An extended abstract of this paper is published in the proceedings of the 19th Annual International Conference on the Theory and Application of Cryptology and Information Security — Asiacrypt 2013. This is the full version.

Bounded Tamper Resilience: How to go beyond the Algebraic Barrier

Ivan Damgård[1], Sebastian Faust[2], Pratyay Mukherjee[1], and Daniele Venturi[1]

[1]*Department of Computer Science, Aarhus University*
[2]*Security and Cryptography Laboratory, EPFL*

October 23, 2013

Abstract

Related key attacks (RKAs) are powerful cryptanalytic attacks where an adversary can change the secret key and observe the effect of such changes at the output. The state of the art in RKA security protects against an a-priori unbounded number of certain algebraic induced key relations, e.g., affine functions or polynomials of bounded degree. In this work, we show that it is possible to go beyond the algebraic barrier and achieve security against *arbitrary* key relations, by restricting the number of tampering queries the adversary is allowed to ask for. The latter restriction is necessary in case of arbitrary key relations, as otherwise a generic attack of Gennaro *et al.* (TCC 2004) shows how to recover the key of almost any cryptographic primitive. We describe our contributions in more detail below.

1. 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 obtain some bounded amount of leakage. Interestingly, for the Okamoto scheme we can allow also independent tampering with the public parameters.
2. 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.
3. Finally, we explain how to boost bounded tampering and leakage resilience (as in 1. and 2. above) to *continuous* tampering and leakage resilience, in the so-called *floppy model* where each user has a personal hardware token (containing leak- and tamper-free information) which can be used to refresh the secret key.

We believe that bounded tampering is a meaningful and interesting alternative to avoid known impossibility results and can provide important insights into the security of existing standard cryptographic schemes.

Keywords: related key security, bounded tamper resilience, public key encryption, identification schemes

Contents

1 Introduction

Related key attacks (RKAs) are powerful cryptanalytic attacks against a cryptographic implementation that allow an adversary to change the key, and subsequently observe the effect of such modification on the output. In practice, such attacks can be carried out, e.g., by heating up the device or altering the internal power supply or clock [4, 11], and may have severe consequences for the security of a cryptographic implementation. 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 schemes.

Bellare and Kohno [8] pioneered the formal security analysis of cryptographic schemes in the presence of related key attacks. In their setting an adversary tampers *continuously* with the key by applying functions T chosen from a set of *admissible* tampering functions $\mathcal{T}$. In the signature example from above, each signing query for message m would be accompanied with a tampering function $T \in \mathcal{T}$ and the adversary obtains $\mathsf{Sign}(T(sk), m)$. Clearly, a result in the RKA setting is stronger if the class of admissible functions $\mathcal{T}$ is larger, and hence several recent works have focussed on further broadening $\mathcal{T}$. The current state of the art (see discussion in Section 1.2) considers certain algebraic relations of the key, e.g., $\mathcal{T}$ is the set of all affine functions or all polynomials of bounded degree. A natural question that arises from these works is if we can further broaden the class of tampering functions — possibly showing security for *arbitrary* relations. In this work, we study this question and show that under certain assumptions security against arbitrary key relations can be achieved.

Is arbitrary key tampering possible? Unfortunately, the answer to the above question in its most general form is negative. As shown by Gennaro *et al.* [26], it is *impossible* to protect any cryptographic scheme against arbitrary key relations. In particular, there is an attack that allows to recover the secret key of most stateless cryptographic primitives after only a few number of tampering queries.[1] To prevent this attack the authors propose to use a *self-destruct* mechanism. That is, before each execution of the cryptographic scheme the key is checked for its validity. In case the key was changed the device self-destructs. In practice, such self-destruct can for instance be implemented by overwriting the secret key with the all-zero string, or by switching to a special mode in which the device outputs $\perp$.[2] In this work, we consider an alternative setting to avoid the impossibility results of [26], and assume that an adversary can only carry out a bounded number of (say t) tampering queries. To explain our setting consider again the example of a digital signature scheme. In our model, we give the adversary access to t tampered signing oracles $\mathsf{Sign}(T_i(sk), \cdot)$, where T_i can be an arbitrary adaptively chosen tampering function. Notice that of course each of these oracles can be queried a polynomial number of times, while t is typically linear in the security parameter.

Is security against bounded tampering useful? Besides from being a natural and non-trivial security notion, we believe that our adversarial model of *arbitrary*, *bounded* tampering is useful for a number of reasons:

1. It is a natural alternative to continuous restricted tampering: our security notion of *bounded*, *arbitrary* tampering is orthogonal to the traditional setting of RKA security where the adversary can tamper *continuously* but is *restricted* to certain classes of attacks. Most previous work in the RKA setting considers algebraic key relations that are tied to the scheme's algebra and may

[1] The impossibility result of [26] leaves certain loopholes, which however seem very hard to exploit.

[2] We notice that the self-destruct has to be permanent as otherwise the attack of [26] may still apply.

Tampering Model	ID Schemes		IND-CCA PKE
	Σ-Protocols	Okamoto	BHHO
Secret Key	✓	✓	✓
Public Parameters	n.a.	✓	n.a.
Continuous Tampering *i*Floppy	✓	✓	✓
Key Length	$\log \lvert\mathcal{X}\rvert$	$\ell \log p$	$\ell \log p$
Tampering Queries	$\lfloor \log \lvert\mathcal{X}\rvert / \log \lvert\mathcal{Y}\rvert \rfloor - 2$	$\ell - 2$	$\ell - 3$

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 3).

not reflect attacks in practice. For instance, it is not clear that heating up the device or shooting with a laser on the memory can be described by, e.g., an affine function — a class that is usually considered in the literature. We also notice that physical tampering may completely destroy the device, or may be detected by hardware countermeasures, and hence our model of bounded but arbitrary tampering may be sufficient in such settings.

2. It allows to analyze the security of *standard* cryptoschemes: as outlined above a common countermeasure to protect against arbitrary tampering is to implement a key validity check and self-destruct (or output a special failure symbol) in case such check fails. Unfortunately, most standard cryptographic implementations do not come with such a built-in procedure to check the validity of the key. Our notion of bounded tamper resilience allows to make formal security guarantees of *standard* cryptographic schemes where neither the construction, nor the implementation needs to be specially engineered.

3. It can be a useful as a building-block: even if the restriction of bounded tamper resilience may be too strong in some settings, it can be useful to achieve results in the stronger continuous tampering setting (we provide some first preliminary results on this in Appendix A). Notice that this is similar to the setting of leakage resilient cryptography which also started mainly with "bounded leakage" that later turned out to be very useful to get results in the continuous leakage setting.

We believe that due to the above points the bounded tampering model is an interesting alternative to avoid known impossibility results for arbitrary tampering attacks.

1.1 Our Contribution

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. While our general techniques use ideas from the leakage realm, we emphasize that bounded leakage resilience does *not* imply bounded tamper resilience. In fact, it is easy to find contrived schemes that are leakage resilient but completely break for a single tampering query. At a more technical level, we observe that a trivial strategy using leakage to simulate, e.g., faulty signatures, has to fail as the adversary can get any polynomial number of faulty signatures — which clearly cannot be simulated with bounded leakage only. Nevertheless, as we show in this work, we are able to identify certain classes of cryptoschemes for which a small amount of leakage is sufficient to simulate faulty outputs. We discuss this in more detail below.

Our concrete schemes are proven secure under standard assumptions (DL, factoring or DDH) and are efficient and simple. Moreover, we show that our schemes 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 1.1 for an overview of our results). Importantly, all our results allow arbitrary key tampering and do not need any kind of tamper detection mechanism.

Identification schemes. It is well known that the Generalized Okamoto identification scheme [36] provides security against bounded leakage from the secret key [3, 32]. In Section 3, 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 any number of 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 show that Σ-protocols where the secret key is significantly longer than the public key are BLT secure. We can instantiate our result with the generalized Guillou-Quisquater ID scheme [29], and its variant based on factoring [25] yielding tamper resilient identification based on factoring. We give more details in Section 3.

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 restrictions are: (i) tampering with the public parameters is independent from tampering with the secret key and (ii) the tampering with public parameters must map to its domain. We also show that the latter restrictions are necessary, by presenting explicit attacks when the adversary can tamper jointly with the secret key and the public parameters or he can tamper the public parameters to some particular range.

Public key encryption. We show how to construct IND-CCA secure public key encryption (PKE) 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 already provides a basic level of tamper resilience for public key encryption schemes. Consider for instance a setting where the adversary can tamper with the decryption key, but has no control over the ciphertexts that are sent to the decryption oracle, e.g., the ciphertexts are sent over a secure authenticated channel.

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) and the randomness. We show how to instantiate our basic tamper security notion under DDH. More concretely, we prove that the BHHO cryptosystem [12] 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 earlier work [31]. 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 when we consider arbitrary tampering. 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.* [31]. 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.* [3] and recently studied in depth by Agrawal *et al.* [2]. In the floppy model a user of a cryptodevice possesses a so-called *floppy* – a secure hardware token – that stores an update key.[3] 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 [2, 3] all users can potentially possess an identical hardware token, 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 hardware tokens, but on the other hand, in practice one would not want to distribute hardware tokens that all contain the same global update key as this constitutes a single point of failure.

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 [2]). 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 security on standard assumptions. Our results in the *i*Floppy model are mainly deferred to Appendix A.

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 [15, 20, 3] 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 (see Appendix B). 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.

1.2 Previous Work

Related key security. We already discussed the relation between BLT security and the traditional notion of RKA security above. Below we give further details on some important results in the RKA area. Bellare and Kohno [8] initiated the theoretical study of related-key-attacks. Their result mainly focused on symmetric key primitives (e.g. PRP, PRF). They proposed various block-cipher based constructions which are RKA-secure against certain restricted classes of tampering functions. Their constructions were further improved by [34, 6]. Following these works other cryptographic primitives were constructed that are provably secure against certain classes of related key attacks. Most of these works consider rather restricted tampering functions that, e.g., can be described by a linear or affine function [8, 34, 6, 5, 37, 39, 10]. A few important exceptions are described below.

In [9] the authors show how to go beyond the linear barrier by extending the class of allowed tampering functions to the class of polynomials of bounded degree for a number of public-key primitives.

[3] Notice that "floppy" is just terminology and we use it for consistency with earlier works.

Also, the work of Goyal, O'Neill and Rao [27] considers polynomial relations that are induced to the inputs of a hash function. Finally Bellare, Cash and Miller [7] develop a framework to transfer RKA security from a pseudorandom function to other primitives (including many public key primitives).

Tamper resilient encodings. A generic method for tamper protection has been put forward by Gennaro *et al.* [26]. 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'. In their construction the original state is signed and the signature is checked before each usage. 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. This drawback is resolved by the concept of non-malleable codes pioneered by Dziembowski, Pietrzak and Wichs [22]. The original construction of [22] considers an adversary that can tamper independently with bits. This has been extended to small size blocks in [13], and recently to so-called split-state tampering [33, 1]. While the above schemes provide surprisingly strong security guarantees, they all require certain assumptions on the hardware (e.g., the memory has to be split into two parts that cannot tampered with jointly), and require significant changes to the implementation for decoding, tamper detection and self-destruct.

Continuous tamper resilience via key updates. Kalai *et al.* [31] provide first feasibility results in the so-called *continuous leakage and tampering* model (CLT). Their constructions achieve strong security requirements where the adversary can arbitrarily tamper continuously with the state. This is achieved by updating the secret key after each usage. While the tampering adversary considered in [31] is clearly stronger (continuous as opposed to bounded tampering), the proposed schemes are non-standard, rather inefficient and rely on non-standard assumptions. Moreover, the approach of key updates requires a stateful device and large amounts of randomness which is costly in practice. The main focus of this work, are simple standard cryptosystems that neither require randomness for key updates nor need to keep state.

Tampering with computation. In all the above works (including ours) it is assumed that the circuitry that computes the cryptographic algorithm using the potentially tampered key runs correctly and is not subject to tampering attacks. An important line of works analyze to what extend we can guarantee security when the complete circuitry is prone to tampering attacks [30, 23, 16]. These works typically consider a restricted class of tampering attacks (e.g., individual bit tampering) and assume that large parts of the circuit (and memory) remain un-tampered.

2 Preliminaries

2.1 Basic Notation

We review the basic terminology used throughout the paper. 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 $(a^{x_1}, \ldots, a^{x_\ell})$.

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.

Languages and relations. 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 $y \in \mathfrak{L}$.

Information theory. 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 [19] of X given a random variable Z (over a set $\mathcal{Z}$) possibly dependent on X, is defined as $\widetilde{\mathbf{H}}_\infty(X|Z) := -\log \mathbb{E}_{z \leftarrow Z}[2^{-\mathbf{H}_\infty(X|Z=z)}]$. Following [3], we sometimes 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.

Definition 2.1 ([3])**.** 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 [19, 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*

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

2.2 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 [3, 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

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

2.3 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.

2.4 Σ-protocols

Definition 2.2 (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 2.2 requires *perfect* HVZK, whereas in general one could ask for a weaker requirement, namely that the HVZK property holds only computationally.

2.5 True Simulation Extractibility

We recall the notion of true-simulation extractable (tSE) NIZKs [18]. This notion is similar to the notion of simulation-sound extractable NIZKs [28], 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 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 3.1. 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 $\mathsf{P}(pp, sk)$ 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 $(\widetilde{sk}_i, \widetilde{pp}_i) = T_i(sk, pp)$. 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 [3] this is reminiscent of so-called *anytime leakage* attacks. Our results can be generalized to this setting, however we stick to Definition 3.1 for simplicity.

The rest of this section is organized as follows. In Section 3.1 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 3.2 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 3.3 we illustrate that the latter assumption is necessary, as otherwise the Okamoto ID scheme can be broken by (albeit contrived) attacks.

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

3.1 Σ-protocols are Tamper Resilient

We start by considering ID schemes based on Σ-protocols [14]. 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 can be found in Section 2.4.

It is well known that Σ-protocols are a natural tool to design ID schemes. The construction is depicted in Figure 1. 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). The proof of the following theorem uses ideas of [3], but is carefully adjusted to incorporate tampering attacks.

Theorem 3.1. *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 2.2). 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 1 is $(\lambda(k), t(k), negl(k))$-BLT secure against impersonation attacks with respect to $\mathcal{T}_{\mathsf{sk}}$, where*

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

Proof. Assume that there exists a polynomial $p(.)$ and an adversary A that succeeds in the BLT experiment (cf. Definition 3.1) 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 2.2).

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,

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

Claim 1. *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 [3, Claim 4.1] and is therefore omitted. □

Claim 2. *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 3.1, 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

$$\widetilde{\mathbf{H}}_\infty(X|\mathcal{E}_0) \geq \widetilde{\mathbf{H}}_\infty(X|\mathcal{E}_1). \tag{2}$$

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

$$\widetilde{\mathbf{H}}_\infty(X|\mathcal{E}_1) = \widetilde{\mathbf{H}}_\infty(X|\mathcal{E}_2). \tag{3}$$

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

$$\widetilde{\mathbf{H}}_\infty(X|\mathcal{E}_2) = \widetilde{\mathbf{H}}_\infty(X|\widetilde{Y}_1, \ldots, \widetilde{Y}_t, Y) \geq \widetilde{\mathbf{H}}_\infty(X|Y) - |(\widetilde{Y}_1, \ldots, \widetilde{Y}_t)| \geq \beta - t \log |\mathcal{Y}|, \tag{4}$$

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 3.1 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. (2)–(4) 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. □

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 2: Generalized Okamoto Identification Scheme

Combining Claim 1 and Claim 2 together with Eq. (1) 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. □

3.2 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 [33]), with the key difference that in our case the secret state does not need to be split into two parts.

We model this through 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 [36], depicted in Figure 2. 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 2.2). As proven in [3], this problem is equivalent to the Discrete Log problem in $\mathbb{G}$.

Corollary 1. *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*

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

Proof (Sketch). We first show that the protocol is BLT-secure against impersonation attacks with respect to $\mathcal{T}_{\mathsf{sk}}$. This follows immediately from Theorem 3.1 as the protocol of Figure 2 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 3.1 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 $\widetilde{\mathbf{H}}_\infty(X|\mathcal{E}_1) = \widetilde{\mathbf{H}}_\infty(X|\mathcal{E}_2)$, 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 $a = (\prod_{i=1}^{\ell} \widetilde{g}_i^{z_i})/\widetilde{y}^c \mod \widetilde{p}$. In case $\widetilde{y} = 0 \mod \widetilde{p}$, then just set $a = 0$.[4] 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 $\prod_{i=1}^{\ell} \widetilde{g}_i^{z_i} = a\widetilde{y}^c \mod \widetilde{p}$.

Therefore the simulation perfectly matches the honest conversation. This proves Eq. (3). Now Eq. (4) follows from the fact that the tampering functions T^{pp} cannot depend on sk. □

3.3 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 functions $\mathcal{T}$ introduced in Definition 3.1.

Claim 3. *The generalized Okamoto ID scheme is* not *BLT-secure against impersonation attacks with respect to $\mathcal{T}$.*

Proof. The attack uses 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 [38].)
- Let $\widetilde{g}$ be a generator of $\widetilde{\mathbb{G}}$ (which exists since $\widetilde{p}$ is a prime) and define the new generators as $\widetilde{g}_i = \widetilde{g}^{x_i} \mod \widetilde{p}$.

Consider now a transcript $(a, c, \mathbf{z})$ produced by a run of $\mathsf{P}(\widetilde{pp}, \mathbf{x})$. We have $a = \widetilde{g}^{\sum_{i=1}^{\ell} x_i r_i} \mod \widetilde{p}$ 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 $\sum_{i=1}^{\ell} x_i r_i = \log_{\widetilde{g}}(a) \mod \widetilde{p}$. 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 $x_i \mod p = x_i \mod \widetilde{p}$ since $|p| \approx |\widetilde{p}|$.) □

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

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

Proof. The attack uses 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 3.)
- 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 $\widetilde{x}_1 = (x_1, \ldots, x_\ell)$. □

3.4 BLT-Secure Signatures

It is well known that every Σ-protocol can be turned into a signature scheme via the Fiat-Shamir heuristic [24]. By applying the Fiat-Shamir transformation to the protocol of Figure 1, we get efficient BLT-secure signatures in the random oracle model.

4 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 4.1. 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 $\widetilde{sk}_i = T_i(sk)$. 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 keeps access to $\mathsf{Dec}(sk, \cdot)$ and outputs a bit b'. We say A wins if $b = b'$, $\sum_j \lambda_j \leq \lambda$ and c_b has not been queried for.

In case $t = 0$ we get the notion of leakage resilient IND-CCA from [35] as a special case. Notice that A is not allowed to tamper with the secret key after seeing the challenge ciphertext. As we show in Section 4.4, this restriction is necessary because otherwise A could overwrite the secret key depending on the plaintext encrypted in c_b, and thus gain some advantage in guessing the value of b by asking additional decryption queries.

We build an IND-CCA BLT-secure PKE scheme in two steps. In Section 4.1 we define a weaker notion which we call IND-CPA BLT security. In Section 4.2 we show a general transformation from IND-CPA BLT security to IND-CCA BLT security relying on tSE NIZK proofs [17] 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 4.3, we prove that a variant of the BHHO encryption scheme [35] satisfies our notion of IND-CPA BLT security.

4.1 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 4.1) 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 4.2. 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 4.1 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 $\widetilde{sk}_i = T_i(sk)$. 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.
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 loses access to all oracles and outputs a bit b'. We say that A wins if $b = b'$ and $\sum_j \lambda_j \leq \lambda$.

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 $(\mathsf{Gen}, \mathsf{Prove}, \mathsf{Verify})$ be a tSE NIZK argument system for the 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 3: How to transform IND-CPA BLT-secure PKE into IND-CCA BLT-secure PKE

4.2 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 2.5). The same construction has been already used by Dodis *et al.* [17] 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 4.1. *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 3 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 4.1 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

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

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 $|\Pr\left[\mathsf{A} \text{ wins in } \mathsf{G}_1\right] - \Pr\left[\mathsf{A} \text{ wins in } \mathsf{G}_2\right]| \leq \delta_1(k)$.

Game G_3. 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. Continue to answer normal decryption queries (c, π) from A as above.
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

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

This concludes the proof. □

4.3 Instantiation from BHHO

We show that the variant of the encryption scheme introduced by Boneh *et al.* [12] used in [35] is IND-CPA BLT-secure. The proof relies on the 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 $m = c_1 \cdot g^{-\langle \mathbf{c}_0, \mathbf{x} \rangle}$, where $\langle \mathbf{c}_0, \mathbf{x} \rangle$ denotes the inner product of $\mathbf{c}_0$ and $\mathbf{x}$.

Proposition 4.1. *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*

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

Proof. Naor and Segev [35, 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 [35, 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. □

4.4 Impossibility of "Post-Challenge" IND-CCA BLT Security

Previous definitions of related-key security for IND-CCA PKE allow the adversary to issue tampering queries even after seeing the challenge ciphertext [39, 9]. The reason why the schemes of [39, 9] can achieve this stronger flavour is that the class of tampering functions is too limited to cause any harm. In fact, as we argue below, when the tampering function can be an arbitrary polynomial time function (as is the case in our schemes), no PKE scheme can be secure if such "post-challenge" tampering queries are allowed.

Proposition 4.2. *No one-bit PKE scheme can be "post-challenge" IND-CCA $(0, 1, negl(k))$-BLT secure.*

Proof. We build a polynomial time adversary A breaking IND-CCA BLT security. A will ask a single tampering query after seeing the challenge ciphertext c_b (corresponding to $m_b \in \{0,1\}$) and then make a single decryption query to the tampered decryption oracle, to learn the bit b with probability negligibly close to 1. Given the public key pk and challenge ciphertext c_b, adversary A proceeds as follows:

1. Sample $m^* \in \{0,1\}$ uniformely at random and compute $c^* \leftarrow \mathsf{Enc}(pk, m^*)$.
2. Define the following tampering query $T_{c_b,c^*,m^*}(sk)$:
   - Run $m_b = \mathsf{Dec}(sk, c_b)$. In case $m_b = 0$, let $\widetilde{sk} = sk$.
   - In case $m_b = 1$, sample $(pk^*, sk^*) \leftarrow \mathsf{KGen}(1^k)$ until $\mathsf{Dec}(sk^*, c^*) \neq m^*$. When this happens, let $\widetilde{sk} = sk^*$.
3. Query the decryption oracle $\mathsf{Dec}(\widetilde{sk}, \cdot)$ with c^*. In case the answer from the oracle is m^* output 0 and otherwise output 1.

For the analysis, assume first that A runs in polynomial time. In this case it is easy to see that the attack is successful with overwhelming probability. In fact, $c^* \neq c_b$ with overwhelming probability and the answer from the tampered decryption oracle clearly allows to recover b.

We claim that A runs in expected polynomial time. This is because if one tries to decrypt c^* using an independent freshly generated secret key sk^*, the resulting plaintext will be uncorrelated, up to a small bias, to the plaintext m^*, for otherwise the underlying PKE scheme would not even be IND-CPA secure. (Recall that c^* is an encryption of m^* under the original public-key pk.) This shows that $\Pr[\mathsf{Dec}(sk^*, c^*) \neq m^*] \approx 1/2$ and thus the loop ends on average after 2 attempts.

If one insists on the tampering function being polynomial time (and not expected polynomial time) we can just put an upped bound on the number of pairs (pk^*, sk^*) that the function can sample in the loop. This comes at the expense of a negligible error probability. □

Acknowledgments

Ivan Damgård and Daniele Venturi acknowledge support from the Danish National Research Foundation, the National Science Foundation of China (under the grant 61061130540), the Danish Council for Independent Research (under the DFF Starting Grant 10-081612) and also from the CFEM research center within which part of this work was performed. Sebastian Faust was partially funded by the above grants. Pratyay Mukherjee's work at Aarhus University was supported by a European Research Commission Starting Grant (no. 279447) and the above grants. Part of this work was done while this author was at the University of Warsaw and was supported by the WELCOME/2010-4/2 grant founded within the framework of the EU Innovative Economy Operational Programme.

References

[1] Divesh Aggarwal, Yevgeniy Dodis, and Shachar Lovett. Non-malleable codes from additive combinatorics. *IACR Cryptology ePrint Archive*, 2013:201, 2013.

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

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

[4] 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.

[5] Benny Applebaum, Danny Harnik, and Yuval Ishai. Semantic security under related-key attacks and applications. In *ICS*, pages 45–60, 2011.

[6] Mihir Bellare and David Cash. Pseudorandom functions and permutations provably secure against related-key attacks. In *CRYPTO*, pages 666–684, 2010.

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

[8] Mihir Bellare and Tadayoshi Kohno. A theoretical treatment of related-key attacks: RKA-PRPs, RKA-PRFs, and applications. In *EUROCRYPT*, pages 491–506, 2003.

[9] Mihir Bellare, Kenneth G. Paterson, and Susan Thomson. RKA security beyond the linear barrier: IBE, encryption and signatures. In *ASIACRYPT*, pages 331–348, 2012.

[10] Rishiraj Bhattacharyya and Arnab Roy. Secure message authentication against related key attack. In *FSE*, 2013.

[11] 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.

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

[13] Seung Geol Choi, Aggelos Kiayias, and Tal Malkin. Bitr: Built-in tamper resilience. In *ASIACRYPT*, pages 740–758, 2011.

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

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

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

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

[18] 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.

[19] 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.

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

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

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

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

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

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

[26] 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.

[27] Vipul Goyal, Adam O'Neill, and Vanishree Rao. Correlated-input secure hash functions. In *TCC*, pages 182–200, 2011.

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

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

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

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

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

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

[34] Stefan Lucks. Ciphers secure against related-key attacks. In *FSE*, pages 359–370, 2004.

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

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

[37] Krzysztof Pietrzak. Subspace LWE. In *TCC*, pages 548–563, 2012.

[38] 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.

[39] Hoeteck Wee. Public key encryption against related key attacks. In *Public Key Cryptography*, pages 262–279, 2012.

A 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 [3, 2]. Recall that in this model we assume the existence of an external tamper-free and leakage-free storage (the floppy), which is needed to refresh the secret key on the tamperable device. An important difference between the floppy model considered in this paper and the model of [2] is that in our case the floppy can contain "user-specific" information, whereas in [2] 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.[5] A solution for this is to assume that each floppy contains a different master key as is the case in the *i*Floppy model, 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.

A.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 A.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 impersonation attacks with respect to $\mathcal{T}_{\mathsf{sk}}$ in the *i*Floppy 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 $accept$.

[5] We note that in the schemes of [2] 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.

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 $accept$.

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 4: Boosting BLT security to CLT security for ID schemes

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 A.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 4.

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.[6] 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 A.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 4 is $(\lambda, 1, \delta')$-CLT against impersonation attacks with respect to $\mathcal{T}_{\mathsf{sk}}$ in the* i*Floppy model, where $\delta' \leq \delta + negl(k)$.*

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:

[6] 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 $\mathcal{ID}$.

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

$$\Pr[\mathsf{A} \text{ wins}] = \Pr[\mathsf{A} \text{ wins} \wedge \text{FRESH}] + \Pr\left[\mathsf{A} \text{ wins} \wedge \overline{\text{FRESH}}\right], \tag{5}$$

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

$$\begin{aligned}\Pr[\mathsf{B}\text{ wins}] &= \Pr[\mathsf{B}\text{ wins} \wedge \text{GUESS}] + \Pr\left[\mathsf{B}\text{ wins} \wedge \overline{\text{GUESS}}\right] \\ &\geq \Pr[\mathsf{B}\text{ wins} \wedge \text{GUESS}] = \frac{1}{q(k)}\Pr\left[\mathsf{A}\text{ wins} \wedge \overline{\text{FRESH}}\right].\end{aligned} \tag{6}$$

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,

$$\Pr[\mathsf{C}\text{ wins}] \geq \Pr[\mathsf{A}\text{ wins} \wedge \text{FRESH}]. \tag{7}$$

Combining Eq. (5)-(7), 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 A.1 with our schemes from Section 3 resulting into tamper resilient identification schemes in the *i*Floppy model under polynomial many tampering and leakage attacks.

A.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 *i*Floppy 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 A.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 *i*Floppy 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:

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

(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 [2], the BHHO PKE scheme (cf. Section 4.3) allows for a very simple update mechanism. When we plug this encryption scheme in the construction of Figure 3, 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 *i*Floppy 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).[7] Unfortunately this requires us to make the proof from scratch. Since the proof relies on ideas already introduced in this paper or borrowed from [2], we give only a sketch here.

Theorem A.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.

[7] We stress that in the PKE case we cannot apply the same trick as for the compiler of Figure 4, since that would require to make the scheme interactive.

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.

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 4.1, 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 [2, Theorem 13] it follows from the affine version of the subspace hiding lemma (see [2, 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 4.1). 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

$$c^* = (g^{\mathbf{c}_0} = g^{r\boldsymbol{\alpha}}, c_1 = g^{\langle \mathbf{c}_0, \mathbf{x} \rangle} \cdot m_b). \tag{8}$$

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. (8) using the above vector $\mathbf{c}_0$.

As shown in [2], 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 $1/2$.

□

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. [21] 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 *i*Floppy model if additionally to correctness (cf. Definition A.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 4 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. [3] 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 4 is a* λ*-CLTC secure identification scheme in the* i*Floppy model.*

Proof (sketch). The proof is similar to the proof of Theorem A.1. The only difference is in the reduction to the security of the underlying identification scheme $\mathcal{ID}$. While in Theorem A.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 A.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 A.1 does not set any such usage restriction.