\mathsf{P}'((pp, \mathsf{help}_i, pk_i), sk_i)$.

(b) Simulate the leakage and tampering query using knowledge of key sk_i.

3. During the impersonation stage:
   (a) Receive $(pk^\star, \mathsf{help}^\star)$ (which is a message-signature pair) from A and verify the signature with public key mpk. If verification fails, output some random guess and abort. (In that case A loses and C can only win with negligible probability.)
   (b) Otherwise, Run $(\mathsf{A} \rightleftarrows \mathsf{V}(pp))(pk^\star)$ and return to A whatever V does.
   (c) Output forgery $(m^\star = pk^\star, \sigma^\star = \mathsf{help}^\star)$.

Whenever FRESH occurs, the pair $(pk^\star, \mathsf{help}^\star)$ returned by A is such that this $pk^\star$ is different from all the pk_i's it has seen during the learning phase. In this case, whenever A wins, the forgery $(m^\star, \sigma^\star)$ output by C is a valid forgery. Hence,

$$\Pr[\mathsf{C} \text{ wins}] \geq \Pr[\mathsf{A} \text{ wins} \wedge \text{FRESH}]. \tag{19}$$

Combining Eq. (17)-(19), we obtain:

$$q(k) \cdot \Pr[\mathsf{B} \text{ wins}] + \Pr[\mathsf{C} \text{ wins}] \geq \Pr\left[\mathsf{A} \text{ wins} \wedge \overline{\text{FRESH}}\right] + \Pr[\mathsf{A} \text{ wins} \wedge \text{FRESH}] = \Pr[\mathsf{A} \text{ wins}] \geq \delta(k).$$

Hence either $\Pr[\mathsf{B} \text{ wins}] \geq \delta/(2q)$ or $\Pr[\mathsf{C} \text{ wins}] \geq \delta/2$, which are both non-negligible. □

Remark 2. Assuming factoring or DL is hard, we can instantiate Theorem 6.1 with our schemes from Section 4 resulting into tamper resilient identification schemes in the iFloppy model under polynomial many tampering and leakage attacks.

6.2 PKE Schemes in the *i*Floppy Model

A PKE scheme $\mathcal{PKE} = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Refresh})$ in the iFloppy model is defined as follows. (1) Algorithm Setup is defined as in a standard PKE scheme. (2) Algorithm KGen outputs an update key uk together with an initial public/secret key pair (pk, sk). (3) Algorithm Enc and Dec are defined as in a standard PKE scheme. (4) Algorithm $\mathsf{Refresh}$ takes as input the update key uk and outputs a new key sk' for the same public key pk.

Definition 6.2. Let $\lambda = \lambda(k)$ and $\delta = \delta(k)$ be parameters and let $\mathcal{T}_{\mathsf{sk}}$ be some set of functions such that $T \in \mathcal{T}_{\mathsf{sk}}$ has a type $T : \mathcal{SK} \to \mathcal{SK}$. We say that $\mathcal{PKE}$ is IND-CCA $(\lambda(k), 1, \delta(k))$-CLT secure with respect to $\mathcal{T}_{\mathsf{sk}}$ in the iFloppy model, if the following properties are satisfied.

(i) *Correctness.* For all $pp \leftarrow \mathsf{Setup}(1^k)$, $(pk, sk, uk) \leftarrow \mathsf{Gen}(1^k)$ we have that:

$$\Pr[\mathsf{Dec}(\mathsf{Refresh}(uk), \mathsf{Enc}(pk, m)) = m] = 1.$$

(ii) *Security.* For all PPT adversaries A we have that $\Pr[\mathsf{A} \text{ wins}] \leq \delta(k)$ in the following game:

1. The challenger runs $pp \leftarrow \mathsf{Setup}(1^k)$ and $(pk, sk, uk) \leftarrow \mathsf{Gen}(1^k)$, and gives (pp, pk) to A; let $sk_1 = sk$.
2. The adversary is given oracle access to $\mathsf{Dec}(sk_1, \cdot)$.
3. The adversary may adaptively ask leakage and tampering queries. During the ith query:
   (a) A specifies a function $L_i : \{0,1\}^* \to \{0,1\}^\lambda$ and receives back $L_i(sk_i)$.

(b) A specifies a function $T_i : \mathcal{SK} \to \mathcal{SK}$ and is given oracle access to $\mathsf{Dec}(\widetilde{sk}_i, \cdot)$, where $\widetilde{sk}_i = T_i(sk_i)$.

(c) The challenger updates the secret key, $sk_{i+1} \leftarrow \mathsf{Refresh}(uk)$.

4. The adversary outputs two messages of the same length $m_0, m_1 \in \mathcal{M}$ and the challenger computes $c_b \leftarrow \mathsf{Enc}(pk, m_b)$ where b is a uniformly random bit.
5. The adversary outputs a bit b' and wins if $b = b'$.

The same considerations of Remark 1 hold here.

Construction from BHHO. As noted in [1], the BHHO PKE scheme (cf. Section 5.3) allows for a very simple update mechanism. When we plug this encryption scheme in the construction of Figure 5, we obtain the following scheme. (1) Algorithm Setup chooses a group $\mathbb{G}$ of prime order p with generator g, runs $(\omega, \mathsf{tk}, \mathsf{ek}) \leftarrow \mathsf{Gen}(1^k)$ and lets $pp = (p, g, \omega)$. (2) Algorithm KGen samples random vectors $\boldsymbol{\alpha}, \mathbf{x} \in \mathbb{Z}_p^\ell$ and sets $uk = (\boldsymbol{\alpha}, \mathbf{x})$; furthermore it chooses $sk = \mathbf{x}_1 = \mathbf{x} + \boldsymbol{\beta}$ (where $\boldsymbol{\beta} \leftarrow \ker(\boldsymbol{\alpha})$) and lets $pk = (h, g^{\boldsymbol{\alpha}})$ for $h = g^{\langle \boldsymbol{\alpha}, \mathbf{x} \rangle}$. (3) Algorithm Enc takes as input pk and a message $m \in \mathcal{M}$, samples a random $r \in \mathbb{Z}_p$ and returns $c = (g^{r\boldsymbol{\alpha}}, h^r \cdot m)$ together with a proof $\pi \leftarrow \mathsf{Prove}^\omega((pk, c), (m, r))$. (4) Algorithm Dec parses $c = (g^{\mathbf{c}_0}, c_1)$, runs $\mathsf{Verify}^\omega((pk, c), \pi)$ and outputs $m = c_1 \cdot g^{-\langle \mathbf{c}_0, \mathbf{x}_1 \rangle}$ in case the verification succeeds and $\perp$ otherwise. (5) Algorithm $\mathsf{Refresh}$ samples $\boldsymbol{\beta}_i \leftarrow \ker(\boldsymbol{\alpha})$ and outputs $\mathbf{x}_i = \mathbf{x} + \boldsymbol{\beta}_i$.

The theorem below shows that the above scheme is IND-CCA CLT-secure in the iFloppy model. One would expect that a proof of this fact is simple, since the keys after each update are completely fresh and independent (given the public key) and thus security should follow from BLT security of the underlying scheme. However, it is easy to see that such a proof strategy does not work directly (at least in a black-box way).[9] Unfortunately this requires us to make the proof from scratch. Since the proof relies on ideas already introduced in this paper or borrowed from [1], we give only a sketch here.

Theorem 6.2. *Let $k \in \mathbb{N}$ be the security parameter. Assume that the DDH assumption holds in $\mathbb{G}$. Then, the PKE scheme described above is IND-CCA $(\lambda(k), 1, negl(k))$-CLT secure with respect to $\mathcal{T}_{\mathsf{sk}}$ in the* i*Floppy model, where $\lambda \leq (\ell - 3) \log p - \omega(\log k)$.*

Proof (sketch). We define a series of games (starting with the original IND-CCA CLT game) and prove that they are all close to each other.

Game G_1. This is the IND-CCA CLT game. In particular the challenge ciphertext is a pair of the form $(c^* = (g^{r\boldsymbol{\alpha}}, h^r \cdot m_b), \pi^*)$ where $\pi^* \leftarrow \mathsf{Prove}^\omega((pk, c^*), (m_b, r))$, for $m_b \in \{m_0, m_1\}$ and $b \leftarrow \{0, 1\}$. We assume that

$$\Pr[\mathsf{A} \text{ wins in } \mathsf{G}_1] = \frac{1}{2} + \delta(k).$$

Game G_2. In this game we change the way the challenge ciphertext is computed by replacing the argument π^* with a simulated argument $\pi^* \leftarrow \mathsf{S}((pk, c^*), \mathsf{tk})$. It follows from the composable NIZK property of the argument system that G_1 and G_2 are computationally close.

[9] We stress that in the PKE case we cannot apply the same trick as for the compiler of Figure 6, since that would require to make the scheme interactive.

Game G_3. In this game we change the way decryption queries are handled. Queries (c, π) to $\mathsf{Dec}(\mathbf{x}_i, \cdot)$ (such that π accepts) are answered by running the extractor Ext on π, yielding $(m, r) \leftarrow \mathsf{Ext}((pk, c), \pi, \mathsf{ek})$, and returning m. Queries (c, π) to $\mathsf{Dec}(\widetilde{\mathbf{x}}_i, \cdot)$ (such that π accepts) are answered as follows. We first extract $(m, r) \leftarrow \mathsf{Ext}((pk, c), \pi, \mathsf{ek})$ as above. Then, instead of returning m, we recompute $c = \mathsf{Enc}(pk, m; r)$ and return $\widetilde{m} = \mathsf{Dec}(\widetilde{\mathbf{x}}_i, c)$.

As argued in the proof of Theorem 5.1, G_2 and G_3 are computationally close by the tSE property of the argument system.

Game G_4. In this game we change the way the secret keys are refreshed. The challenger first chooses a random $(\ell - 2)$-dimensional subspace $S \subset \ker(\boldsymbol{\alpha})$ and samples the new keys $\mathbf{x}_i$ from the affine subspace $\mathbf{x} + S$. We prove that G_3 and G_4 are statistically close by a hybrid argument. Assume there are $q = poly(k)$ updates and define for each $i = 0, \ldots, q$ the following hybrid distribution:

Game $\mathsf{G}_{3,i}$. Sample at the beginning a random $(\ell-2)$-dimensional subspace $S \subset \ker(\boldsymbol{\alpha})$ and modify the refreshing of the key as follows.

- For every $1 < j \leq q - i$, let $\mathbf{x}_j = \mathbf{x} + \boldsymbol{\beta}_j$ where $\boldsymbol{\beta}_j \leftarrow \ker(\boldsymbol{\alpha})$.
- For every $q - i < j \leq q$, let $\mathbf{x}_j = \mathbf{x} + \mathbf{s}_j$ where $\mathbf{s}_j \leftarrow S$.

Note that $\mathsf{G}_3 = \mathsf{G}_{3,0}$ and $\mathsf{G}_4 = \mathsf{G}_{3,q}$. As argued in [1, Theorem 13] it follows from the affine version of the subspace hiding lemma (see [1, Corollary 8]) that as long as the leakage is bounded an adversary cannot distinguish leakage on $\boldsymbol{\beta}_i \leftarrow \ker(\boldsymbol{\alpha})$ from leakage on $\mathbf{s}_i \leftarrow S$ (and this holds even if $\boldsymbol{\alpha}$ is public and known at the beginning of the experiment and S becomes known after the leakage occurs). We do loose an additional factor $\log p$ in the leakage bound here, due to the fact that we use one additional leakage query to leak the group element $\widetilde{h}_i$ needed to simulate the tampered decryption oracle $\mathsf{Dec}(\widetilde{\mathbf{x}}_i, \cdot)$ (as we do in the proof of Proposition 5.1). This yields the bound $\lambda \leq (\ell - 3) \log p - \omega(\log k)$ on the tolerated leakage.

Game G_5. In this game we compute the component c^* of the challenge ciphertext (c^*, π^*) as

$$c^* = (g^{\mathbf{c}_0} = g^{r\boldsymbol{\alpha}}, c_1 = g^{\langle \mathbf{c}_0, \mathbf{x} \rangle} \cdot m_b). \tag{20}$$

This is only a syntactical change since $g^{\langle \mathbf{c}_0, \mathbf{x} \rangle} \cdot m_b = (g^{\langle \boldsymbol{\alpha}, \mathbf{x} \rangle})^r \cdot m_b = h^r \cdot m_b$.

Game G_6. In this game the challenger chooses $\boldsymbol{\alpha}$, $\mathbf{x}$ as before and in addition samples a vector $\mathbf{c}_0 \leftarrow \mathbb{Z}_p^\ell$ and sets S to be the $(\ell - 2)$-dimensional subspace $S = \ker(\boldsymbol{\alpha}, \mathbf{c}_0)$. The secret keys $\mathbf{x}_i$ are chosen as in the previous game from S. The component c^* of the challenge ciphertext (c^*, π^*) is computed as in Eq. (20) using the above vector $\mathbf{c}_0$.

As shown in [1], G_5 and G_6 are computationally close by the extended rank-hiding assumption (which is equivalent to DDH).

Game G_7. In this game we change again the way the keys are refreshed, namely each key $\mathbf{x}_i$ is sampled from the full original $(\ell - 1)$-dimensional space $\mathbf{x} + \ker(\boldsymbol{\alpha})$. As before, the last two games are close by the affine subspace hiding lemma.

Game G_8. In the last game we change the way the challenge ciphertext is chosen. Namely, we choose a random $v \leftarrow \mathbb{Z}_p$ and let $c^* = (g^{\mathbf{c}_0}, g^v)$. Game G_8 and G_7 are statistically close since G_7 does not reveal anything about $\mathbf{x}$ beyond $\langle \boldsymbol{\alpha}, \mathbf{x} \rangle$ from the public key, and thus $\langle \mathbf{c}_0, \mathbf{x} \rangle$ are statistically close to uniform.

Note that the second element is now independent of the message. Hence, the probability that A wins in G_8 is $1/2$.

□

Acknowledgments

The authors thank Krzysztof Pietrzak for helpful discussions and insights into the tamper resilience of weak PRFs. We also thank Jesper Buus Nielsen for helpful suggestions.

References

[1] Shweta Agrawal, Yevgeniy Dodis, Vinod Vaikuntanathan, and Daniel Wichs. On continual leakage of discrete log representations. *IACR Cryptology ePrint Archive*, 2012:367, 2012.

[2] Joël Alwen, Yevgeniy Dodis, and Daniel Wichs. Leakage-resilient public-key cryptography in the bounded-retrieval model. In *CRYPTO*, pages 36–54, 2009.

[3] Ross Anderson and Markus Kuhn. Tamper resistance: a cautionary note. In *WOEC'96: Proceedings of the 2nd conference on Proceedings of the Second USENIX Workshop on Electronic Commerce*, pages 1–1, Berkeley, CA, USA, 1996. USENIX Association.

[4] Mihir Bellare, David Cash, and Rachel Miller. Cryptography secure against related-key attacks and tampering. In *ASIACRYPT*, pages 486–503, 2011.

[5] Mihir Bellare and Tadayoshi Kohno. A theoretical treatment of related-key attacks: Rka-prps, rka-prfs, and applications. In *EUROCRYPT*, pages 491–506, 2003.

[6] Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. On the importance of eliminating errors in cryptographic computations. *J. Cryptology*, 14(2):101–119, 2001.

[7] Dan Boneh, Shai Halevi, Michael Hamburg, and Rafail Ostrovsky. Circular-secure encryption from decision diffie-hellman. In *CRYPTO*, pages 108–125, 2008.

[8] Zvika Brakerski, Yael Tauman Kalai, Jonathan Katz, and Vinod Vaikuntanathan. Overcoming the hole in the bucket: Public-key cryptography resilient to continual memory leakage. In *FOCS*, pages 501–510, 2010.

[9] Ronald Cramer. *Modular Design of Secure yet Practical Cryptographic Protocols*. PhD thesis, University of Amsterdam, November 1996.

[10] Giovanni Di Crescenzo, Richard J. Lipton, and Shabsi Walfish. Perfectly secure password protocols in the bounded retrieval model. In *TCC*, pages 225–244, 2006.

[11] Dana Dachman-Soled and Yael Tauman Kalai. Securing circuits against constant-rate tampering. In *CRYPTO*, pages 533–551, 2012.

[12] Yevgeniy Dodis, Kristiyan Haralambiev, Adriana López-Alt, and Daniel Wichs. Cryptography against continuous memory attacks. In *FOCS*, pages 511–520, 2010.

[13] Yevgeniy Dodis, Kristiyan Haralambiev, Adriana López-Alt, and Daniel Wichs. Efficient public-key cryptography in the presence of key leakage. In *ASIACRYPT*, pages 613–631, 2010.

[14] Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM J. Comput.*, 38(1):97–139, 2008.

[15] Yevgeniy Dodis and Yu Yu. Overcoming weak expectations. In *TCC*, pages 1–22, 2013.

[16] Stefan Dziembowski. Intrusion-resilience via the bounded-storage model. In *TCC*, pages 207–224, 2006.

[17] Stefan Dziembowski, Tomasz Kazana, and Daniel Wichs. One-time computable self-erasing functions. In *TCC*, pages 125–143, 2011.

[18] Stefan Dziembowski, Krzysztof Pietrzak, and Daniel Wichs. Non-malleable codes. In *ICS*, pages 434–452, 2010.

[19] Sebastian Faust, Krzysztof Pietrzak, and Daniele Venturi. Tamper-proof circuits: How to trade leakage for tamper-resilience. In *ICALP (1)*, pages 391–402, 2011.

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

[21] Marc Fischlin and Roger Fischlin. The representation problem based on factoring. In *CT-RSA*, pages 96–113, 2002.

[22] Rosario Gennaro, Anna Lysyanskaya, Tal Malkin, Silvio Micali, and Tal Rabin. Algorithmic tamper-proof (atp) security: Theoretical foundations for security against hardware tampering. In *TCC*, pages 258–277, 2004.

[23] Jens Groth. Simulation-sound nizk proofs for a practical language and constant size group signatures. In *ASIACRYPT*, pages 444–459, 2006.

[24] Louis C. Guillou and Jean-Jacques Quisquater. A "paradoxical" indentity-based signature scheme resulting from zero-knowledge. In *CRYPTO*, pages 216–231, 1988.

[25] Yuval Ishai, Manoj Prabhakaran, Amit Sahai, and David Wagner. Private circuits II: Keeping secrets in tamperable circuits. In *EUROCRYPT*, pages 308–327, 2006.

[26] Yael Tauman Kalai, Bhavana Kanukurthi, and Amit Sahai. Cryptography with tamperable and leaky memory. In *CRYPTO*, pages 373–390, 2011.

[27] Jonathan Katz and Vinod Vaikuntanathan. Signature schemes with bounded leakage resilience. In *ASIACRYPT*, pages 703–720, 2009.

[28] Eike Kiltz, Krzysztof Pietrzak, Martijn Stam, and Moti Yung. A new randomness extraction paradigm for hybrid encryption. In *EUROCRYPT*, pages 590–609, 2009.

[29] Feng-Hao Liu and Anna Lysyanskaya. Tamper and leakage resilience in the split-state model. In *CRYPTO*, pages 517–532, 2012.

[30] Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. *J. ACM*, 51(2):231–262, 2004.

[31] Moni Naor and Gil Segev. Public-key cryptosystems resilient to key leakage. In *CRYPTO*, pages 18–35, 2009.

[32] Tatsuaki Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In *CRYPTO*, pages 31–53, 1992.

[33] Krzysztof Pietrzak. Subspace lwe. In *TCC*, pages 548–563, 2012.

[34] Stephen Pohlig and Martin Hellman. An improved algorithm for computing logarithms over and its cryptographic significance. *IEEE Transactions on Information Theory*, 24(1):106–110, 1978.

A Necessity of Update/Self-Destruct, Revisited

Having at hand some scheme which is resilient against a bounded number of tampering queries, a natural question is whether it is possible to get unbounded tamper resilience in a cryptographic setting where the key is never updated. This question has been already addressed by Gennaro et al. [22]. They showed that for many natural primitives such as public key encryption and signatures, there is a general attack which extracts the entire secret key, as long as the card never self-destructs (or the key is never updated).

Assume that the functionality has a testing procedure Test-Dev which can test the device for malfunctioning. Such a procedure enjoys two properties: (1) It always accepts a device whose secret content sc is correct; (2) If it accepts a secret device with secret content sc' with non-negligible probability, then finding sc' is a valid attack for the functionality the device is implementing. In case of a signature scheme, Test-Dev would just store the public key corresponding to the original secret key and run the verification algorithm on faulty signatures (i.e., signatures generated with a tampered key). In this way it is possible to learn the secret key bit by bit. A similar attack works in case of encryption schemes where one has access to a decryption oracle and, more generally, for any cryptographic primitive with a Test-Dev procedure as described above.

Although very general, we point out that there are contexts in which the above attack does not work. Consider for instance any *weak* pseudorandom function (wPRF). Security of a wPRF $F(sc, x)$ requires that it is hard to distinguish the output of the function from the output of a random function *on random inputs*. Now an adversary A applying the above attack in the security game of a wPRF would first query the challenge oracle once, obtaining a pair (x, y) where y is either the output of a random function or the output of $F(sc, x)$ for a random x. If A tampers with the first bit of sc (i.e., by setting it to 0) and queries the oracle again, it will receive a pair (x', y') for some value x' which is chosen randomly and independently of x. In this case, it is not possible for A to verify if its guess on the first bit of sc was correct or not. So Gennaro et al.'s attack does not apply here.

Our attack. We define a different testing procedure (which we call Test-Dev'), such that it does not suffer from the issue above. Test-Dev' contains a description of the functionality F implemented by the device at hand. Moreover, it takes (a subset) of the following values as inputs: (i) The public content pc; (ii) Input-output pairs (x, y); (iii) A candidate secret key sc; (iv) Some auxiliary value aux. Hence Test-Dev' outputs 1 if and only if (x, y) is a consistent input-output pair for F with respect to values pc, sc and aux.

Concretely, in case of a signature scheme the functionality F would consist of the signing and verification algorithm of the underlying signature scheme and Test-Dev' would take as input the public content pc (i.e., the public key) together with a message-signature pair (m, σ) and some candidate secret key sc'. Hence, Test-Dev' can compute a new signature σ' of m using sc and output 1 provided that both signatures σ and σ' verify correctly with respect to pc. For wPRFs, the functionality F consists in the description of the wPRF. The adversary can input a candidate key sc' together with some input-output pair (x, y): Test-Dev' outputs 1 if and only if $y = F(sc', x)$. The key difference here, is that the adversary cannot run Test-Dev' without choosing some key that he wants to test.

Claim 6. *No cryptographic device that can be efficiently test for malfunctioning can be made tamper-proof without updating the key (or without the self-destruct capability).*

Proof. Without loss of generality, assume the secret content is n bit long. We are given a device which implements some functionality F (think of it as a wPRF or a signature scheme). We show how to extract the key using n tampering queries. During the first query, we let $T_1(\cdot)$ set the first bit of sc to some value $b_1 \in \{0, 1\}$ and leave the remaining $n - 1$ bits of sc unchanged. In other words, $sc_1 = (b_1, sc[2], \ldots, sc[n])$. We now get from the oracle some pair (x_1, y_1), where y_1 is either random or computed as $F(sc_1, x_1)$. Similarly, we set the second bit of the key to $b_2 \in \{0, 1\}$ and obtain a second pair (x_2, y_2) where y_2 is either random or computed as $F(sc_2, x_2)$ for $sc_2 = (b_1, b_2, sc[3] \ldots, sc[n])$.

After obtaining $n - 1$ pairs $(x_1, y_1), \ldots, (x_{n-1}, y_{n-1})$ as above, we end-up with a tampered key $sc_{n-1} = (b_1, \ldots, b_{n-1}, sc[n])$. We now guess the value $b^\star$ and run Test-Dev' upon input $sc_n^\star = (b_1, \ldots, b_{n-1}, b_n^\star)$ and the pair (x_{n-1}, y_{n-1}): If the output is 1 we guess $sc[n] = b_n^\star$ and otherwise $sc[n] = \overline{b_n^\star}$. (In case Test-Dev' has only non-negligible probability of success we simply run it many times and take majority of the outputs.) We thus recover the other bits of the secret key going backwards: We sample $b_{n-1}^\star \in \{0, 1\}$, run Test-Dev' upon input $sc_{n-1}^\star = (b_1, \ldots, b_{n-2}, b_{n-1}^\star, sc[n])$ and (x_{n-2}, y_{n-2}) and guess $sc[n-1] = b_{n-1}^\star$ or $sc[n-1] = \overline{b_{n-1}^\star}$ depending on the output of Test-Dev' and so on. In this way we can extract the entire key. □

B Tampering with Computation

We allow the adversary A to tamper in an arbitrary way with the algorithm of the prover P as long as the interfaces of the algorithm stay unchanged (input/output domain consistency) and the adversary can run the tampered algorithm only a bounded number of times between two key updates. To model the input/output consistency, we let A replace the algorithm P with an arbitrarily different algorithm P′ as long as P and P′ have the same input/output domain. Formally, we model such arbitrary tampering with the computation by an adversary that corrupts the prover P, and we denote the adversarial controlled prover by P′. Of course, P cannot be corrupted by the adversary A itself as this would enable A to learn the entire secret key and completely break security

of the identification scheme. We follow Dziembowski et al. [17] and consider a big adversary A and a small adversary B, where we can think of B as a "virus" that corrupts the prover while A is the adversary that observes (possibly corrupted) protocol executions with P'. Notice that the only way in which B can "communicate" with the big adversary A is via the output of the tampered prover P'. We formally describe security with respect to tampering with the computation in the definition below. For simplicity, we assume that the adversary only gets a single protocol transcript after each tampering query. This can be generalized to an arbitrary constant number but we omit the details here.

Definition B.1. Let $\lambda = \lambda(k)$ be the leakage parameter. We say that $\mathcal{ID}$ is a λ-*continuous leakage and tampering with computation (CLTC)* secure identification scheme in the iFloppy model if additionally to correctness (cf. Definition 6.1) the scheme satisfies the following property:

CLTC-Security: For all PPT adversaries A there exists a negligible function $\delta : \mathbb{N} \to [0,1]$ such that $\Pr[\mathsf{A} \text{ wins}] \leq \delta(k)$ in the following game:

1. The challenger runs $pp \leftarrow \mathsf{Setup}(1^k)$ and $(pk, sk, uk) \leftarrow \mathsf{Gen}(1^k)$, and gives (pp, pk) to A. Let $sk_1 = sk$ and uk be stored on the floppy.
2. We repeat the following steps a polynomial number of times, where the adversary may adaptively ask leakage and tampering queries and each round is completed with an update of the secret key using the floppy. More precisely, in the ith round the following happens:
   (a) A specifies a function $L_i : \{0,1\}^* \to \{0,1\}^\lambda$ and receives back $L_i(sk_i)$.
   (b) A specifies a tampering algorithm B_i and obtains the faulty transcript $(\mathsf{B}_i(pp, sk_i), \mathsf{V}(pp))(pk)$.
   (c) The challenger updates the secret key, $sk_{i+1} \leftarrow \mathsf{Refresh}(uk)$.
3. The adversary loses access to all oracles and interacts with an honest verifier V (holding pk). We say that A *wins* if $(\mathsf{A} \rightleftarrows \mathsf{V}(pp))(pk)$ outputs *accept*.

In the theorem below we show that when we instantiate the general compiler from Figure 6 with an appropriate identification scheme with key size $k \gg s$ (s is the length of the transcript) and security against s bits of leakage, we can achieve security with respect to Definition B.1. Identification schemes that are secure in the Bounded Retrieval Model (BRM) satisfy these condition and have been constructed by Alwen et al. [2] based on the Generalized Okamoto ID scheme.

Theorem B.1. *Let $\mathcal{SIG} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vrfy})$ be an EUF-CMA secure signature scheme and $\mathcal{ID} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{P}, \mathsf{V})$ be an $(s+\lambda)$-leakage and 0-tamper resilient identification scheme with transcript length s. Then, $\mathcal{ID}'$ from Figure 6 is a λ-CLTC secure identification scheme in the* i*Floppy model.*

Proof (sketch). The proof is similar to the proof of Theorem 6.1. The only difference is in the reduction to the security of the underlying identification scheme $\mathcal{ID}$. While in Theorem 6.1 we simulate the tampering with access to the tampering oracle, we here simulate the tampering queries B_i, i.e., the faulty transcript $(\mathsf{B}_i(pp, sk_i), \mathsf{V}(pp))(pk)$ with access to the leakage oracle. As the transcript has length s, we can learn the entire faulty transcript from the leakage oracle. This is where we loose s bits in the leakage bound compared to the underlying identification scheme. □

We note that the above result seemingly achieves a stronger security notion than Theorem 6.1 (tampering with the computation vs. tampering only with the state) while not requiring a bounded

tamper resilient identification scheme as the underlying primitive. The fundamental difference between both theorems comes from the fact that in the theorem above we can only use the identification scheme a bounded number of times between each two key updates, while when we tamper only with the secret state Theorem 6.1 does not set any such usage restriction.

C Proof of Lemma 3.1

Denote with $d = \log|\mathcal{H}|$ the size of the set $\mathcal{H}$. For random variables Z, Z' such that Z' is an independent copy of Z we write $Col(Z) = \Pr[Z = Z']$ for the collision probability of Z. In particular,

$$\begin{aligned}
&Col((h_S, h_S(X_1), h_S(X_2), \ldots, h_S(X_t))) \\
&= \Pr\left[(h_S, h_S(X_1), h_s(X_2), \ldots, h_S(X_t)) = (h'_S, h'_S(X'_1), h'_s(X'_2), \ldots, h'_s(X'_t))\right] \\
&= \Pr\left[h_S = h'_S\right] \cdot \Pr\left[(h_S, h_S(X_1), h_S(X_2), \ldots, h_S(X_t)) = (h'_S, h'_S(X'_1), h'_S(X'_2), \ldots, h'_S(X'_t)) \mid h_S = h'_S\right] \\
&= 2^{-d} \cdot \Pr\left[(h_S, h_S(X_1), h_S(X_2), \ldots, h_S(X_t)) = (h_S, h_S(X'_1), h_S(X'_2), \ldots, h_S(X'_t))\right] \qquad (21)
\end{aligned}$$

where the probabilities above are over the choices of $h_S, (X_1, X_2, \ldots, X_t)$ and $h'_S, (X'_1, X'_2, \ldots, X'_t)$.

We define an event E such that conditioning on E happening we can apply the assumption that h_S is $2t$-wise independent, and thus bound the probability in Eq. (21) by $2^{-t\cdot\ell}$. The event E becomes true when $X_1, X_2, \ldots, X_t, X'_1, X'_2, \ldots, X'_t$ are pairwise different. Notice that there are $\binom{2t}{2}$ such pairs, however by assumption $(X_1, \ldots, X_t)$ are pairwise different; this leaves us with $\binom{2t}{2} - 2\binom{t}{2} = t^2$ pairs. Hence, by the union bound

$$\Pr\left[\overline{E}\right] = \Pr\left[X_1 = X_2 \vee X_2 = X_3 \vee \ldots \vee X'_{t-1} = X'_t\right] \leq t^2 \cdot 2^{-\beta},$$

where the inequality comes from the assumption that all random variables have individually min-entropy at least β and by applying the union bound.

Plugging the last expression in Eq. (21) and using the fact that h_S is $2t$-wise independent yields

$$\begin{aligned}
&Col((h_S, h_S(X_1), h_S(X_2), \ldots, h_S(X_t))) \\
&\leq 2^{-d} \cdot \left(\Pr\left[(h_S, h_S(X_1), h_S(X_2), \ldots, h_S(X_t)) = (h_S, h_S(X'_1), h_S(X'_2), \ldots, h_S(X'_t)) \mid E\right] + \Pr\left[\overline{E}\right]\right) \\
&\leq 2^{-d} \cdot (2^{-t\cdot\ell} + t^2 \cdot 2^{-\beta}).
\end{aligned}$$

Let Z be a random variable with support $\mathcal{Z}$ and U be uniform over $\mathcal{Z}$. Then $\|Z - U\|_2^2 = Col(Z) - |\mathcal{Z}|^{-1}$. In particular,

$$\begin{aligned}
\|(h_S, h_S(X_1), h_S(X_2), \ldots, h_S(X_t))\|_2^2 &= Col((h_S, h_S(X_1), h_s(X_2), \ldots, h_S(X_t))) - 2^{-d-t\cdot\ell} \\
&\leq 2^{-d} \cdot \left(2^{-t\cdot\ell} + t^2 \cdot 2^{-\beta}\right) - 2^{-d-t\cdot\ell} = t^2 \cdot 2^{-d-\beta}.
\end{aligned}$$

Finally, using that $\|Z\|_1 \leq \sqrt{|\mathcal{Z}|} \cdot \|Z\|_2$ for any random variable Z with support $\mathcal{Z}$, we obtain

$$\begin{aligned}
&\Delta((h_S, h_S(X_1), h_s(X_2), \ldots, h_S(X_t)); (h_S, \underbrace{U_{\mathcal{Y}}, \ldots, U_{\mathcal{Y}}}_{t \text{ times}})) \\
&\qquad = \frac{1}{2}\|(h_S, h_S(X_1), h_S(X_2), \ldots, h_S(X_t)) - (h_S, \underbrace{U_{\mathcal{Y}}, \ldots, U_{\mathcal{Y}}}_{t \text{ times}})\|_1 \\
&\qquad \leq \frac{1}{2}\sqrt{2^{d+t\cdot\ell}} \cdot \sqrt{t^2 \cdot 2^{-d-\beta}} = \frac{t}{2} \cdot 2^{(t\cdot\ell-\beta)/2}.
\end{aligned}$$

D Proof of the Chaining Lemma

Before coming to the actual proof, we state and prove two general lemmas.

The first lemma states that if the support of a distribution is sufficiently large then there always exists an event E such that conditioned on E the conditional distribution has high min-entropy.

Lemma D.1. *For $n \in \mathbb{N}_{>1}$ let c be some parameter such that $\sqrt{n} < c < n$. Let $\mathcal{X}$ be a set of size $2^n = |\mathcal{X}|$ and X be a distribution over $\mathcal{X}$ with $|\mathsf{sup}(X)| > 2^c$ such that for all $x \in \mathsf{sup}(X)$ we have $\Pr[X = x] \geq \frac{1}{2^n}$. There exists an event E such that:*

1. $\mathbf{H}_\infty(X_{|E}) > c - 2\sqrt{n}$, *and*
2. $|\mathsf{sup}(X_{|\overline{E}})| < |\mathsf{sup}(X)|$.

Proof. Intuitively, the lemma is proven by showing that if a distribution has sufficiently large support, then over a large subset of the support the distribution must be "almost" flat. We will describe below what it means for a distribution to be "almost flat". We then define an event E that occurs when X takes some value in the almost flat area. Clearly, X conditioned on E must be "almost" uniformly distributed, and if furthermore the support of X conditioned on E is still sufficiently large, we get that $\mathbf{H}_\infty(X_{|E})$ must be large. We proceed with the formal proof.

We introduce a parameter b which is a positive integer such that $c > n/b$. Later we fix b to its optimal value. For ease of description we assume that n is a multiple of b. We start by defining what it means for an area to be flat. For some probability distribution X we define $k \in [2^{n/b} - 1]$ sets as follows:

1. For $k \in [2^{n/b} - 1]$ we have: $I_k := \left\{x \in \mathsf{sup}(X) : \frac{k^b}{2^n} \leq \Pr[X = x] < \frac{(k+1)^b}{2^n}\right\}$, and
2. $I_{2^{n/b}} = \{x \in \mathsf{sup}(X) : \Pr[X = x] = 1\}$.

These sets characterize the (potential) flat areas in the distribution X as the probability of all values in some set I_k lie in a certain range that is bounded from below and above. Clearly, the sets I_k are pairwise disjoint and cover the whole space between $1/2^n$ and 1. Therefore, each $x \in \mathsf{sup}(X)$ with some probability $\Pr[X = x]$ must fall into some unique set I_k.

We denote by I_m the set that contains the most elements among all sets I_k, and define the event E as the event that occurs when $x \in \mathsf{sup}(X)$ falls into I_m, i.e., X takes a value that falls in the largest set I_m. We now lower bound the probability that E occurs.

$$\Pr[E] \geq |I_m| \frac{m^b}{2^n} \tag{22}$$

$$\geq 2^{c-n/b} \frac{m^b}{2^n} \tag{23}$$

Inequality (22) holds as for all $x \in I_m$ we have $\Pr[X = x] \geq \frac{m^b}{2^n}$. (23) follows from the fact that I_m must have size at least $2^{c-n/b}$, as there are $2^{n/b}$ sets and there are at least 2^c elements in the support of X.

As $\mathbf{H}_\infty(X_{|E}) = \max_x \Pr[X = x|E]$, we can give a lower bound for the min entropy of $X_{|E}$ by upper bounding $\Pr[X = x|E]$. More precisely,

$$\begin{aligned}
\Pr[X = x|E] &= \frac{\Pr[X = x \wedge E]}{\Pr[E]} \\
&< \frac{(m+1)^b/2^n}{2^{(c-n/b)}m^b/2^n} \qquad (24) \\
&= \left(1 + \frac{1}{m}\right)^b 2^{-c+n/b} \\
&\leq 2^{b-c+n/b} \qquad (25)
\end{aligned}$$

Inequality (24) uses (23) and the fact that $\Pr[X = x \wedge E] < \frac{(m+1)^b}{2^n}$ by definition of I_m. (25) follows from $m \geq 1$. This implies that $\mathbf{H}_\infty(X_{|E}) > c - n/b - b$ as required in the lemma.

For the second requirement, it is easy to see from the definition of E that the support of the conditional probability distribution $X_{|\overline{E}}$ decreases by at least $2^{(c-n/b)}$ points (as these points belong to E). Clearly, $|\mathsf{sup}(X_{|\overline{E}})| \leq |\mathsf{sup}(X)| - 2^{c-n/b} < |\mathsf{sup}(X)|$ as stated in the lemma.

Now, we observe that, the loss in min-entropy, given by $(b + n/b)$ is minimum when $b = \sqrt{n}$. Since b is a free parameter, we fix $b := \sqrt{n}$ (note that, since $c > \sqrt{n}$, the constraint $c > n/b$ holds) to get $\mathbf{H}_\infty(X_{|E}) > n - 2\sqrt{n}$ as stated in the lemma.

□

In the following lemma we consider an arbitrary distribution X with sufficiently high min-entropy and some arbitrary function T. We show that if the support of $Y = T(X)$ is sufficiently large, then there exists an event E such that one of the following happens:

(i) The min-entropy of Y conditioned on the event E is high, i.e., Y conditioned on E has an almost flat area with large support;

(ii) If $\overline{E}$ happens, then the average min-entropy of X given Y is high. Intuitively, this means that Y conditioned on $\overline{E}$ has small support as then it does not "reveal" too much about X.

We formalize this statement in the lemma below.

Lemma D.2. *For $n \in \mathbb{N}_{>1}$ let c, α be some parameters such that $\sqrt{n} < c < \alpha \leq n$. Let $\mathcal{X}$ be some set of size $2^n = |\mathcal{X}|$ and X be an (α, n)-good distribution over $\mathcal{X}$. For any function $T : \mathcal{X} \to \mathcal{X}$, let $Y = T(X)$ be such that $|\mathsf{sup}(Y)| > 2^c$. There exists an event E such that the following holds:*

(i) $\mathbf{H}_\infty(Y_{|E}) > c - 2\sqrt{n}$.

(ii) $\widetilde{\mathbf{H}}_\infty(X_{|\overline{E}}|Y_{|\overline{E}}) \geq \alpha - c - \log \frac{1}{1-\Pr[E]}$.

Proof. Intuitively, in the proof below we apply Lemma D.1 iteratively to the distribution Y to find flat areas in Y. We "cut off" these flat areas until we have a distribution (derived from Y) which has sufficiently small support. Clearly such restricted Y cannot reveal too much information about X. To formalize this approach, we construct iteratively an event E by combining the events E_i obtained by applying Lemma D.1 to Y. If E happens then Y takes values that lie in a large flat area. On the other hand $\overline{E}$ characterizes only a relatively small support, and hence giving such Y does

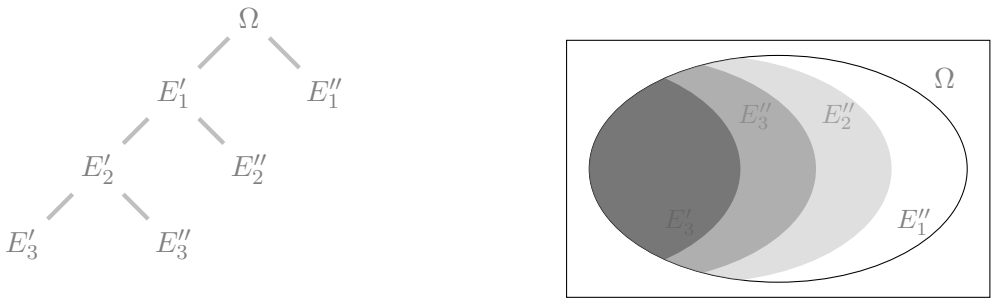

Figure 7: Events covering the probability space in the proof of Lemma D.2 and Lemma 3.2.

not reveal much information (on average) about X. The formal proof with an explicit calculation of the parameters follows. We will define the event E depending on events $\{E_i, E'_i, E''_i\}_{i\in\{0,\dots,m-1\}}$ (for some integer m) which we will specify later. These events partition the probability space as follows (cf. Figure 7):

$$E'_i := \bigwedge_{j=0}^{i} \overline{E}_j = \overline{E}_i \wedge E'_{i-1} \qquad E''_i := E_i \wedge \left(\bigwedge_{j=0}^{i-1} \overline{E}_j\right) = E_i \wedge E'_{i-1}. \tag{26}$$

We will rely on some properties of the above partition. In particular, note that for all $i \in \{0,\dots,m-1\}$ we have

$$E'_i \vee E''_i = E'_{i-1} \qquad E'_i \wedge E''_i = \emptyset. \tag{27}$$

We start by constructing the events $\{E_i, E'_i, E''_i\}$ and conditional probability distributions $Y^{(i)}$ that are derived from Y by applying Lemma D.1. Lemma D.1 requires the following two conditions:

1. $|\mathsf{sup}(Y^{(i)})| > 2^c$, and
2. for all $y \in \mathsf{sup}(Y^{(i)})$ we have $\Pr[Y^{(i)} = y] \geq 2^{-n}$.

Clearly these two conditions are satisfied by $Y^{(0)} = Y$, since $Y^{(0)}$ is computed from X by applying a function T and for all $x \in \mathsf{sup}(X)$ the statement assumes $\Pr[X = x] \geq 2^{-n}$. Hence, Lemma D.1 gives us an event E_0. We set and we define $Y^{(1)} = Y^{(0)}_{|\overline{E}_0}$. For all $i \geq 1$ we proceed to construct events E_i and conditional distributions $Y^{(i+1)} = Y^{(i)}_{|\overline{E}_i}$ as long as the requirements from above in 1 and 2 are satisfied. Notice that by applying Lemma D.1 to distribution $Y^{(i)}$ we get for each event E_i:

1. $\mathbf{H}_\infty(Y^{(i)}_{|E_i}) > c - 2\sqrt{n}$, and
2. $|\mathsf{sup}(Y^{(i+1)})| < |\mathsf{sup}(Y^{(i)})|$.

Clearly, there are only finitely many (say m) events before we stop the iteration as the size of the support is strictly decreasing. At the stopping point we have $|\mathsf{sup}(Y^{(m-1)})| > 2^c$ and $|\mathsf{sup}(Y^{(m)})| \leq 2^c$. We define $E = \bigvee_{i=0}^{m-1} E_i = \bigvee_{i=0}^{m-1} E''_i$ and $\overline{E} = \bigwedge_{i=0}^{m-1} \overline{E}_i = E'_{m-1}$ and show in the claims below that they satisfy the conditions of the lemma.

Claim 7. $\mathbf{H}_\infty(Y_{|E}) > c - 2\sqrt{n}$.

Proof. Recall that for each $0 \leq i \leq m-1$ we have

$$Y^{(i)}_{|E_i} = Y_{|E_i \wedge \overline{E}_{i-1} \ldots \wedge \overline{E}_0} \tag{28}$$

$$= Y_{|E''_i} \tag{29}$$

Eq. (28) follows from the definition of the conditional probability distribution $Y^{(i)}_{|E_i}$. Eq. (29) from the definition of the constructed events. From Eq. (29) and Lemma D.1 we have for each $0 \leq i \leq m-1$ that $\mathbf{H}_\infty(Y_{|E''_i}) > c - 2\sqrt{n}$. As for each $0 \leq i \leq m-1$ we have $|\mathsf{sup}(Y_{|E}) \geq |\mathsf{sup}(Y_{|E''_i})|$ we get that $\mathbf{H}_\infty(Y_{|E}) > c - 2\sqrt{n}$. This concludes the proof of this claim. □

Claim 8. $\widetilde{\mathbf{H}}_\infty(X_{|\overline{E}}|Y_{|\overline{E}}) \geq \alpha - c - \log \frac{1}{1-\Pr[E]}$.

Proof. We first lower bound $\mathbf{H}_\infty(X_{|\overline{E}})$.

$$\mathbf{H}_\infty(X_{|\overline{E}}) = -\log\left(\max_x \frac{\Pr[X = x \wedge \overline{E}]}{\Pr[\overline{E}]}\right) \tag{30}$$

$$\geq -\log\left(\frac{1}{\Pr[\overline{E}]} \max_x \Pr[X = x]\right) \tag{31}$$

$$\geq \mathbf{H}_\infty(X) - \log \frac{1}{\Pr[\overline{E}]} \geq \alpha - \log \frac{1}{1-\Pr[E]}. \tag{32}$$

Eq. (30) follows from the definition of min-entropy and the definition of conditional probability. Eq. (31) follows from the basic fact that for two event $\Pr[E \wedge E'] \leq \Pr[E]$. Finally, we get Eq. (32) from our assumption that $\mathbf{H}_\infty(X) \geq \alpha$. To conclude the claim we compute:

$$\widetilde{\mathbf{H}}_\infty(X_{|\overline{E}}|Y_{|\overline{E}}) \geq \mathbf{H}_\infty(X_{|\overline{E}}, Y_{|\overline{E}}) - \log|\mathsf{sup}(Y_{|\overline{E}})| \tag{33}$$

$$= \mathbf{H}_\infty(X_{|\overline{E}}) - \log|\mathsf{sup}(Y_{|\overline{E}})| \tag{34}$$

$$\geq \alpha - \log \frac{1}{1-\Pr[E]} - c = \alpha - c - \log \frac{1}{1-\Pr[E]}. \tag{35}$$

Eq. (33) follows from Lemma 2.1 and (34) from the fact that $Y_{|\overline{E}}$ is computed as a function from $X_{|\overline{E}}$. Inequality (35) follows from (32) and the fact that the size of $\mathsf{sup}(Y_{|\overline{E}})$ is at most c. The latter follows from the definition of the event $\overline{E} = E'_{m-1}$ which in turn implies that $|\mathsf{sup}(Y_{|\overline{E}})| = |\mathsf{sup}(Y_{|E'_{m-1}})| = |\mathsf{sup}(Y^{(m-1)}_{|\overline{E}_{m-1}})| = |\mathsf{sup}(Y^{(m)})| \leq 2^c$, which concludes the proof. □

The above two claims finish the proof. □

We now turn to the proof of the chaining lemma.

Lemma 3.2. *For $n \in \mathbb{N}_{>1}$ let $\alpha, \beta, t, \epsilon$ be some parameters where $t \in \mathbb{N}$, $0 < \alpha \leq n$, $\beta > 0$, $\epsilon \in (0,1]$ and $t \leq \frac{\alpha-\beta}{\beta+2\sqrt{n}}$. Let $\mathcal{X}$ be some set of size $|\mathcal{X}| = 2^n$ and let $X^{(0)}$ be a (α, n)-good distribution over $\mathcal{X}$. For $i \in [t]$ let $T_i : \mathcal{X} \to \mathcal{X}$ be arbitrary functions and $X^{(i)} = T_i(X^{(i-1)})$. There exists an event E such that:*

(i) If $\Pr[E] > 0$, for all $i \in [t]$, $\mathbf{H}_\infty(X^{(i)}_{|E}) \geq \beta$.

(ii) If $\Pr\left[\overline{E}\right] \geq \epsilon$ *there exists an index* $j \in [t]$ *such that*

$$\widetilde{\mathbf{H}}_\infty(X^{(j-1)}_{|\overline{E}}|X^{(j)}_{|\overline{E}}) \geq \beta - \log\frac{t}{\epsilon}.$$

Proof of Lemma 3.2. Consider the chain of random variables $X^{(0)} \xrightarrow{T_1} X^{(1)} \xrightarrow{T_2} \ldots \xrightarrow{T_t} X^{(t)}$. Given a pair of random variables in the chain, we refer to $X^{(i-1)}$ as the "source distribution" and to $X^{(i)}$ as the "target distribution". The main idea is to consider different cases depending on the characteristics of the target distribution. In case the min-entropy of $X^{(i)}$ is high enough to start with, we get immediately property (i) of the statement and we can immediately move to the next pair of random variables in the chain. In case the min-entropy of $X^{(i)}$ is small, we further consider two different sub-cases depending on some bound on the support of the variable. If the support of $X^{(i)}$ happens to be "small", intuitively we can condition on the target distribution since this cannot reveal much about the source; roughly this implies property (ii) of the statement. On the other hand, if the support happens to be not small enough, we are not in a position which allows us to condition on $X^{(i)}$.

In the latter case, we will invoke Lemma D.2. Roughly this guarantees that there exists some event such that, conditioned on this event happening, the target lies in a large "flat" area and the conditional distribution has high min-entropy; this yields property (i) of the statement. If instead the event does not happen, then conditioning on the event not happening we get a "restricted" distribution with small enough support which leads again to property (ii) of the second statement.

Whenever we are in those cases where (possibly conditioning on some event) the target distribution has high min-entropy, we move forward in the chain by considering $X^{(i)}$ as the source and $X^{(i+1)}$ as the target. However, when we reach a situation where we can "reveal" the target distribution we do not proceed further, since the remaining values can be computed as a deterministic function of the revealed distribution and, as such, do not constrain the min-entropy further. We now proceed with the formal proof.

Similar to Lemma D.2, we will define the event E depending on events $\{E_i, E'_i, E''_i\}_{i\in[t]}$ which we will specify later. These events partition the probability space as follows (cf. Figure 7):

$$E'_i := \bigwedge_{j=1}^{i} E_j = E_i \wedge E'_{i-1} \qquad\qquad E''_i := \overline{E}_i \wedge \left(\bigwedge_{j=1}^{i-1} E_j\right) = \overline{E}_i \wedge E'_{i-1}. \tag{36}$$

We will rely on some properties of the above partition. In particular, note that for all $i \in [t]$ we have

$$E'_i \vee E''_i = E'_{i-1} \qquad\qquad E'_i \wedge E''_i = \emptyset. \tag{37}$$

For all $i \in [t+1]$, define the following parameters:

$$s_i = (t - i + 1)(\beta + 2\sqrt{n}) \tag{38}$$

$$\alpha_{i-1} = \beta + s_i. \tag{39}$$

Note that using the bound on t from the statement of the lemma, we get $\alpha \geq \alpha_0$; moreover, it is easy to verify that $\alpha_{i-1} > s_i > \sqrt{n}$ for all $i \in [t]$.

In the next claim we construct the events $\{E_i, E'_i, E''_i\}_{i\in[t]}$.

Claim 9. *For all* $i = 0, \ldots, t-1$*, there exist events* E'_{i+1} *and* E''_{i+1} *(as given in Eq.* (37)*) such that the following hold:*

(*) If $\Pr\left[E'_{i+1}\right] > 0$, $\mathbf{H}_\infty(X^{(i+1)}_{|E'_{i+1}}) \geq \alpha_{i+1}$.

(**) If $\Pr\left[E''_{i+1}\right] \geq \epsilon'$, $\widetilde{\mathbf{H}}_\infty(X^{(i)}_{|E''_{i+1}}|X^{(i+1)}_{|E''_{i+1}}) \geq \beta - \log\frac{1}{\epsilon'}$. *where* $0 < \epsilon' \leq 1$.

Proof. We prove the claim by induction.

Base Case: In this case we let E_0 denote the whole probability space and thus $\Pr[E_0] = 1$. note that $\mathbf{H}_\infty(X^{(0)}_{|E_0}) = \mathbf{H}_\infty(X^{(0)}) = \alpha \geq \alpha_0$. The rest of the proof for the base case is almost the same to that of the inductive step except the use of above statement instead of induction hypothesis. Therefore we only prove the induction step in detail here. The proof-details for the base case would be a straightforward extension of the induction step with some notational changes.

Induction Step: The following holds by *induction hypothesis*:

(*) If $\Pr[E'_i] > 0$, then $\mathbf{H}_\infty(X^{(i)}_{|E'_i}) \geq \alpha_i$.

(**) If $\Pr[E''_i] \geq \epsilon'$ then, $\widetilde{\mathbf{H}}_\infty(X^{(i-1)}_{|E''_i}|X^{(i)}_{|E''_i}) \geq \beta - \log\frac{1}{\epsilon'}$ where $0 < \epsilon' \leq 1$.

By construction of events, E'_i is partitioned into two sub-events E'_{i+1} and E''_{i+1} (cf. Eq. 37). From the statement of the claim, we observe that: since we are assuming $\Pr\left[E'_{i+1}\right] > 0$ in (*) and $\Pr\left[E''_{i+1}\right] \geq \epsilon' > 0$ in (**), in both cases we have $\Pr[E'_i] > 0$. Then (*) from the induction hypothesis holds: $\mathbf{H}_\infty(X^{(i)}_{|E'_i}) \geq \alpha_i$ which we use to prove the inductive step. We will define the events E'_{i+1} and E''_{i+1} differently depending on several (complete) cases. For each of these cases we will show that property (*) and (**) hold.

The case where $\mathbf{H}_\infty(X^{(i+1)}_{|E'_i}) \geq \alpha_{i+1}$. In this case we define E'_{i+1} to be E'_i, which implies $E''_{i+1} = \emptyset$ by Eq. (37). Moreover property (*) holds since, if $\Pr\left[E'_{i+1}\right] > 0$, then $\Pr[E'_i] > 0$ and $\mathbf{H}_\infty(X^{(i+1)}_{|E'_{i+1}}) = \mathbf{H}_\infty(X^{(i+1)}_{|E'_i}) \geq \alpha_{i+1}$; as for property (**) there is nothing to prove, since $\Pr\left[E''_{i+1}\right] = 0$ in this case.

The case where $\mathbf{H}_\infty(X^{(i+1)}_{|E'_i}) < \alpha_{i+1}$. Here we consider two sub-cases, depending on the support size of $X^{(i+1)}$.

1. $|\mathsf{sup}(X^{(i+1)}_{|E'_i})| \leq 2^{s_{i+1}}$. We define $E''_{i+1} = E'_i$, which implies $E'_{i+1} = \emptyset$ by Eq. (37). As for property (*) there is nothing to prove, since $\Pr\left[E'_{i+1}\right] = 0$. To prove property (**) we observe the following:
   If $\Pr\left[E''_{i+1}\right] \geq \epsilon' > 0$, then $\Pr[E'_i] > 0$. Hence,

$$\begin{aligned}
\widetilde{\mathbf{H}}_\infty(X^{(i)}_{|E''_{i+1}}|X^{(i+1)}_{|E''_{i+1}}) &= \widetilde{\mathbf{H}}_\infty(X^{(i)}_{|E'_i}|X^{(i+1)}_{|E'_i}) && (40)\\
&\geq \mathbf{H}_\infty(X^{(i)}_{|E'_i}, X^{(i+1)}_{|E'_i}) - \log(|\mathsf{sup}(X^{(i+1)}_{|E'_i})|) && (41)\\
&\geq \alpha_i - s_{i+1} && (42)\\
&= \beta + s_{i+1} - s_{i+1} = \beta.
\end{aligned}$$

Eq. (40) follows as $E''_{i+1} = E'_i$. Eq. (41) follows from Lemma 2.1. Eq. (42) follows from two facts: (i) $X^{(i+1)}$ is a deterministic function of $X^{(i)}$, which means $\mathbf{H}_\infty(X^{(i+1)}_{|E'_i}, X^{(i+1)}_{|E'_i}) = \mathbf{H}_\infty(X^{(i)}_{|E'_i}) \geq \alpha_i$ (plugging-in the value from induction hypothesis), and (ii) $|\mathsf{sup}(X^{(i+1)}_{|E'_i})| \leq 2^{s_{i+1}}$.

2. $|\mathsf{sup}(X^{(i+1)}_{|E'_i})| > 2^{s_{i+1}}$. Here using induction hypothesis: $\mathbf{H}_\infty(X^{(i)}_{|E'_i}) \geq \alpha_i$, we invoke Lemma D.2 on the distribution $X^{(i+1)}_{|E'_i}$ (recall that $\alpha_i > s_{i+1} > \sqrt{n}$), to obtain the event E_{i+1} such that:

$$\mathbf{H}_\infty(X^{(i+1)}_{|E'_i \wedge E_{i+1}}) > s_{i+1} - 2\sqrt{n} \tag{43}$$

$$\widetilde{\mathbf{H}}_\infty(X^{(i)}_{|E'_i \wedge \overline{E}_{i+1}} | X^{(i+1)}_{|E'_i \wedge \overline{E}_{i+1}}) > \alpha_i - s_{i+1} - \log \frac{1}{1 - \Pr[E_{i+1}]}. \tag{44}$$

Note that by our definitions of the events E'_i, E''_i (cf. Eq. (36)), we have $E'_i \wedge E_{i+1} = E'_{i+1}$ and $E'_i \wedge \overline{E}_{i+1} = E''_{i+1}$.

Now, to prove (*) we have: if $\Pr\left[E'_{i+1}\right] > 0$, then $\Pr[E'_i] > 0$ and $\Pr[E_{i+1}] > 0$. Plugging the values of α_i and s_{i+1} from Eq. (39) and (38) into Eq. (43), we get

$$\begin{aligned}
\mathbf{H}_\infty(X^{(i+1)}_{|E'_{i+1}}) &> s_{i+1} - 2\sqrt{n} \\
&= (t-i)(\beta + 2\sqrt{n}) - 2\sqrt{n} \\
&= \beta + (t-i-1)(\beta + 2\sqrt{n}) \\
&= \beta + s_{i+2} = \alpha_{i+1},
\end{aligned}$$

Similarly, to prove (**) we have: if $\Pr\left[E''_{i+1}\right] \geq \epsilon'$, then $\Pr[E'_i] \geq \epsilon' > 0$ and $\Pr\left[\overline{E}_{i+1}\right] \geq \epsilon'$. Using Eq. (44), we obtain:

$$\begin{aligned}
\widetilde{\mathbf{H}}_\infty(X^{(i)}_{|E''_{i+1}} | X^{(i+1)}_{|E''_{i+1}}) &> \alpha_i - s_{i+1} - \log \frac{1}{\Pr\left[\overline{E}_{i+1}\right]} \\
&= \beta - \log \frac{1}{\Pr\left[\overline{E}_{i+1}\right]} \\
&\geq \beta - \log \frac{1}{\epsilon'},
\end{aligned}$$

This concludes the proof of the claim. □

We define the event E to be $E = E'_t = \bigwedge_{i=1}^t E_i = \bigwedge_{i=1}^t E'_i$. It is easy to verify that this implies $\overline{E} = \bigvee_{i=1}^t E''_i$. We distinguish two cases:

- If $\Pr[E] > 0$, by definition of E we get that $\Pr[E'_i] > 0$ for all $i \in [t]$. In particular, $\Pr[E'_t] > 0$. Hence, $\mathbf{H}_\infty(X^{(t)}_{|E}) = \mathbf{H}_\infty(X^{(t)}_{|E'_t}) \geq \alpha_t = \beta$, where the last inequality follows from (*) of Claim 9 putting $i = t-1$. Also, we observe that for all $i \in [t]$, $\mathbf{H}_\infty(X^{(i-1)}_{|E}) \geq \mathbf{H}_\infty(X^{(i)}_{|E})$. This proves property (i) of the lemma.

- If $\Pr\left[\overline{E}\right] \geq \epsilon$, then we get

$$\Pr\left[\bigvee_{i=1}^{t} E_i''\right] \geq \epsilon. \tag{45}$$

$$\sum_{i=1}^{t} \Pr\left[E_i''\right] \geq \epsilon. \tag{46}$$

Eq. 45 follows from the definition of E and Eq. 46 follows applying union bound. Clearly, from Eq. 46, there must exists some j such that $\Pr\left[E_j''\right] \geq \epsilon/t$.

Hence, putting $i = j - 1$ and $\epsilon' = \epsilon/t$ in (**) of Claim 9, we get:

$$\widetilde{\mathbf{H}}_\infty(X^{(j-1)}_{|E_j''} | X^{(j)}_{|E_j''}) \geq \beta - \log \frac{t}{\epsilon}.$$

From the definition of E, E_j'' implies $\overline{E}$ and hence property (ii) of the lemma follows.

□