Journal of
CRYPTOLOGY

CrossMark

Bounded Tamper Resilience: How to Go Beyond the Algebraic Barrier

Ivan Damgård · Pratyay Mukherjee

Department of Computer Science, Aarhus University, Aarhus, Denmark

Sebastian Faust

Horst Görtz Institute, Ruhr-University Bochum, Bochum, Germany

Daniele Venturi

Department of Computer Science, Sapienza University of Rome, Rome, Italy
venturi@di.uniroma1.it

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 CCA-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.

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 [8,15] 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 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 [12] 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 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 Sect. 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 tamper resilience against arbitrary attacks possible? Unfortunately, the answer to the above question in its most general form is negative. As shown by Gennaro et al. [46], 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 [46] and assume that an adversary can only carry out a bounded

---

[1] The impossibility result of [46] 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 [46] may still apply.

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 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 cryptoschemes already used "in the wild": 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 cryptographic implementations do not come with such a built-in procedure to check the validity of the key; furthermore, such a self-destruct feature might not always be desirable, for instance in settings where faults are not adversarial, but due to some characteristic of the environment where the device is used (e.g., the temperature). Our notion of bounded tamper resilience allows to make formal security statements about algorithms running directly in implementations without self-destruct, so that 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 Sect. 5). 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* tampering 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 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 [58] provides security against bounded leakage from the secret key [7,53]. In Sect. 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, our analysis shows that by leaking the public keys corresponding to the modified secret keys allows, for each tampering query, to simulate *any number* of faulty transcripts (under the modified keys) by running the honest-verifier zero-knowledge simulator. Since the public key is significantly shorter than the secret key, 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), namely to all Σ-protocols where the secret key is significantly longer than the public key. In particular, we can instantiate our result with the generalized Guillou-Quisquater ID scheme [49], and its variant based on factoring [44], yielding tamper resilient identification based on factoring. We give more details in Sect. 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

Table 1. An overview of our results for bounded leakage and tamper resilience.

Tampering model	ID schemes		IND-CCA PKE
	Σ-Protocols	Okamoto	BHHO
Secret key	✓	✓	✓
Public parameters	n.a.	✓	n.a.
Continuous tampering iFloppy	✓	✓	✓
Key length	$\log|\mathcal{X}|$	$\ell \log p$	$\ell \log p$
Tampering queries	$\lfloor \log|\mathcal{X}|/\log|\mathcal{Y}| \rfloor - 2$	$\ell - 2$	$\ell - 3$

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 actually achieve a slightly worse bound depending on the conditional average min-entropy of the witness given the statement (cf. Sect. 3)

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 CCA-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, by proving that the BHHO cryptosystem [16] already satisfies it. The proof uses similar ideas as in the proof of the Okamoto identification scheme. In particular our analysis shows that by leaking a *single* group element per tampering query, one can answer *any number* of (restricted) decryption queries; hence restricted IND-CCA BLT security of BHHO follows from its leakage resilience (which was proven in [57]).

We then show how to transform the above weaker CCA-like notion to full-fledged 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 [52]. 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 simulate arbitrary decryption queries with only access to the restricted decryption oracle.

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. [52]. 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. [7] and recently studied in depth by Agrawal et al. [6]. 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 [6,7] 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 [6]). 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 described in Sect. 5 (Sect. 5.1 for ID schemes, and Sect. 5.2 for PKE schemes).

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 [7,25,34] 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 Sect. 6). 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. *Related 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 [12] 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 [10,56]. 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 [9,10,12,14,56,59,62]. A few important exceptions are described below.

---

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

In [13], 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. Also, the work of Goyal, O'Neill and Rao [47] considers polynomial relations that are induced to the inputs of a hash function. Finally Bellare, Cash and Miller [11] 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. [46]. 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 [37]. The original construction of [37] considers an adversary that can tamper independently with bits, a model further explored in [22,23]. This has been extended to small size blocks in [21], permutations [4,5], and recently to so-called split-state tampering [1–3,18,20,35,39,55] and global tampering [19,41,51]. Recently, non-malleable codes have also been used to protect a generic random access machine against tampering attacks [28,40].

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 be 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. [52] provide the 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 [52] 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 extent we can guarantee security when the complete circuitry is prone to tampering attacks [26,27,42,45,50,54]. 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.

Subsequent work. A preliminary version of this paper appeared as [29]. Subsequent work [30] shows how to transform an arbitrary cryptoscheme into one satisfying a slightly weaker form of BLT security; the number of tampering queries tolerated, however, is significantly smaller than the one achieved by the constructions analyzed in this paper. The transformation in [30] can be understood as applying a "non-malleable key derivation function" [41] to the state, a paradigm that was later extended in [61].

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 $(a^{x_1}, \ldots, a^{x_\ell})$. The inner product of $\mathbf{x} = (x_1, \ldots, x_\ell)$ and $\mathbf{y} = (y_1, \ldots, y_\ell)$ is $\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{i=1}^{\ell} x_i \cdot y_i$.

We denote with k the security parameter. A function $\delta(k)$ is called *negligible* in k (or simply negligible) if it vanishes faster than the inverse of any polynomial in k. A machine A is called *probabilistic polynomial time* (PPT) if for any input $x \in \{0, 1\}^*$ the computation of $\mathsf{A}(x)$ terminates in at most $poly(|x|)$ steps and A is probabilistic (i.e., it uses randomness as part of its logic). Random variables are usually denoted by capital letters. We sometimes abuse notation and denote a distribution and the corresponding random variable with the same capital letter, say X.

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 with 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 [33] of X given a random variable Z (over a set $\mathcal{Z}$) possibly dependent on X, is defined as $\widetilde{\mathbf{H}}_\infty(X|Z) := -\log \mathbf{E}_{z \leftarrow Z}[2^{-\mathbf{H}_\infty(X|Z=z)}]$. Following [7], 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. ([7]) The conditional min-entropy of a random variable X, conditioned on the experiment $\mathcal{E}$ is $\widetilde{\mathbf{H}}_\infty(X|\mathcal{E}) = -\log(\max_\mathsf{A} \Pr[\mathsf{A}^{\mathcal{E}(\cdot)}() = X])$. 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 notation.

We will rely on the following basic properties (see [33, 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

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

The ℓ-representation problem is equivalent to the Discrete Log problem [7, Lemma 4.1].

Representation problem based on RSA. Let Setup be a group generation algorithm that upon input 1^k outputs (N, e, d), where $N = p \cdot q$ such that p and q are two primes and also $ed \equiv 1 \mod ((p - 1)(q - 1))$. The RSA assumption states that for all PPT adversaries A there exists a negligible function $\delta : \mathbb{N} \to [0, 1]$ such that

$$\Pr\left[y = x^e \mod N : x \leftarrow \mathsf{A}(N, e, y), y \leftarrow \mathbb{Z}_N^*, (N, e, d) \leftarrow \mathsf{Setup}(1^k)\right] \leq \delta(k).$$

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

$$\mathbb{P}\big[(\mathbf{x}^\star, \rho^\star) \neq (\mathbf{x}, \rho); (y, (\mathbf{x}, \rho)), (y, (\mathbf{x}^\star, \rho^\star)) \in \mathfrak{R}_{\mathsf{RSA}} :$$
$$(y, (\mathbf{x}, \rho), (\mathbf{x}^\star, \rho^\star)) \leftarrow \mathsf{A}(N, e, g, g^{\boldsymbol{\alpha}}); (N, e, d) \leftarrow \mathsf{Setup}(1^k);$$
$$g \leftarrow \mathbb{Z}_N^*; \boldsymbol{\alpha} \leftarrow \mathbb{Z}_e^\ell\big] \leq \delta(k).$$

The ℓ-representation problem in $\mathbb{Z}_N$ is equivalent to the RSA problem (see [44,58]).

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

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

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 $\Pr\big[\mathsf{A} \text{ wins}\big] \leq \delta(k)$ 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*

Σ-protocols [24] 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 modeled as PPT algorithms) that it possesses a witness to the fact that a given element y

is in some language $\mathfrak{L}$. 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 from some challenge space $\mathcal{S}$ 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 below.

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 Extractability*

We recall the notion of true-simulation extractable (tSE) NIZKs [32]. This notion is similar to the notion of simulation-sound extractable NIZKs [48], 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 (Gen, Prove, 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 $\mathsf{Verify}^\omega(y, \pi) = 1$.
Composable non-interactive zero knowledge.	There exists a PPT simulator S such that, for any PPT adversary A, there exists a negligible function $\delta : \mathbb{N} \rightarrow [0, 1]$ such that $\lvert \Pr\left[\mathsf{A} \text{ wins}\right] - \frac{1}{2}\rvert \leq \delta(k)$ in the following game:

       1. The challenger samples $(\omega, \mathsf{tk}, \mathsf{ek}) \leftarrow \mathsf{Gen}(1^k)$ and gives ω to A.

2. 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 $b' = b$.

Strong 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 $\bot$ otherwise. There exists a PPT algorithm $\mathsf{Ext}(y, \pi, \mathsf{ek})$ such that, for all PPT adversaries A, there exists a negligible function $\delta : \mathbb{N} \rightarrow [0, 1]$ such that $\Pr\left[\mathsf{A} \text{ wins}\right] \leq \delta(k)$ 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) $(y^\star, x^\star) \notin \mathfrak{R}$.

In case A is given only one query to $\mathsf{S}'_{\mathsf{tk}}(\cdot)$, we speak of *one-time* strong tSE.

2.6. *A Note on Deterministic Versus 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 parameters 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 verifier V (holding pk) on common input pp. Such interaction outputs a result in $\{accept, reject\}$, where *accept* means P's identity is considered as valid.

Definition 3.1. Let $\lambda = \lambda(k)$ and $t = t(k)$ 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 bounded λ-leakage and t-tamper secure (in short (λ, t)-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, there exists a negligible function $\delta : \mathbb{N} \to [0, 1]$, such that $\Pr\big[\mathsf{A} \text{ wins}\big] \leq \delta(k)$ 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 $\mathsf{P}(\tilde{pp}_i, \tilde{sk}_i)$, where $(\tilde{sk}_i, \tilde{pp}_i) = T_i(sk, pp)$. The adversary can interact with the oracle $\mathsf{P}(\tilde{pp}_i, \tilde{sk}_i)$ a polynomially number of times, where the prover uses the tampered secret key $\tilde{sk}_i$ and the public parameter $\tilde{pp}_i$.

    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 $(\mathsf{A}(pp, pk) \rightleftarrows \mathsf{V}(pp, pk))$ outputs *accept* and $\sum_j \lambda_j \leq \lambda$.
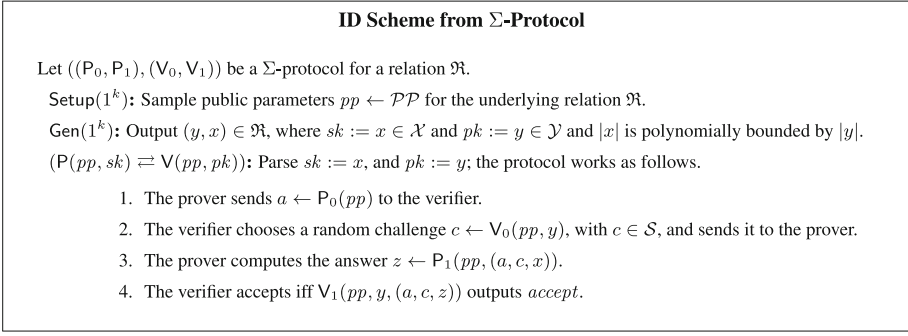
Notice that in the above definition the leakage is from the original secret key sk. This is without loss of generality as our tampering functions are modeled as deterministic circuits.

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 [7] 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 Sect. 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 Sect. 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 Sect. 3.3 we illustrate that the latter assumption is necessary, as otherwise the Okamoto ID scheme can be broken by (albeit contrived) attacks.

3.1. Σ-*Protocols are Tamper Resilient*

It is well known that Σ-protocols (see Sect. 2.4) are a natural tool to design ID schemes. The construction is depicted in Fig. 1. We restrict our analysis to Σ-protocols for so-called *complete* relations $\mathfrak{R}$ such that for each possible witness $x \in \mathcal{X}$, there is always

---

ID Scheme from Σ-Protocol

Let $((P_0, P_1), (V_0, V_1))$ 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 $(y, x) \in \mathfrak{R}$, where $sk := x \in \mathcal{X}$ and $pk := y \in \mathcal{Y}$ and $|x|$ is polynomially bounded by $|y|$.

$(\mathsf{P}(pp, sk) \rightleftarrows \mathsf{V}(pp, pk))$: Parse $sk := x$, and $pk := y$; 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 $c \leftarrow \mathsf{V}_0(pp, y)$, with $c \in \mathcal{S}$, 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*.

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

a corresponding statement $y \in \mathcal{Y}$ such that $(y, x) \in \mathfrak{R}$. As discussed later, the relations considered to instantiate our result satisfy this property.

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 [7], 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 a complete relation $\mathfrak{R}$, with challenge space $\mathcal{S}$ of size $O(k^{\log k})$, such that the representation problem is hard for $\mathfrak{R}$ (cf. Sect. 2.2). Assume that conditioned on the distribution of the public input $y \in \mathcal{Y}$, the witness $x \in \mathcal{X}$ has average min entropy at least β, i.e., $\widetilde{\mathbf{H}}_\infty(X|Y) \geq \beta$. Then, the ID scheme of Fig. 1 is $(\lambda(k), t(k))$-BLT secure against impersonation attacks with respect to $\mathcal{T}_{\mathsf{sk}}$, where*

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

Proof. Assume that there exists a polynomial $p(\cdot)$ 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:

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

Since $|\mathcal{S}|$ is super-polynomial in k, this contradicts the assumption that the representation problem is hard for $\mathfrak{R}$ (cf. Sect. 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 $\mathsf{P}(pp, x)$, $\mathsf{P}(pp, \tilde{x}_i)$ where $\tilde{x}_i = T_i(x) = T_i(sk) = \tilde{sk}_i$

for all $i \in [t]$. During the impersonation stage, B chooses a random challenge $c \in \mathcal{S}$ 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 $c' \in \mathcal{S}$ resulting in a transcript (a, c', z'). Whenever the two transcripts are accepting and $c' \neq c$, the special soundness property ensures that adversary B has extracted successfully some value $x^\star$ such that $(y, x^\star) \in \mathfrak{R}$. Let us call the event described above E_1, and the event $x = x^\star$ is denoted by E_2. Clearly,

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

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

Proof. The proof is identical to the proof of [7, Claim 4.1]. We repeat it here for completeness.

Denote with V the random variable corresponding to the view of A in one execution of the BLT game up to the challenge phase; this includes the public values (pp, y), the coins of A, the leakage, and the transcripts obtained via the oracles $\text{P}(pp, x)$, $\text{P}(pp, \tilde{x}_1), \ldots, \text{P}(pp, \tilde{x}_t)$. Notice that B is essentially playing the role of the challenger for A (as it knows a correctly distributed pair (y, x)), but at the end of the execution it rewinds A after it already sent the value a in the challenge phase, and samples a new challenge $c' \leftarrow \mathcal{S}$ hoping that $c' \neq c$ (where $c \leftarrow \mathcal{S}$ is the challenge sampled in the first run of A). Hence, the probability space of the event E_1 includes the randomness of the BLT game, the coins of A, and the randomness used to sample $c' \in \mathcal{S}$.

Let W be an indicator random variable, such that $W = 1$ when A wins in one execution of the BLT game (and $W = 0$ otherwise). By definition, $\mathbf{E}[W] := \delta$. Notice that $\Pr\left[E_1|V = v\right] \geq \Pr\left[W^2 = 1|V = v\right] - |\mathcal{S}|^{-1}$, since the probability that $c = c'$ is at most $|\mathcal{S}|^{-1}$ and this probability is independent of the fact that the two conversations (a, c, z) and (a, c', z') are accepting. Therefore,

$$\Pr\left[E_1\right] = \sum_v \Pr\left[E_1|V = v\right]\Pr\left[V = v\right] \geq \sum_v \Pr\left[W^2 = 1|V = v\right]\Pr\left[V = v\right] - |\mathcal{S}|^{-1}$$
$$= \mathbf{E}[W^2] - |\mathcal{S}|^{-1} \geq (\mathbf{E}[W])^2 - |\mathcal{S}|^{-1} = \delta^2 - |\mathcal{S}|^{-1} \tag{2}$$

where the first inequality of Eq. (2) follows by Jensen's inequality. This concludes the proof of the claim.   □

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

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 tampered secret keys) together

with the original public input y and the public parameters pp (which are tamper-free), i.e., $view_{\mathsf{A}}^{\mathcal{E}_0} = \{\boldsymbol{\Psi}, \boldsymbol{\Psi}_1, \dots, \boldsymbol{\Psi}_t\} \cup \{y, pp\}$. The vector $\boldsymbol{\Psi}$ and each of the vectors $\boldsymbol{\Psi}_i$ contains polynomially many transcripts of the form (a, c, z), corresponding (respectively) to the original key and to the ith tampered secret key.

We now move to experiment $\mathcal{E}_1$, which is the same as $\mathcal{E}_0$ with the modification that we add to A's view, for each tampering query, the public value $\tilde{y}_i \in \mathcal{Y}$ corresponding to the tampered witness $\tilde{x}_i = T_i(x) \in \mathcal{X}$; note that such value always exists, by our assumption that the relation $\mathfrak{R}$ is complete. Hence, $view_{\mathsf{A}}^{\mathcal{E}_1} = view_{\mathsf{A}}^{\mathcal{E}_0} \cup \{(\tilde{y}_1, \dots, \tilde{y}_t)\}$. Clearly,

$$\widetilde{\mathbf{H}}_\infty(X|\mathcal{E}_0) \geq \widetilde{\mathbf{H}}_\infty(X|\mathcal{E}_1). \tag{3}$$

Next, we consider experiment $\mathcal{E}_2$ where A is given only the values $(\tilde{y}_1, \dots, \tilde{y}_t)$, i.e., $view_{\mathsf{A}}^{\mathcal{E}_2} = \{\tilde{y}_1 \dots, \tilde{y}_t\} \cup \{y, pp\}$. 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 $\{\boldsymbol{\Psi}, \boldsymbol{\Psi}_i\}_{i\in[t]}$ can be computed as a deterministic function of $(y, \tilde{y}_1, \dots, \tilde{y}_t)$ as follows: For a randomly chosen challenge c run the HVZK simulator M upon input $(pp, \tilde{y}_i, c)$ and append the output (a, c, z) to $\boldsymbol{\Psi}_i$. (The same can be done to simulate $\boldsymbol{\Psi}$, by running $\mathsf{M}(pp, y, c)$.) It follows from perfect HVZK that this generates an identical distribution to that of experiment $\mathcal{E}_1$ and thus

$$\widetilde{\mathbf{H}}_\infty(X|\mathcal{E}_1) = \widetilde{\mathbf{H}}_\infty(X|\mathcal{E}_2). \tag{4}$$

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

$$\widetilde{\mathbf{H}}_\infty(X|\mathcal{E}_2) = \widetilde{\mathbf{H}}_\infty(X|\tilde{Y}_1, \dots, \tilde{Y}_t, Y) \geq \widetilde{\mathbf{H}}_\infty(X|Y) - |(\tilde{Y}_1, \dots, \tilde{Y}_t)| \geq \beta - t \log|\mathcal{Y}|, \tag{5}$$

where we used Lemma 2.1 together with the fact that the joint distribution $(\tilde{Y}_1, \dots, \tilde{Y}_t)$ 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 Eqs. (3)–(5) 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. $\qquad\square$

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

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

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

Instantiations. Below, we discuss a number of concrete instantiations for Theorem 3.1 based on standard hardness assumptions:

- *Generalized Okamoto.* This instantiation is described in detail in Sect. 3.2, where we additionally show that the generalized Okamoto ID scheme [58] remains secure also in case the public parameters (and not only the secret key) are subject to tampering.
- *Generalized Guillou-Quisquater.* Consider the relation $\mathfrak{R}_{\mathsf{RSA}}$ of Sect. 2.2. The relation is easily seen to be complete. Hardness of the ℓ-representation problem for $\mathfrak{R}_{\mathsf{RSA}}$ follows from the RSA assumption and was shown in [58]. A suitable Σ-protocol is described in [49]. A variant based on factoring can be obtained following Fischlin and Fischlin [44].

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 [55]), 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 [58], depicted in Fig. 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. Sect. 2.2). As proven in [7], this problem is equivalent to the Discrete Log problem in $\mathbb{G}$.

We first argue 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 relation $\mathfrak{R}_{\mathsf{DL}}$ is complete, and the protocol of Fig. 2 is a Σ-protocol which satisfies perfect HVZK; moreover $|\mathcal{Y}| = |\mathcal{S}| = p$ and the size of prime p is super-polynomial in k to ensure hardness of the Discrete Log problem. 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 obtain parameters $\lambda \leq (\ell - 1 - t)\log(p) - k$ and $t \leq \ell - 2$.

Next, we show that the generalized Okamoto ID scheme is actually secure for $\mathcal{T}_{\mathsf{split}}$ (with the same parameters).

Theorem 3.2. *Let $k \in \mathbb{N}$ be the security parameter and assume the Discrete Log problem is hard in $\mathbb{G}$. Then, the generalized Okamoto ID scheme is $(\lambda(k), t(k))$-BLT secure against impersonation attacks with respect to $\mathcal{T}_{\mathsf{split}}$, where*

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 $pp = (\mathbb{G}, g, g^{\boldsymbol{\alpha}}, p)$ where $g^{\boldsymbol{\alpha}} = (g_1, \ldots, g_\ell)$.

Gen(1^k): Select a vector $\mathbf{x} := (x_1, \ldots, x_\ell) \leftarrow \mathbb{Z}_p^\ell$ and set $pk := y = \prod_{i=1}^\ell g_i^{x_i}$ and $sk := \mathbf{x}$.

($\mathsf{P}(pp, sk) \rightleftarrows \mathsf{V}(pp, pk)$): Parse $pp = (\mathbb{G}, g, g^{\boldsymbol{\alpha}}, p)$, $sk := (x_1, \ldots, x_\ell)$, and $pk := y$; the protocol works as follows.

    1. The prover chooses a random vector $\mathbf{r} := (r_1, \ldots, r_\ell) \leftarrow \mathbb{Z}_p^\ell$ and sends $a = \prod_{i=1}^\ell g_i^{r_i}$.

    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 $\prod_{i=1}^\ell g_i^{z_i} = a \cdot y^c$.

Fig. 2. Generalized Okamoto identification scheme.

$$\lambda \leq (\ell - 1 - t)\log(p) - k \quad and \quad t \leq \ell - 2.$$

Proof (Sketch).. The proof follows closely the proof of Theorem 3.1, with the key difference that we also have to take care of tampering with respect to $pp = (\mathbb{G}, g, g^{\boldsymbol{\alpha}}, p)$. We sketch how this can be done below.

Given an adversary A winning the BLT security experiment with non-negligible advantage $\delta(k)$, consider the same reduction B outlined in the proof of Theorem 3.1, attacking the ℓ-representation problem in $\mathbb{G}$. Notice that the reduction can, as before, perfectly simulate the environment for A as it knows honestly generated parameters (pp, pk, sk). In particular, Claim 1 still holds here with $|\mathcal{S}| = p$.

It remains to prove Claim 2. 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 $\tilde{pp}_i$ for all $i \in [t]$. In particular, the elements $(a, c, \mathbf{z})$ contained in each vector $\boldsymbol{\Psi}_i$ in the view of experiment $\mathcal{E}_0$ are now sampled from $\mathsf{P}(\tilde{pp}_i, \tilde{\mathbf{x}}_i)$, where $\tilde{\mathbf{x}}_i = T_i^{sk}(\mathbf{x})$ and $\tilde{pp}_i = T_i^{pp}(pp)$ for all $i \in [t]$. We then modify $\mathcal{E}_1$ and $\mathcal{E}_2$ by additionally including the values of the tampered public parameters $\{\tilde{pp}_i\}_{i \in [t]}$.

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 $\{\tilde{pk}_i, \tilde{pp}_i\}_{i \in [t]}$. 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 $\tilde{pp} = (\tilde{\mathbb{G}}, \tilde{g}, \tilde{g}_1, \ldots, \tilde{g}_\ell, \tilde{p})$. (Of course in this case the protocol is not sound.) The HVZK simulator $\mathsf{M}(\tilde{pp}, \tilde{y}, c)$ works as follows: Choose $z_1, \ldots, z_\ell$ at random in $\mathbb{Z}_{\tilde{p}}$ and if $\tilde{y} \neq 0 \mod \tilde{p}$, then compute $a = (\prod_{i=1}^\ell \tilde{g}_i^{z_i})/\tilde{y}^c \mod \tilde{p}$. In case $\tilde{y} = 0 \mod \tilde{p}$, then just set $a = 0$.[4] For any $(\tilde{\mathbf{x}}, \tilde{pp}) = (T^{sk}(\mathbf{x}), T^{pp}(pp))$, the distributions $\mathsf{M}(\tilde{pp}, \tilde{y}, c)$ and $(\mathsf{P}(\tilde{pp}, \tilde{\mathbf{x}}) \rightleftarrows \mathsf{V}(\tilde{pp}, \tilde{y}))$ are both uniformly random over all values $(a, c, \mathbf{z} = (z_1, \ldots, z_\ell))$ such that $\prod_{i=1}^\ell \tilde{g}_i^{z_i} = a\tilde{y}^c \mod \tilde{p}$.

---

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

Therefore the simulation perfectly matches the honest conversation. This proves Eq. (4). Now Eq. (5) follows from the fact that the tampering functions T^{pp} cannot depend on sk. The rest of the proof remains the same.   □

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 to 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 $T(\mathbf{x}, pp) = (\tilde{\mathbf{x}}, \tilde{pp})$ to be as follows:

- The witness is unchanged, i.e., $\mathbf{x} = \tilde{\mathbf{x}}$.
- The value $\tilde{p}$ is some prime of size $|\tilde{p}| \approx |p|$ such that the Discrete Log problem is easy in the corresponding group $\tilde{\mathbb{G}}$. (This can be done efficiently by choosing $\tilde{p} - 1$ to be the product of small prime (power) factors [60].)
- Let $\tilde{g}$ be a generator of $\tilde{\mathbb{G}}$ (which exists since $\tilde{p}$ is a prime) and define the new generators as $\tilde{g}_i = \tilde{g}^{x_i} \bmod \tilde{p}$.

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

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 $\tilde{p}$ to be a prime of size $|\tilde{p}| = \Omega(\ell |p|)$, such that the Discrete Log problem is easy in $\tilde{\mathbb{G}}$. (This can be done as in the proof of Claim 3.)

- Choose a generator $\tilde{g}$ of $\tilde{\mathbb{G}}$; define $\tilde{g}_1 = \tilde{g}$ and $\tilde{g}_j = 1$ for all $j = 2, \ldots, \ell$.
- Define the witness to be $\tilde{\mathbf{x}}$ such that $\tilde{x}_1 = x_1 || \ldots || x_\ell$ and $\tilde{x}_j = 0$ for all $j = 2, \ldots, \ell$.

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

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 [43]. By applying the Fiat-Shamir transformation to the protocol of Fig. 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} = $ (Setup, KGen, Enc, 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 $\bot$.

Definition 4.1. Let $\lambda = \lambda(k)$ and $t = t(k)$ 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 $(\lambda(k), t(k))$-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, there exists a negligible function $\delta(k) : \mathbb{N} \to [0, 1]$, such 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 $\mathsf{Dec}(\tilde{sk}_i, \cdot)$, where $\tilde{sk}_i = T_i(sk)$. This oracle outputs polynomially many decryptions of ciphertexts using secret key $\tilde{sk}_i$.
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 $b = b'$, $\sum_j \lambda_j \le \lambda$ and c_b has not been queried for to the decryption oracle.

In case $t = 0$ we get, as a special case, the notion of semantic security against a-posteriori chosen-ciphertext $\lambda(k)$-key-leakage attacks from [57]. Notice that A is not allowed to tamper with the secret key after seeing the challenge ciphertext. As we show in Sect. 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 Sect. 4.1 we define a weaker notion which we call *restricted* IND-CCA BLT security. In Sect. 4.2 we show a general transformation from restricted IND-CCA BLT security to full-fledged IND-CCA BLT security relying on tSE NIZK proofs [31] 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 Sect. 4.3, we prove that a variant of the BHHO encryption scheme [57] satisfies our notion of restricted IND-CCA BLT security.

4.1. *Restricted IND-CCA 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 $\lambda = \lambda(k)$ and $t = t(k)$ 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 *restricted* IND-CCA $(\lambda(k), t(k))$-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, there exists a negligible function $\delta(k) : \mathbb{N} \to [0, 1]$, such that $\Pr\left[\mathsf{A\ wins}\right] \le \frac{1}{2} + \delta(k)$ 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 $\mathsf{Dec}^*(\tilde{sk}_i, \cdot, \cdot)$, where $\tilde{sk}_i = T_i(sk)$. 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 $\tilde{m} = \mathsf{Dec}(\tilde{sk}_i, c)$ 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 $\sum_j \lambda_j \leq \lambda$.

We note that, by setting $t = 0$, we recover the original notion of semantic security under λ-key-leakage attacks for public key encryption, as defined in [57].

4.2. *A General Transformation*

We compile an arbitrary restricted IND-CCA BLT-secure encryption scheme into a full-fledged IND-CCA BLT-secure one by appending to the ciphertext c an argument of "plaintext knowledge" π computed through a (one-time, strong) tSE NIZK argument system (cf. Sect. 2.5). The same construction has been already used by Dodis et al. [31] 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 restricted IND-CCA BLT security.

Theorem 4.1. *Let $k \in \mathbb{N}$ be the security parameter. Assume that $\mathcal{PKE}$ is a restricted IND-CCA $(\lambda(k), t(k))$-BLT secure encryption scheme and that* (Gen, Prove, Verify) *is a one-time strong tSE NIZK argument system for relation $\mathfrak{R}_{\mathsf{PKE}}$. Then, the encryption scheme $\mathcal{PKE}'$ of Fig. 3 is IND-CCA $(\lambda(k), t(k))$-BLT secure.*

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

---

From Restricted to Full-Fledged 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 $\bot$; otherwise, output $m = \mathsf{Dec}(sk, c)$.

Fig. 3. How to transform a restricted IND-CCA BLT-secure PKE into an IND-CCA BLT-secure PKE.

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. Our goal is to upper bound $|\Pr[\mathsf{A} \text{ wins in } \mathsf{G}_1] - 1/2|$.

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

Game G_3. 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 $\mathsf{Dec}(\tilde{sk}_i, \cdot)$ (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 $\tilde{m} = \mathsf{Dec}(\tilde{sk}_i, c)$. It follows from one-time strong tSE 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 restricted IND-CCA BLT security of $\mathcal{PKE}$. Assume there exists a distinguisher D between G_3 and G_4. We build an adversary B breaking restricted IND-CCA 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 $T_i \in \mathcal{T}_{sk}$, 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 $\mathsf{Dec}^*(\tilde{sk}_i, \cdot, \cdot)$ and receive the answer $\tilde{m}$. Return $\tilde{m}$ 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 restricted IND-CCA (λ, t)-BLT secure, it must be $|\Pr[\mathsf{A} \text{ wins in } G_3] - \Pr[\mathsf{A} \text{ wins in } G_4]| \leq \delta_3(k)$ for a negligible function $\delta_3 : \mathbb{N} \to [0, 1]$.

As clearly $\Pr[\mathsf{A} \text{ wins in } G_4] = 1/2$, we have obtained:

$$
\begin{aligned}
|\Pr[\mathsf{A} \text{ wins in } G_1] - 1/2| &= |\Pr[\mathsf{A} \text{ wins in } G_1] - \Pr[\mathsf{A} \text{ wins in } G_4]| \\
&\leq |\Pr[\mathsf{A} \text{ wins in } G_1] - \Pr[\mathsf{A} \text{ wins in } G_2]| \\
&+ |\Pr[\mathsf{A} \text{ wins in } G_2] \\
&- \Pr[\mathsf{A} \text{ wins in } G_3]| + |\Pr[\mathsf{A} \text{ wins in } G_3] \\
&- \Pr[\mathsf{A} \text{ wins in } G_4]| \\
&\leq \delta_1(k) + \delta_2(k) + \delta_3(k) = negl(k).
\end{aligned}
$$

This concludes the proof. □

4.3. *Instantiation from BHHO*

We show that the variant of the encryption scheme introduced by Boneh et al. [16] used in [57] is restricted IND-CCA 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 of BHHO (which was already proven in [57]).

The BHHO PKE scheme works as follows: (1) Algorithm Setup chooses a group $\mathbb{G}$ of prime order p with generator g and let $pp = (\mathbb{G}, g, p)$; (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* $\mathbb{G}$ *(cf.* Sect. 2.2*). Then, the BHHO encryption scheme is* restricted *IND-CCA* $(\lambda(k), t(k))$-*BLT secure, where*

$$
\lambda \leq (\ell - 2 - t) \log p - \omega(\log k) \quad and \quad t \leq \ell - 3.
$$

Proof. Naor and Segev [57, Section 5.2] showed that BHHO is restricted IND-CCA $(\lambda', 0)$-BLT secure up to $\lambda' \leq (\ell - 2) \log p - \omega(\log k)$.[5] Assume there exists an adversary A which breaks restricted IND-CCA (λ, t)-BLT security with probability $\delta(k) = 1/p(k)$, for some polynomial $p(\cdot)$ and infinitely many values of $k \in \mathbb{N}$. We build an adversary B which breaks restricted IND-CCA $(\lambda', 0)$-BLT security of the encryption scheme, with the same advantage, yielding a contradiction.

---

[5] Recall that for $t = 0$ no decryption query is allowed, and thus restricted IND-CCA $(\lambda', 0)$-BLT security collapses to the notion of semantic security against λ'–key-leakage attacks from [57].

Adversary B uses A as a black-box and is described below.

Reduction B^{A} :

1. Receive (pp, pk) from the challenger and forward these 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 query in order to retrieve the value $\tilde{h}_i = \prod_{j=1}^{\ell} g_j^{-\tilde{x}_{j,i}}$, where $\tilde{\mathbf{x}}_i = T_i(\mathbf{x}) = (\tilde{x}_{1,i}, \ldots, \tilde{x}_{\ell,i})$. When A asks for a decryption query (m, r), return $\tilde{m} = (h^r \cdot m) \cdot \tilde{h}_i^r$.
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; forward c_b to A.
5. Output whatever A does.

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

$$\tilde{m} = (h^r \cdot m) \cdot \tilde{h}_i^r = c_1 \cdot \left(\prod_{j=1}^{\ell} g_j^{-\tilde{x}_{j,i}} \right)^r = c_1 \cdot \prod_{j=1}^{\ell} g_j^{-r \cdot \tilde{x}_{j,i}} = c_1 \cdot g^{-\sum_{j=1}^{\ell} r\alpha_j \cdot \tilde{x}_{j,i}}$$
$$= c_1 \cdot g^{\langle \mathbf{c}_0, \tilde{\mathbf{x}}_i \rangle},$$

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 $\mathsf{Dec}^*(\tilde{sk}_i, \cdot, \cdot)$. Hence, B has the same advantage as A which concludes the proof. $\qquad\square$

We remark that efficient proofs of plaintext knowledge for the BHHO PKE scheme (to use within the transformation of Fig. 3) are already known (see, e.g., [17,38]).

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 [13,62]. The reason why the schemes of [13,62] can achieve this stronger flavor 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 $(0, 1)$- 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 $m^* \in \{0, 1\}$ uniformly 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 $\tilde{sk} = sk$.
   - In case $m_b = 1$, sample $(pk^*, sk^*) \leftarrow \mathsf{KGen}(1^k)$ until $\mathsf{Dec}(sk^*, c^*) \neq m^*$. When this happens, let $\tilde{sk} = sk^*$.

3. Query the decryption oracle $\mathsf{Dec}(\tilde{sk}, \cdot)$ 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 $\Pr\left[\mathsf{Dec}(sk^*, c^*) \neq m^*\right] \approx 1/2$ 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 upper 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. $\qquad\square$

5. 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 [6,7]. 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 [6] is that in our case the floppy can contain "user-specific" information, whereas in [6] 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*Floppymodel.

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.[6] 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

---

[6] We note that in the schemes of [6] 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.

the attacker is allowed to leak adaptively λ bits from the current secret key and tamper with it for some bounded number of times.

5.1. *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 5.1. Let $\lambda = \lambda(k)$ be a parameter, 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 $(\lambda(k), 1)$-CLT secure against impersonation attacks with respect to $\mathcal{T}_{\mathsf{sk}}$ in the *i*Floppy model, if the following properties are satisfied.

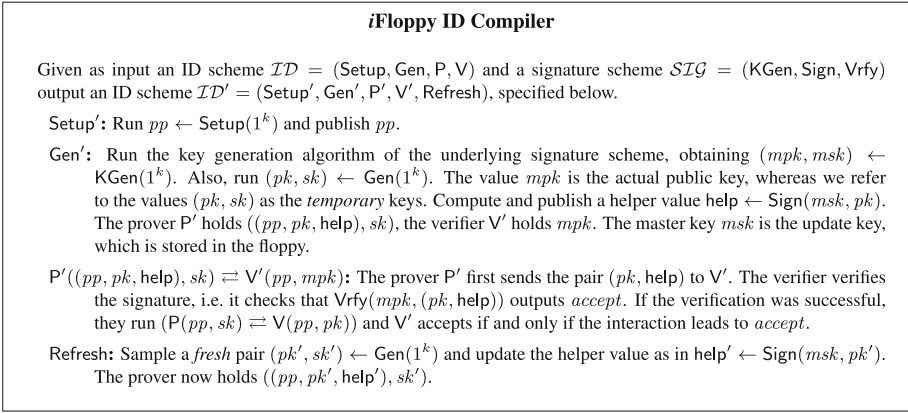(i) *Correctness.* For all $pp \leftarrow \mathsf{Setup}(1^k)$, $(pk, sk, uk) \leftarrow \mathsf{Gen}(1^k)$ we have that:

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

(ii) *Security.* For all PPT adversaries A, there exists a negligible function $\delta : \mathbb{N} \to [0, 1]$, such that $\Pr\big[\mathsf{A} \text{ wins}\big] \leq \delta(k)$ 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 $\tilde{T}_i : \mathcal{SK} \to \mathcal{SK}$ and is given oracle access to $\mathsf{P}(pp, \tilde{sk}_i)$, where $\tilde{sk}_i = T_i(sk_i)$.
   (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 $(\mathsf{A}(pp, pk) \rightleftarrows \mathsf{V}(pp, pk))$ outputs *accept*.

Remark 1. One could also consider a more general definition where between two key updates A is allowed to ask multiple leakage queries with output size λ_j, as long as $\sum_j \lambda_j \leq \lambda$. Similarly, we could allow A to tamper in each round for t times with the secret key sk_i. The constructions in this section can be proven secure in this extended setting, but we stick to Definition 5.1 for simplicity.

A general compiler. We now describe a compiler to boost any (λ, t)-BLT secure ID scheme (P, V), to a (λ, t)-CLT secure 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 Fig. 4.

---

iFloppy ID Compiler

Given as input an ID scheme $\mathcal{ID} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{P}, \mathsf{V})$ and a signature scheme $\mathcal{SIG} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vrfy})$ output 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 $((pp, pk, \mathsf{help}), sk)$, the verifier V' holds mpk. The master key msk is the update key, which is stored in the floppy.

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

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

Fig. 4. Boosting BLT security to CLT security for ID schemes.

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 wants to prove its identity, it first sends the temporary pk together with the helper value and V verifies this signature using mpk.[7] 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 prove the following result.

Theorem 5.1. *If $\mathcal{SIG}$ is EUF-CMA and $\mathcal{ID}$ is $(\lambda, 1)$-BLT secure against impersonation attacks with respect to $\mathcal{T}_{\mathsf{sk}}$, then the scheme $\mathcal{ID}'$ output by the compiler of Fig. 4 is $(\lambda, 1)$-CLT secure against impersonation attacks with respect to $\mathcal{T}_{\mathsf{sk}}$ in the iFloppy model.*

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 $\Pr\big[\mathsf{A} \text{ wins}\big] \geq \delta(k)$, where $\delta(k) = 1/p(k)$ for some polynomial $p(\cdot)$ and infinitely many $k \in \mathbb{N}$. The CLT experiment for $\mathcal{ID}'$ is specified below:

CLT Experiment:

1. The challenger runs $pp \leftarrow \mathsf{Setup}'(1^k)$ and $(mpk, msk) \leftarrow \mathsf{KGen}(1^k)$ and gives (pp, mpk) to A.

---

[7] 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 $\mathcal{ID}$.

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 $\mathsf{P}'((pp, pk_i, \mathsf{help}_i), sk_i)$.
   – Answer the leakage and tampering query from A using key sk_i. The leakage query consists of a function $L_i : \{0, 1\}^* \rightarrow \{0, 1\}^\lambda$; the tampering query consists of a tampering function $T_i : \mathcal{SK} \rightarrow \mathcal{SK}$.

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 $(\mathsf{A}(pp, pk^\star) \rightleftarrows \mathsf{V}(pp, pk^\star))$ 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

$$\Pr\left[\mathsf{A} \text{ wins}\right] = \Pr\left[\mathsf{A} \text{ wins} \wedge \text{FRESH}\right] + \Pr\left[\mathsf{A} \text{ wins} \wedge \overline{\text{FRESH}}\right], \tag{6}$$

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 query 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 $(\mathsf{A}(pp, pk^\star) \rightleftarrows \mathsf{V}(pp, pk^\star))$ 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 honestly generated 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

$$\Pr\left[\mathsf{B} \text{ wins}\right] = \Pr\left[\mathsf{B} \text{ wins} \wedge \text{GUESS}\right] + \Pr\left[\mathsf{B} \text{ wins} \wedge \overline{\text{GUESS}}\right]$$
$$\geq \Pr\left[\mathsf{B} \text{ wins} \wedge \text{GUESS}\right] = \frac{1}{q(k)}\Pr\left[\mathsf{A} \text{ wins} \wedge \overline{\text{FRESH}}\right]. \tag{7}$$

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 $\mathsf{P}'((pp, \mathsf{help}_i, pk_i), sk_i)$ using knowledge of key 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 $(\mathsf{A}(pp, pk^\star) \rightleftarrows \mathsf{V}(pp, pk^\star))$ 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,

$$\Pr\left[\mathsf{C} \text{ wins}\right] \geq \Pr\left[\mathsf{A} \text{ wins} \wedge \text{FRESH}\right]. \tag{8}$$

Combining Eqs. (6)–(8), we obtain:

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

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

Remark 2. Assuming factoring or DL is hard, we can instantiate Theorem 5.1 with our schemes from Sect. 3 resulting into tamper resilient identification schemes in the *i*Floppy model under polynomial many tampering and leakage attacks.

5.2. *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 5.2. Let $\lambda = \lambda(k)$ be a parameter, and let $\mathcal{T}_{\mathsf{sk}}$ be some set of functions such that $T \in \mathcal{T}_{\mathsf{sk}}$ has a type $T : \mathcal{SK} \rightarrow \mathcal{SK}$. We say that $\mathcal{PKE}$ is IND-CCA $(\lambda(k), 1)$-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, there exists a negligible function $\delta : \mathbb{N} \rightarrow [0, 1]$, such that $\Pr\left[\mathsf{A} \text{ wins}\right] \leq 1/2 + \delta(k)$ 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 $L_i : \{0, 1\}^* \rightarrow \{0, 1\}^\lambda$ and receives back $L_i(sk_i)$.
    (b) A specifies a function $T_i : \mathcal{SK} \rightarrow \mathcal{SK}$ and is given oracle access to $\mathsf{Dec}(\tilde{sk}_i, \cdot)$, where $\tilde{sk}_i = T_i(sk_i)$.
    (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 [6], the BHHO PKE scheme (cf. Sect. 4.3) allows for a very simple update mechanism. When we plug this encryption scheme in the construction of Fig. 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 $pp = (\mathbb{G}, g, p, \omega)$. (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

$\pi \leftarrow \mathsf{Prove}^{\omega}((pk, c), (m, r))$ for $((pk, c), (m, r)) \in \mathfrak{R}_{\mathsf{PKE}}$ (cf. Fig. 3). (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).[8] Unfortunately this requires us to make the proof from scratch. Since the proof relies on ideas already introduced in this paper or borrowed from [6], we give only a sketch here.

Theorem 5.2. *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 $(\lambda(k), 1)$-CLT secure with respect to $\mathcal{T}_{\mathsf{sk}}$ in the iFloppy model, where $\lambda \leq (\ell - 3) \log p - \omega(\log k)$.*

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

Game G_1. This is the IND-CCA CLT game. In particular the challenge ciphertext is a pair of the form $(c^* = (g^{r\boldsymbol{\alpha}}, h^r \cdot m_b), \pi^*)$ where $\pi^* \leftarrow \mathsf{Prove}^{\omega}((pk, c^*), (m_b, r))$, for $m_b \in \{m_0, m_1\}$ and $b \leftarrow \{0, 1\}$. Our goal is to bound $|\Pr[\mathsf{A} \text{ wins in } G_1] - 1/2|$.

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 $\mathsf{Dec}(\tilde{\mathbf{x}}_i, \cdot)$ (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 $\tilde{m} = \mathsf{Dec}(\tilde{\mathbf{x}}_i, c)$. As argued in the proof of Theorem 4.1, G_2 and G_3 are computationally close by the one-time strong 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 $G_{3,i}$. Sample at the beginning a random $(\ell - 2)$-dimensional subspace $S \subset \ker(\boldsymbol{\alpha})$ and modify the refreshing of the key as follows.

---

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

- 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 [6, Theorem 13] it follows from the affine version of the subspace hiding lemma (see [6, 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 $\tilde{h}_i$ needed to simulate the tampered decryption oracle $\mathsf{Dec}(\tilde{\mathbf{x}}_i, \cdot)$ (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

$$c^* = (g^{\mathbf{c}_0} = g^{r\boldsymbol{\alpha}}, c_1 = g^{\langle \mathbf{c}_0, \mathbf{x} \rangle} \cdot m_b). \tag{9}$$

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. (9) using the above vector $\mathbf{c}_0$. As shown in [6, Theorem 13], 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 $1/2$ concluding the proof.

6. Tampering with the Computation in the *i*Floppy Model

We finally consider the question of tampering with the computation for ID schemes in the *i*Floppy model. In particular, 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 P̃ as long as P and P̃ 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 P̃. 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. [36] 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 P̃. Notice that the only way in which B can "communicate" with the big adversary A is via the output of the tampered prover P̃.

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 6.1. 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 5.1), 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 $\Pr\left[\mathsf{A} \text{ wins}\right] \le \delta(k)$ 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 *i*th 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 an algorithm $\tilde{\mathsf{P}}_i$ and obtains the faulty transcript $(\tilde{\mathsf{P}}_i(pp, sk_i) \rightleftarrows \mathsf{V}(pp, pk))$.
   (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 $(\mathsf{A}(pp, pk) \rightleftarrows \mathsf{V}(pp, pk))$ outputs *accept*.

In the theorem below we show that when we instantiate the general compiler from Fig. 4 with an appropriate identification scheme with key size $k \gg s + s_{\mathsf{sig}}$ (s is the length of the transcript, and s_{sig} is the length of a message/signature pair) and security against $s + s_{\mathsf{sig}}$ bits of leakage, we can achieve security with respect to Definition 6.1. Identification schemes that are secure in the Bounded Retrieval Model (BRM) satisfy these conditions and have been constructed, e.g., by Alwen et al. [7] based on the Generalized Okamoto ID scheme.

Theorem 6.1. *Let* $\mathcal{SIG} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vrfy})$ *be an EUF-CMA secure signature scheme with message/signature pairs of size* s_{sig}*, and* $\mathcal{ID} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{P}, \mathsf{V})$ *be*

an $(s + s_{\mathsf{sig}} + \lambda)$-leakage and 0-tamper resilient identification scheme with transcript length s. Then, $\mathcal{ID}'$ from Fig. 4 *is a λ-CLTC secure identification scheme in the* i*Floppy model.*

The proof is similar to the proof of Theorem 5.1; hence, we provide only a sketch here. The only difference is in the reduction to the security of the underlying identification scheme $\mathcal{ID}$. While in Theorem 5.1 we simulate the tampering with access to the tampering oracle, we now simulate the tampering queries $\tilde{\mathsf{P}}'_i$, i.e., the faulty transcript $(\tilde{\mathsf{P}}'_i((pp, pk_i, \mathsf{help}_i), sk_i) \rightleftarrows \mathsf{V}'(pp, mpk))$ with access to the leakage oracle. As the transcript has length $s + s_{\mathsf{sig}}$, we can learn the entire faulty transcript from the leakage oracle. This is where we loose $s + s_{\mathsf{sig}}$ bits in the leakage bound compared to the underlying identification scheme.

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

CLTC Experiment:

1. The challenger runs $pp \leftarrow \mathsf{Setup}'(1^k)$ and $(mpk, msk) \leftarrow \mathsf{KGen}(1^k)$ and gives (pp, mpk) to A.
2. For each $i = 1, \ldots, q(k)$ (where $q(k)$ is some polynomial in the security parameter), the challenger does the following:
   - During round i sample $(pk_i, sk_i) \leftarrow \mathsf{Gen}(1^k)$ and compute $\mathsf{help}_i \leftarrow \mathsf{Sign}(msk, pk_i)$.
   - Give A oracle access to $\mathsf{P}'((pp, \mathsf{help}_i, pk_i), sk_i)$.
   - Answer the leakage and tampering query from A using key sk_i. The leakage query consists of a function $L_i : \{0, 1\}^* \rightarrow \{0, 1\}^\lambda$, yielding $L_i(sk_i)$; the tampering query consists of a modified algorithm $\tilde{\mathsf{P}}'_i$ (with the same input/output domain as the honest prover algorithm P'), yielding a transcript from $(\tilde{\mathsf{P}}'_i((pp, pk_i, \mathsf{help}_i), sk_i) \rightleftarrows \mathsf{V}'(pp, mpk))$.
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 $(\mathsf{A}(pp, pk^\star) \rightleftarrows \mathsf{V}(pp, pk^\star))$ and outputs whatever V does.

Let FRESH be the same event as defined in the proof of Theorem 5.1. We have

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

where all probabilities are taken over the randomness space of the CLTC experiment and over the randomness of A. We now describe a reduction B (using A as a black-box) which breaks leakage resilience 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 query 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 by submitting the same function to the target leakage oracle.
   - Simulate tampering query $\tilde{\mathsf{P}}'_j$ by submitting the leakage function corresponding to $\tilde{\mathsf{P}}'_j((pp, \overline{\mathsf{help}}, \overline{pk}), \cdot)$ to the target leakage oracle, obtaining the corresponding modified transcript.[9]
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 $(\mathsf{A}(pp, pk^\star) \rightleftarrows \mathsf{V}(pp, pk^\star))$ 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 honestly generated keys, whereas round j is simulated using the target prover oracle and the leakage oracle which allows for simulating A's leakage and tampering queries using a total of $\lambda + s + s_{\mathsf{sig}}$ bits of leakage from $\overline{sk}$ (i.e., λ bits for A's leakage queries, and $(s + s_{\mathsf{sig}})$ bits for the faulty transcript $\tilde{\mathsf{P}}'_j \rightleftarrows \mathsf{V}'$ corresponding to A's tampering query). Denote with GUESS the same event as in the proof of Theorem 5.1. As before, we have

$$\Pr\left[\mathsf{B} \text{ wins}\right] \geq \frac{1}{q(k)} \Pr\left[\mathsf{A} \text{ wins} \wedge \overline{\text{FRESH}}\right].$$

Finally, consider the same reduction C defined in the proof of Theorem 5.1, trying to break existential unforgeability of $\mathcal{SIG}$ using A as a black-box. Notice that the reduction

---

[9] Here is how B simulates the transcript with more details. Without loss of generality, assume that a basic interaction $\mathsf{P} \rightleftarrows \mathsf{V}$ consists of μ messages for odd $\mu \in \mathbb{N}$; recall that the interaction $\mathsf{P}' \rightleftarrows \mathsf{V}'$ also consists of μ messages, where the pair $(\overline{pk}, \overline{\mathsf{help}})$ is appended to the first message sent by P. Thus, the ith message of the interaction $\tilde{\mathsf{P}}'_j \rightleftarrows \mathsf{V}'$, for $i \in [\mu]$ can be simulated by a leakage query hard-wiring a description of $\tilde{\mathsf{P}}'_j$ together with $(m_1, \ldots, m_{i-1}, pp, \overline{pk}, \overline{\mathsf{help}})$, where $(m_1, \ldots, m_{i-1})$ is the current partial transcript.

can still perfectly simulate all of A's queries, because it knows all the pairs (pk_i, sk_i) for all $i \in [q]$. Hence, as in the proof of Theorem 5.1,

$$\Pr\left[\mathsf{C} \text{ wins}\right] \geq \Pr\left[\mathsf{A} \text{ wins} \wedge \text{Fresh}\right].$$

Combining the previous equations, we obtain $q(k) \cdot \Pr\left[\mathsf{B} \text{ wins}\right] + \Pr\left[\mathsf{C} \text{ wins}\right] \geq \Pr\left[\mathsf{A} \text{ wins}\right] \geq \delta(k)$. A contradiction. $\qquad\square$

We note that the above result seemingly achieves a stronger security notion than Theorem 5.1 (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 the two 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 5.1 does not set any such usage restriction.

Acknowledgements

References

[1] D. Aggarwal, Y. Dodis, T. Kazana, M. Obremski, Non-malleable reductions and applications, in *STOC* (2015)

[2] D. Aggarwal, Y. Dodis, S. Lovett, Non-malleable codes from additive combinatorics, in *STOC*, (2014), pp. 774–783

[3] D. Aggarwal, S. Dziembowski, T. Kazana, M. Obremski, Leakage-resilient non-malleable codes, in *TCC*, (2015), pp. 398–426

[4] S. Agrawal, D. Gupta, H.K. Maji, O. Pandey, M. Prabhakaran, Explicit non-malleable codes against bit-wise tampering and permutations, in *CRYPTO*, (2015), pp 538–557.

[5] S. Agrawal, D. Gupta, H.K. Maji, O. Pandey, M. Prabhakaran, A rate-optimizing compiler for non-malleable codes against bit-wise tampering and permutations, in *TCC*, (2015), pp. 375–397

[6] S. Agrawal, Y. Dodis, V. Vaikuntanathan, D. Wichs, On continual leakage of discrete log representations, in *ASIACRYPT*, (2013), pp. 401–420

[7] J. Alwen, Y. Dodis, D. Wichs, Leakage-resilient public-key cryptography in the bounded-retrieval model, in *CRYPTO*, (2009), pp. 36–54

[8] R. Anderson, M. Kuhn, Tamper resistance: a cautionary note, in *WOEC'96: Proceedings of the 2nd conference on Proceedings of the Second USENIX Workshop on Electronic Commerce*, (USENIX Association, Berkeley, 1996), p. 1

[9] B. Applebaum, D. Harnik, Y. Ishai, Semantic security under related-key attacks and applications, in *ICS*, (2011), pp. 45–60

[10] M. Bellare, D. Cash, Pseudorandom functions and permutations provably secure against related-key attacks, in *CRYPTO*, (2010), pp. 666–684

[11] M. Bellare, D. Cash, R. Miller, Cryptography secure against related-key attacks and tampering. In *ASIACRYPT*, (2011), pp. 486–503

[12] M. Bellare, T. Kohno, A theoretical treatment of related-key attacks: RKA-PRPs, RKA-PRFs, and applications, in *EUROCRYPT*, (2003), pp. 491–506

[13] M. Bellare, K.G. Paterson, S. Thomson. RKA security beyond the linear barrier: IBE, encryption and signatures, in *ASIACRYPT*, (2012), pp. 331–348

[14] R. Bhattacharyya, A. Roy, Secure message authentication against related key attack, in *FSE* (2013)

[15] D. Boneh, R.A. DeMillo, R.J. Lipton, On the importance of eliminating errors in cryptographic computations. *J. Cryptol.*, **14**(2):101–119 (2001)

[16] D. Boneh, S. Halevi, M. Hamburg, R. Ostrovsky, Circular-secure encryption from decision diffie-hellman, in *CRYPTO*, (2008), pp. 108–125

[17] J. Camenisch, N. Chandran, V. Shoup, A public key encryption scheme secure against key dependent chosen plaintext and adaptive chosen ciphertext attacks, in *EUROCRYPT*, (2009), pp. 351–368

[18] E. Chattopadhyay, D. Zuckerman. Non-malleable codes against constant split-state tampering, in *FOCS*, (2014), pp. 306–315

[19] M. Cheraghchi, V. Guruswami, Capacity of non-malleable codes, in *Innovations in Theoretical Computer Science, ITCS*, (2014), pp. 155–168

[20] M. Cheraghchi, V. Guruswami, Non-malleable coding against bit-wise and split-state tampering, in *TCC*, (2014), pp. 440–464

[21] S.G. Choi, A. Kiayias, T. Malkin, BiTR: Built-in tamper resilience, in *ASIACRYPT*, (2011), pp. 740–758

[22] S. Coretti, Y. Dodis, B. Tackmann, D. Venturi, Non-malleable encryption: simpler, shorter, stronger. *IACR Cryptol. ePrint Archive*, **772** (2015)

[23] S. Coretti, U. Maurer, B. Tackmann, D. Venturi, From single-bit to multi-bit public-key encryption via non-malleable codes, in *TCC*, (2015), pp. 532–560

[24] R. Cramer, *Modular Design of Secure yet Practical Cryptographic Protocols*. PhD thesis, University of Amsterdam, (1996)

[25] G. Di Crescenzo, R.J. Lipton, S. Walfish, Perfectly secure password protocols in the bounded retrieval model, in *TCC*, (2006), pp. 225–244

[26] D. Dachman-Soled, Y.T. Kalai, Securing circuits against constant-rate tampering, in *CRYPTO*, (2012), pp. 533–551

[27] D. Dachman-Soled, Y.T. Kalai, Securing circuits and protocols against 1/poly(k) tampering rate, in *TCC*, (2014), pp. 540–565

[28] D. Dachman-Soled, F.-H. Liu, E. Shi, H.-S. Zhou, Locally decodable and updatable non-malleable codes and their applications, in *TCC*, (2015), pp. 427–450

[29] I. Damgård, S. Faust, P. Mukherjee, D. Venturi, Bounded tamper resilience: How to go beyond the algebraic barrier, in *ASIACRYPT*, (2013), pp. 140–160

[30] I. Damgård, S. Faust, P. Mukherjee, D. Venturi, The chaining lemma and its application, in *ICITS*, (2015), pp. 181–196

[31] Y. Dodis, K. Haralambiev, A. López-Alt, D. Wichs, Cryptography against continuous memory attacks, in *FOCS*, (2010), pp. 511–520

[32] Y. Dodis, K. Haralambiev, A. López-Alt, D. Wichs, Efficient public-key cryptography in the presence of key leakage, in *ASIACRYPT*, (2010), pp. 613–631

[33] Y. Dodis, R. Ostrovsky, L. Reyzin, A. Smith, Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM J. Comput.***38**(1), 97–139 (2008)

[34] S. Dziembowski, Intrusion-resilience via the bounded-storage model, in *TCC*, (2006), pp. 207–224

[35] S. Dziembowski, T. Kazana, M. Obremski, Non-malleable codes from two-source extractors, in *CRYPTO*, (2013), pp. 239–257

[36] S. Dziembowski, T. Kazana, D. Wichs, One-time computable self-erasing functions, in *TCC*, (2011), pp. 125–143

[37] S. Dziembowski, K. Pietrzak, D. Wichs, Non-malleable codes, in *ICS*, (2010), pp. 434–452

[38] S. Faust, M. Kohlweiss, G.A. Marson, D. Venturi, On the non-malleability of the fiat-shamir transform, in *INDOCRYPT*, (2012), pp. 60–79

[39] S. Faust, P. Mukherjee, J.B. Nielsen, D. Venturi, Continuous non-malleable codes, in *TCC* (2014)

[40] S. Faust, P. Mukherjee, J.B. Nielsen, D. Venturi, A tamper and leakage resilient von Neumann architecture, in *PKC*, (2015), pp. 579–603

[41] S. Faust, P. Mukherjee, D. Venturi, D. Wichs, Efficient non-malleable codes and key-derivation for poly-size tampering circuits, in *EUROCRYPT*, (2014), pp. 111–128

[42] S. Faust, K. Pietrzak, D. Venturi, Tamper-proof circuits: How to trade leakage for tamper-resilience. In *ICALP (1)*, (2011), pp. 391–402

[43] A. Fiat, A. Shamir, How to prove yourself: practical solutions to identification and signature problems, in *CRYPTO*, (1986), pp. 186–194

[44] M. Fischlin, R. Fischlin, The representation problem based on factoring, in *CT-RSA*, (2002), pp. 96–113

[45] D. Genkin, Y. Ishai, M. Prabhakaran, A. Sahai, E. Tromer, Circuits resilient to additive attacks with applications to secure computation, in *STOC*, (2014), pp. 495–504

[46] R. Gennaro, A. Lysyanskaya, T. Malkin, S. Micali, T. Rabin, Algorithmic tamper-proof (ATP) security: theoretical foundations for security against hardware tampering, in *TCC*, (2004), pp. 258–277

[47] V. Goyal, A. O'Neill, V. Rao, Correlated-input secure hash functions, in *TCC*, (2011), pp. 182–200

[48] J. Groth, Simulation-sound NIZK proofs for a practical language and constant size group signatures, in *ASIACRYPT*, (2006), pp. 444–459

[49] L.C. Guillou, J.-J. Quisquater, A "paradoxical" identity-based signature scheme resulting from zero-knowledge, in *CRYPTO*, (1988), pp. 216–231

[50] Y. Ishai, M. Prabhakaran, A. Sahai, D. Wagner, Private circuits II: keeping secrets in tamperable circuits, in *EUROCRYPT*, (2006), pp. 308–327

[51] Z. Jafargholi, D. Wichs, Tamper detection and continuous non-malleable codes, in *TCC*, (2015), pp. 451–480,

[52] Y.T. Kalai, B. Kanukurthi, A. Sahai, Cryptography with tamperable and leaky memory, in *CRYPTO*, (2011), pp. 373–390

[53] J. Katz, V. Vaikuntanathan, Signature schemes with bounded leakage resilience, In *ASIACRYPT*, (2009), pp. 703–720

[54] A. Kiayias, Y. Tselekounis, Tamper resilient circuits: the adversary at the gates, in *ASIACRYPT*, (2013), pp. 161–180

[55] F.-H. Liu, A. Lysyanskaya, Tamper and leakage resilience in the split-state model, in *CRYPTO*, (2012), pp. 517–532

[56] S. Lucks, Ciphers secure against related-key attacks, in *FSE*, (2004), pp. 359–370

[57] M. Naor, G. Segev, Public-key cryptosystems resilient to key leakage, in *CRYPTO*, (2009), pp. 18–35

[58] T. Okamoto, Provably secure and practical identification schemes and corresponding signature schemes, in *CRYPTO*, (1992), pp. 31–53

[59] K. Pietrzak, Subspace LWE, in *TCC*, (2012), pp. 548–563

[60] S. Pohlig, M. Hellman, An improved algorithm for computing logarithms over and its cryptographic significance. *IEEE Trans. Inform. Theory*, **24**(1), 106–110 (1978)

[61] B. Qin, S. Liu, T.H. Yuen, R.H. Deng, K. Chen, Continuous non-malleable key derivation and its application to related-key security, in *PKC*, (2015), pp. 557–578

[62] H. Wee, Public key encryption against related key attacks, in *PKC*, (2012), pp. 262–279