Digital Signatures with Memory-Tight Security in the Multi-Challenge Setting

Denis Diemert, Kai Gellert, Tibor Jager, and Lin Lyu

Bergische Universität Wuppertal, Germany
{denis.diemert, kai.gellert, tibor.jager, lin.lyu}@uni-wuppertal.de

Abstract. The standard security notion for digital signatures is "single-challenge" (SC) EUF-CMA security, where the adversary outputs a single message-signature pair and "wins" if it is a forgery. Auerbach *et al.* (CRYPTO 2017) introduced *memory-tightness* of reductions and argued that the right security goal in this setting is actually a stronger "multi-challenge" (MC) definition, where an adversary may output many message-signature pairs and "wins" if at least one is a forgery. Currently, no construction from simple standard assumptions is known to achieve full tightness with respect to time, success probability, and memory simultaneously. Previous works showed that memory-tight signatures cannot be achieved via certain natural classes of reductions (Auerbach *et al.*, CRYPTO 2017; Wang *et al.*, EUROCRYPT 2018). These impossibility results may give the impression that the construction of memory-tight signatures is difficult or even impossible.
We show that this impression is false, by giving the first constructions of signature schemes with full tightness in all dimensions in the MC setting. To circumvent the known impossibility results, we first introduce the notion of *canonical reductions* in the SC setting. We prove a general theorem establishing that every signature scheme with a canonical reduction is already memory-tightly secure in the MC setting, provided that it is strongly unforgeable, the adversary receives only one signature per message, and assuming the existence of a tightly-secure pseudorandom function. We then achieve memory-tight *many-signatures-per-message* security in the MC setting by a simple additional generic transformation. This yields the first memory-tightly, strongly EUF-CMA-secure signature schemes in the MC setting. Finally, we show that standard security proofs often already can be viewed as canonical reductions. Concretely, we show this for signatures from lossy identification schemes (Abdalla *et al.*, EUROCRYPT 2012), two variants of RSA Full-Domain Hash (Bellare and Rogaway, EUROCRYPT 1996), and two variants of BLS signatures (Boneh *et al.*, ASIACRYPT 2001).

1 Introduction

Work-factor-tightness. The security of many cryptosystems depends on computational hardness assumptions, where security is proven by a reduction from

Supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme, grant agreement 802823.

breaking the cryptosystem with respect to some security definition to breaking the hardness assumption. When such cryptosystems are concretely instantiated, cryptographic parameters such as the size of algebraic groups and moduli must be determined. If this is done in theoretically-sound way, that is, supported by the security guarantees provided by a reduction from breaking the cryptosystem to breaking the underlying assumption, then the *security loss* of the reduction has to be taken into account.

Let $\mathcal{A}$ be an adversary on a given cryptosystem with respect to a given security model, and let $\mathcal{R}$ be a reduction in a security proof that turns $\mathcal{A}$ into an algorithm solving some assumed-to-be-hard computational problem. Let $(t_{\mathcal{A}}, \epsilon_{\mathcal{A}})$ and $(t_{\mathcal{R}}, \epsilon_{\mathcal{R}})$ be the running time and advantage of $\mathcal{A}$ and $\mathcal{R}$, respectively. Then, the security loss is defined as L such that

$$L \cdot \frac{\epsilon_{\mathcal{R}}}{t_{\mathcal{R}}} = \frac{\epsilon_{\mathcal{A}}}{t_{\mathcal{A}}}$$

where $\epsilon_{\mathcal{A}}/t_{\mathcal{A}}$ and $\epsilon_{\mathcal{R}}/t_{\mathcal{R}}$ are the *work factors* of $\mathcal{A}$ and $\mathcal{R}$, respectively.[1] This is the standard approach to measure concrete security, which was established by Bellare and Ristenpart [8, 9].

In the classical asymptotic setting a reduction is considered *efficient* if L is bounded by some polynomial, which may be large. However, if L is large, then a theoretically-sound concrete instantiation must compensate the security loss with larger parameters, at the cost of efficiency of the deployed cryptosystem. Often L depends on deployment parameters (such as the number of users and the number of issued signatures, for instance), which are determined by the application context. These might not be exactly known at the time of initial deployment, or they might unexpectedly encounter significant increase over time. Hence, these parameters must be chosen conservatively, based on a strict upper bounds, which may lead to overly large parameters that come with very significant performance overhead. Therefore it is desirable to have *tight* security proofs, where L is a constant, and thus independent of such deployment parameters. Such schemes can be efficiently instantiated with optimal cryptographic parameters in arbitrary application contexts, independent of the number of users, the number of issued signatures, and other application parameters. If L is a constant, then we usually call $\mathcal{R}$ a *tight* reduction. In this paper, we will refer to this notion as *work-factor-tightness*, in order to distinguish it from the notion of *memory-tightness* discussed below.

Memory-tightness. Auerbach *et al.* [4] explained that in addition to the work factor also the *memory* consumed by a reduction is relevant. This is particularly relevant when security is reduced to so-called *memory-sensitive* computational problems, where the efficiency of known algorithms depends on the amount of

[1] In the asymptotic setting, $\epsilon_{\mathcal{A}}$, $t_{\mathcal{A}}$, $\epsilon_{\mathcal{R}}$, and $t_{\mathcal{R}}$ are functions in a security parameter. In this case L is a function in the security parameter, too. In the concrete security setting the running times, success probabilities, and the security loss are real numbers.

memory that is available. This includes, for instance, known algorithms for the classical discrete logarithm problem modulo a prime number, the integer factorization problem, Learning With Errors (LWE), or Short Integer Solutions (SIS), and many more. Other problems are (currently) not considered memory-sensitive, such as the discrete logarithm problem in elliptic curve groups. However, whether a given computational problem is memory-sensitive or not may change with the discovery of new algorithms and the impact of memory on their performance. See [4] for an in-depth discussion of memory-sensitivity.

In order to address this gap, Auerbach *et al.* [4] introduced the notion of *memory-tightness*, which additionally takes the memory consumed by a reduction into account. In addition to discussing the memory-sensitivity of computational problems, they also consider the memory-tightness of finding multi-collisions for hash functions and of reductions between different security notions of digital signature schemes.

Since its introduction in 2017, the concept of memory tightness has drawn much attention and led to many follow-up works. This includes works on memory lower bounds of reductions by Wang *et al.* [51] (EUROCRYPT 2018), memory tightness of authenticated encryption by Ghoshal, Jaeger, and Tessaro [30] (CRYPTO 2020), memory tightness of hashed ElGamal by Ghoshal and Tessaro [31] (EUROCRYPT 2020), and memory tightness for key encapsulation mechanisms by Bhattacharyya [12] (PKC 2020). Hence, memory tightness is already a well-established concept in cryptography that receives broad interest.

Memory-tightly secure signatures. In the standard *existential unforgeability under chosen-message attacks* (EUF-CMA) security model, the adversary receives a public key pk and then has access to a signing oracle that, on input of any message m from the message space of the signature scheme, computes a signature $\sigma \xleftarrow{\$} \mathsf{Sign}(sk, m)$, stores m in a list $\mathcal{Q}$, and returns σ. The adversary successfully breaks the security of the signature scheme if it outputs a forgery (m^*, σ^*) such that σ^* is a valid signature for m^* with respect to pk, and $m^* \notin \mathcal{Q}$. Auerbach *et al.* call this the *single-challenge* setting, since the adversary has only one attempt to forge a signature. They also introduce a stronger *multi-challenge* security definition, where the adversary may output multiple valid message-signature pairs and it "wins" if at least one of them is a new forgery in the sense that no signature was requested for the corresponding message throughout the security experiment.

Obviously, when considering the random-access memory (RAM) model, both security notions are tightly equivalent when memory consumption is not considered. In one direction, given a multi-challenge adversary, one can simply store all message-signature pairs that the adversary has obtained from its experiment in a list. Whenever the adversary outputs a message-signature pair, it is checked whether it is contained in the list. If not, then it is a valid forgery in the single-challenge setting. The opposite direction is even more trivial. However, note that this reduction is not memory-tight, as it requires memory linear in the number of signing queries. Auerbach *et al.* even showed that it is very difficult to prove that both notions are memory-tightly equivalent, by giving an impossibility result that covers a large class of natural reductions. This result was subsequently revisited and extended by Wang *et al.* [51].

The only known construction of a signature scheme with memory-tight security proof is due to Auerbach *et al.* [4]. They show that the RSA full-domain hash signature scheme can be proven memory-tightly secure under the RSA assumption. This is already a significant result, since it introduces clever tricks to deal with a programmable random oracle in a memory-tight way. However, it is still limited, since the reduction is only memory-tight, but not work-factor-tight. This is because the tightness lower bounds from [6, 19, 42, 43] still apply, such that a linear security loss in the number of signature queries is unavoidable.[2] Furthermore, Auerbach *et al.* only achieve memory-tightness in the weaker single-challenge setting, but not yet in the stronger multi-challenge setting. To the best of our knowledge, there exists currently no signature scheme, which has a security proof that is *fully* tight, that is, simultaneously memory-tight *and* work-factor-tight.

One main difficulty of achieving memory-tightly-secure signatures in the multi-challenge setting is to build a reduction which does not have to store the sequence of random oracle queries made by the adversary. While it seems easy to replace a random oracle with a pseudorandom function, this must be done very carefully, in particular in security proofs that “program” a random oracle, in order to achieve consistency. Here we can partially build upon techniques developed by Auerbach *et al.* [4]. Furthermore, another major difficulty in achieving security in the multi-challenge setting is to build a reduction which does not have to store the history of message-signature pairs obtained by the adversary through signing queries.

Our contributions. We summarize our contributions as follows.

- We present a sequence of transforms that give rise to the first digital signature schemes that simultaneously achieve tightness in all three dimensions: running time, success probability, and memory. The construction is efficient and yields practical signature schemes.
- On a technical level, we show how to circumvent known impossibility result by introducing the notion of “canonical reductions”, which can be seen as a new “non-black-box” perspective that applies to many well-known standard reductions in security proofs for signature schemes.
- We show the applicability of this approach by considering the construction of signatures from lossy identification schemes (LID) by Abdalla *et al.* [1, 2], which can be viewed as a generalization of the security proof for Katz–Wang signatures [44]. We further demonstrate the versatility of our technique by applying it to well-known signature schemes like RSA-FDH [11] (with the proof following [19] with a loss linear in the number of signing queries). Then, we additionally show that by using the technique by Katz and Wang [44] of

[2] There is also a work-factor-tight security proof for RSA full domain hash based on the Phi Hiding assumption [42, 43], but this proof seems not compatible with the memory-tight implementation of the random oracle from [4].

signing the message together with an extra random bit, we can eliminate the linear security loss and achieve both memory and working factor tightness. We also show similar results for Boneh–Lynn–Shacham (BLS) signatures [14]. All of our results directly achieve *strong* unforgeability. For a comparison of our result with previous analyses of these scheme, consider Table 1.

Table 1. Comparison of our result to previous analyses of the considered schemes. All analyses are in the random oracle model. Let λ be the security parameter, let q_{H} be the number of random oracle queries, let q_{S} the number of signing queries, let e be the basis of the natural logarithm, let $|\mathbb{G}|$ be the size of the representation of a group element of a cyclic group $\mathbb{G}$ of prime order q, let $|\mathbb{Z}_N|$ denote the size of the representation of an element of $\mathbb{Z}_N$, let N be a RSA modulus, let e be a RSA public exponent, and let $|\mathbb{G}_1|$ (resp. $|\mathbb{G}_2|$) be the size of the representation of a group element of group $\mathbb{G}_1$ (resp. $\mathbb{G}_2$) of some bilinear group $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$. Note that for comparability, we chose to instantiate the LID-based schemes with DDH. Due to collision resistance, the nonce length chosen for our transform from Section 4 is 2λ.

Constr.	**Proof**	**Asm.**	**Sec.**	**Sec. Loss**	**Mem. Loss**	$\|\boldsymbol{pk}\|$	$\|\boldsymbol{\sigma}\|$
LID-based	[1,2]	DDH	EUF-CMA	$\mathcal{O}(1)$	$\mathcal{O}(q_{\mathsf{H}} + q_{\mathsf{S}})$	$4\,\|\mathbb{G}\|$	$3\,\|\mathbb{Z}_q\|$
	Ours	DDH	msEUF-CMA	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$4\,\|\mathbb{G}\|$	$3\,\|\mathbb{Z}_q\| + 2\lambda$
RSA-FDH	[18]	RSA	EUF-CMA	$\mathrm{e} \cdot q_{\mathsf{S}}$	$\mathcal{O}(q_{\mathsf{H}} + q_{\mathsf{S}})$	$\|N\| + \|e\|$	$\|\mathbb{Z}_N\|$
	[4]	RSA	EUF-CMA	$\mathrm{e} \cdot q_{\mathsf{S}}$	$\mathcal{O}(1)$	$\|N\| + \|e\|$	$\|\mathbb{Z}_N\|$
	Ours	RSA	msEUF-CMA	$\mathrm{e} \cdot q_{\mathsf{S}}$	$\mathcal{O}(1)$	$\|N\| + \|e\|$	$\|\mathbb{Z}_N\|$
RSA-FDH+	[44]	RSA	EUF-CMA	$\mathcal{O}(1)$	$\mathcal{O}(q_{\mathsf{H}} + q_{\mathsf{S}})$	$\|N\| + \|e\|$	$\|\mathbb{Z}_N\|$
	Ours	RSA	msEUF-CMA	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\|N\| + \|e\|$	$\|\mathbb{Z}_N\| + 2\lambda$
BLS	[14]	(co-)CDH	EUF-CMA	$\mathrm{e} \cdot (q_{\mathsf{S}} + 1)$	$\mathcal{O}(q_{\mathsf{H}} + q_{\mathsf{S}})$	$\|\mathbb{G}_2\|$	$\|\mathbb{G}_1\|$
	Ours	(co-)CDH	msEUF-CMA	$\mathrm{e} \cdot (q_{\mathsf{S}} + 1)$	$\mathcal{O}(1)$	$\|\mathbb{G}_2\|$	$\|\mathbb{G}_1\|$
BLS+	[44]	(co-)CDH	EUF-CMA	$\mathcal{O}(1)$	$\mathcal{O}(q_{\mathsf{H}} + q_{\mathsf{S}})$	$\|\mathbb{G}_2\|$	$\|\mathbb{G}_1\|$
	Ours	(co-)CDH	msEUF-CMA	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\|\mathbb{G}_2\|$	$\|\mathbb{G}_1\| + 2\lambda$

Our approach. Our approach can be divided into two steps.

1. At first we show how to generically transform an entire class of signature schemes from the single-challenge setting to the multi-challenge setting. During this step, it is actually useful to consider a weaker "one-signature-per-message" security notion, where an adversary may only request one (instead of many) signature per message via its signing oracle.[3]

[3] Of course, one-signature-per-message security is equivalent to standard security for signature schemes with deterministic signing algorithm, however, we are not aware of any such signature scheme which achieves tight security, not even in the clas-

We require that the security reduction of the underlying scheme follows a canonical pattern that is compatible with our approach to prove memory tightness. Essentially, we require that the reduction can be split into *stateless* "canonical procedures" for simulating signatures, extracting solutions from forgeries, and computing hash values (e.g., if a random oracle is needed).
The main idea is now to "de-randomize" all canonical procedures, meaning that we give all procedures access to the *same* random function but require that they otherwise behave deterministically. Note that the "one-signature-per-message" restriction helps us here, as the procedures can rely on the random function to derive randomness for one signature per message from the message by calling the random function. Giving all procedures access to the same random function, ensures consistency across procedures (e.g., a signature may need to be consistent with the simulation of a random oracle).
We also show that many standard security proofs for signatures indeed can be seen as canonical reductions, so that our generic result applies.
Finally, to generically achieve memory-tightness in the multi-challenge setting, we can replace the "global random function" with a pseudorandom function. This yields a generic transform (with tightness in all dimensions) producing a signature scheme secure in the "one-signature-per-message" and multi-challenge setting.
2. In the second step we apply a simple generic transform (again, with tightness in all dimensions) that lifts any signature scheme from the "one-signature-per-message" to the standard "many-signatures-per-message" setting. To this end, any message is signed alongside a random nonce, which intuitively "expands" the set of valid signatures per message.

Applying both steps sequentially does not influence the tightness of a signature scheme in any dimension.

Related work. In the literature, "tightness" usually refers to what we call work-factor tightness in this paper. That is, running times and success probabilities are considered, but memory is not. There is a large number of research results in this area, with tightly-secure constructions of many different types of cryptosystems, including digital signatures [22, 37, 38, 44, 49], public-key encryption [7, 29, 37], (hierarchical) identity-based encryption [13, 16], authenticated key exchange [5, 17, 32, 46], and symmetric encryption [34, 36, 41], for instance. Tight security is also increasingly considered for real-world cryptosystems, such as [20, 23, 34, 39]. There are also various impossibility results for different types and classes of cryptosystems, such as [19, 26–28, 41–43, 48, 50], for instance.

As already mentioned, the notion of memory-tightness was only relatively recently introduced in [4]. They also introduced the single- and multi-challenge security model, and gave the first (and currently only) memory-tight security proof for a digital scheme in the weaker single-challenge setting, which however

sical sense that does not consider memory tightness. There are several impossibility results, showing that tightness is often difficult to achieve for such signature schemes [6, 19, 43].

is not yet work-factor-tight. They also gave a first impossibility result, showing that a certain class of reductions cannot be used to reduce multi-challenge security to single-challenge security. Wang *et al.* [51] revisited this impossibility result and showed that multi-challenge security is impossible to achieve for a large class of reductions, unless a work-factor tightness is sacrificed. They showed a lower bound on the memory of a large class of black-box reductions from the multi-challenge unforgeability of unique signatures to any computational hardness assumption, another lower bound for restricted reductions from multi-challenge security to single-challenge security for cryptographic primitives with unique keys, and a lower bound for multi-collisions of hash functions with large domain, which extends a similar result from [4]. Bhattacharyya [12] and Ghoshal and Tessaro [31] independently considered the memory-tightness of hashed El-Gamal public-key encryption. Ghoshal, Jaeger, and Tessaro [30] considered the memory-tightness of authenticated encryption.

Outline. The remainder of this paper is organized as follows. In Section 2, we define the computational model and the used complexity measures, alongside with standard definitions of cryptographic primitives. In Section 3, we present how to achieve multi-challenge security from any signature scheme secure in the single-challenge setting that follows a canonical reduction. In Section 4, we present our generic transform to lift any signature scheme from "one-signature-per-message" to the standard "many-signatures-per-message" setting. Finally, we show how our transforms can be applied to existing signature schemes, achieving the first fully tight signature schemes in the multi-challenge setting.

2 Preliminaries

For strings a and b, we denote the concatenation of these strings by $a \parallel b$. We denote the operation of assigning a value y to a variable x by $x \coloneqq y$. If S is a finite set, we denote by $x \xleftarrow{\$} S$ the operation of sampling a value uniformly at random from set S and assigning it to variable x. For any probabilistic algorithm $\mathcal{A}$, we denote $y \leftarrow \mathcal{A}(x; r)$ the process of running $\mathcal{A}$ on input x with random coins r and assign the output to y, and we denote $y \xleftarrow{\$} \mathcal{A}(x)$ as $y \leftarrow \mathcal{A}(x; r)$ for uniformly random r.

2.1 Computational Model and Complexity Measures

In this paper, we adapt the computation model used in [4] and recall the most important aspects in this section.

Algorithms. We assume all algorithms in this paper to be random access machines (RAMs). A RAM has access to memory using words of a fixed size λ and a constant number of registers each holding a single word. If an algorithm $\mathcal{A}$ is probabilistic, then the corresponding RAM is equipped with a special instruction that fills a distinguished register with (independent) random bits. However, we do not allow the RAM to rewind random bits to access previously used random bits. That is, $\mathcal{A}$ needs to store the random bits in this case. To run algorithm $\mathcal{A}$, the RAM is executed, where the input of the algorithm is written in the RAM's memory. To denote this, we overload notation and write $x \stackrel{\$}{\leftarrow} \mathcal{A}(y_1, y_2, \dots)$ to denote that random variable x takes on the value of algorithm $\mathcal{A}$ ran on inputs $y_1, y_2, \dots$ with fresh random coins. Sometimes we also denote this random variable simply by $\mathcal{A}(y_1, y_2, \dots)$. In case $\mathcal{A}$ is deterministic, we write $x \coloneqq \mathcal{A}(y_1, y_2, \dots)$, to denote that $\mathcal{A}$ on inputs $y_1, y_2, \dots$ outputs x.

Oracles. In addition, algorithm $\mathcal{A}$ sometimes has access to (stateful) oracles $(\mathcal{O}_1, \mathcal{O}_2, \dots)$. Each of these oracles also is defined by a RAM. To interact with an oracle $\mathcal{O}_i$, the RAM of algorithm $\mathcal{A}$ has three fixed regions in the memory only used for the oracle state $st_{\mathcal{O}}$, the input to the oracle and the output of the oracle. By default, these regions are empty. To query the oracle $\mathcal{O}_i$, $\mathcal{A}$ writes the query in the region of its memory reserved for the oracle input and executes a special instruction to run the RAM of $\mathcal{O}_i$ on this input together with the oracle state $st_{\mathcal{O}}$. The RAM implementing $\mathcal{O}_i$ uses its own memory and both the output and the updated oracle state $st_{\mathcal{O}}$ in the designated regions in $\mathcal{A}$'s memory. For notation, we denote that an algorithm $\mathcal{A}$ has oracle access to an algorithm *oracle* by $\mathcal{A}^{\mathcal{O}}$.

Security experiment. The security definition and proofs presented in this paper are mostly game-based. A security experiment (or game) can simply be viewed as an algorithm that runs another algorithm as subroutine, e.g., an adversary $\mathcal{A}$, and the subroutine may also be provided with a series of (stateful) oracles. As a security experiment is simply an algorithm it is also implemented by a RAM.

Complexity measures for runtime and memory consumption. We define the complexity measures for runtime and memory according to Auerbach *et al.* [4].

Runtime. Let $\mathcal{A}$ be an algorithm and Exp be a security game. We define $\mathbf{Time}(\mathcal{A})$ to be the runtime of $\mathcal{A}$ as the worst-case number of computation steps over all inputs of length λ and all possible random choices. In addition, we define $\mathbf{LocalTime}(\mathcal{A})$ to be the number of computation steps of $\mathcal{A}$ playing Exp *without* the additional steps induced by the oracle access to Exp. This quantifier allows us to precisely measure how much additional computation steps are necessary per oracle.

Memory consumption. Let $\mathcal{A}$ be an algorithm and Exp be a security game. We define $\mathbf{Mem}(\mathcal{A})$ to be the memory (in λ-width words) of the code of $\mathcal{A}$ plus the worst-case number of registers used at any point during computation, over all inputs of length λ and all possible random choices. Similar to before, we define $\mathbf{LocalMem}(\mathcal{A})$ to be the memory required to execute Exp with algorithm $\mathcal{A}$ *without* the additional memory induced by the oracle access to Exp. This quantifier allows us to precisely measure how much additional memory is necessary per oracle.

2.2 Pseudorandom Functions

We recall the standard indistinguishability definition for pseudorandom functions. This is one of the main tools used to make reductions memory-tight.

Definition 1. *Let* $\lambda \in \mathbb{N}$. *Let* $\mathsf{F}\colon \{0,1\}^\lambda \times \{0,1\}^* \to \mathcal{R}$ *be a keyed function, where* $\mathcal{R}$ *is a finite set. We define the advantage of an adversary* $\mathcal{A}$ *in breaking the pseudorandomness of* F *as*

$$\mathsf{Adv}_{\mathsf{F}}^{\mathsf{PRF\text{-}sec}}(\mathcal{A}) \coloneqq \left| \Pr\left[\mathcal{A}^{\mathsf{F}(k,\cdot)} = 1\right] - \Pr\left[\mathcal{A}^{f(\cdot)} = 1\right] \right|$$

where $k \xleftarrow{\$} \{0,1\}^\lambda$ *and* $f\colon \{0,1\}^* \to \mathcal{R}$ *is a random function.*

2.3 Digital Signatures

We recall the standard definition of a *digital signature scheme* by Goldwasser, Micali, and Rivest [33] and its standard security notion.

Definition 2. *A* digital signature scheme *for message space* M *is a triple of algorithms* $\mathsf{Sig} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ *such that*

1. Gen *is the randomized key generation algorithm generating a public (verification) key* pk *and a secret (signing) key* sk *and takes no input.*
2. $\mathsf{Sign}(sk, m)$ *is the randomized signing algorithm outputting a signature* σ *on input message* $m \in M$ *and signing key* sk.
3. $\mathsf{Vrfy}(pk, m, \sigma)$ *is the deterministic verification algorithm outputting either* 0 *or* 1.

We say that a digital signature scheme Sig *is* correct *if for any* $m \in M$, *and* $(pk, sk) \xleftarrow{\$} \mathsf{Gen}$, *it holds that* $\mathsf{Vrfy}(pk, m, \mathsf{Sign}(sk, m)) = 1$.

One-signature-per-message unforgeability of digital signature. We adapt the one-signature-per-message unforgeability defined by Fersch et al. [24]. First, we consider the "strong" variant of the definition given in [24], i.e., a pair (m, σ) output by the adversary is only considered a valid forgery if σ was not returned to the adversary as answer to an signing query m. In the "standard" variant, the pair is considered valid if for message m never a signature has been queried by the adversary. Second, we implement the fact that the adversary only receives one signature per message different to the original definition. Instead of aborting the whole experiment in case the adversary queries a signature for a message that it already received a signature for, we simply return the same signature to the adversary. Therefore, the adversary still gets only one signature per message, but is allowed to query a message multiple times.

We note that, for deterministic signature schemes, the one-signature-per-message security is equivalent to the many-signatures-per-message security.

Definition 3. *Let* $\mathsf{Sig} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ *be a digital signature scheme. Consider the following experiment* $\mathsf{Exp}_{\mathsf{Sig}}^{\mathsf{sEUF\text{-}CMA1}}(\mathcal{A})$ *played between a challenger and an adversary* $\mathcal{A}$:

1. *The challenger initializes the set of chosen-message queries* $\mathcal{Q} := \emptyset$*, generates a fresh key pair* $(pk, sk) \xleftarrow{\$} \mathsf{Gen}$ *and forwards* pk *to the adversary as input.*
2. *The adversary may issue queries to the following oracle adaptively:*
   - $\mathsf{Sign}(m)$*: If* $(m, \sigma) \in \mathcal{Q}$*, the challenger returns* σ*. Otherwise, it returns* $\sigma \xleftarrow{\$} \mathsf{Sign}(sk, m)$ *and adds* (m, σ) *to* $\mathcal{Q}$*.*
3. *Finally, the adversary outputs a candidate forgery* (m, σ) *and the challenger outputs* 1 *if* $\mathsf{Vrfy}(pk, m, \sigma) = 1$ *and* $(m, \sigma) \notin \mathcal{Q}$*, and 0 otherwise.*

We denote the advantage of an adversary $\mathcal{A}$ in forging signatures for Sig in the $\mathsf{sEUF\text{-}CMA1}$ security experiment *by*

$$\mathsf{Adv}^{\mathsf{sEUF\text{-}CMA1}}_{\mathsf{Sig}}(\mathcal{A}) := \Pr\left[\mathsf{Exp}^{\mathsf{sEUF\text{-}CMA1}}_{\mathsf{Sig}}(\mathcal{A}) = 1\right]$$

where $\mathsf{Exp}^{\mathsf{sEUF\text{-}CMA1}}_{\mathsf{Sig}}(\mathcal{A})$ *is as defined above.*

Next, we generalize Definition 3 to the multi-challenge setting. Unforgeability in the multi-challenge setting was proposed by Auerbach et al. [4] and is a generalized version of the standard existential unforgeability against chosen-message attackers notion, in which the adversary has additional access to a "forging oracle" allowing multiple forgery attempts. The adversary wins in this setting if at least one of the forgery attempts is "valid" in the same sense as in the single challenge setting.

Definition 4. *Let* $\mathsf{Sig} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ *be a digital signature scheme. Consider the following experiment* $\mathsf{Exp}^{\mathsf{msEUF\text{-}CMA1}}_{\mathsf{Sig}}(\mathcal{A})$ *played between a challenger and an adversary* $\mathcal{A}$*:*

1. *The challenger initializes the set of chosen-message queries* $\mathcal{Q} := \emptyset$ *and the winning flag* $\mathsf{win} := 0$*. Then, it generates a fresh key pair* $(pk, sk) \xleftarrow{\$} \mathsf{Gen}$ *and forwards* pk *to the adversary as input.*
2. *The adversary may issue queries to the following oracles adaptively:*
   - $\mathsf{Sign}(m)$*: If* $(m, \sigma) \in \mathcal{Q}$ *for some* σ*, the challenger returns* σ*. Otherwise, it returns* $\sigma \xleftarrow{\$} \mathsf{Sign}(sk, m)$ *and adds* (m, σ) *to* $\mathcal{Q}$*.*
   - $\mathsf{Forge}(m, \sigma)$*: If* $\mathsf{Vrfy}(pk, m, \sigma) = 1$ *and* $(m, \sigma) \notin \mathcal{Q}$*, then set* $\mathsf{win} := 1$*.*
3. *Finally, the adversary halts and the experiment outputs* win*.*

We denote the advantage of an adversary $\mathcal{A}$ in forging signatures for Sig in the $\mathsf{msEUF\text{-}CMA1}$ security experiment *by*

$$\mathsf{Adv}^{\mathsf{msEUF\text{-}CMA1}}_{\mathsf{Sig}}(\mathcal{A}) := \Pr\left[\mathsf{Exp}^{\mathsf{msEUF\text{-}CMA1}}_{\mathsf{Sig}}(\mathcal{A}) = 1\right]$$

where $\mathsf{Exp}^{\mathsf{msEUF\text{-}CMA1}}_{\mathsf{Sig}}(\mathcal{A})$ *is as defined above.*

Many-signatures-per-message unforgeability. The security notions $\mathsf{sEUF\text{-}CMA1}$ and $\mathsf{msEUF\text{-}CMA1}$ defined above can be generalized to the "many-signatures-per-message" setting by dropping the condition that the respective security experiments return σ if the Sign-oracle is queried with a message m such that $(m, \sigma) \in \mathcal{Q}$, i.e., a message m that was already queried before. Without this condition we obtain the standard strong existential unforgeability under chosen-message attacks ($\mathsf{sEUF\text{-}CMA}$) and its multi-challenge variant (as defined in [4]) $\mathsf{msEUF\text{-}CMA}$.

Adversary behavior. In this work we consider adversaries that are not necessarily well-behaved. That is, an adversary $\mathcal{A}$ may, for instance, submit a forgery (m^*, σ^*) such that σ^* was obtained by a signing query m^*. In principle, any such adversary can be converted to a well-behaved adversary by performing "sanity checks" whenever the adversary submits a forgery. This conversion, however, is not memory-tight as it leads to an increase in memory needed to store the set of chosen-message queries $\mathcal{Q}$.

Considering that there might exist adversaries that are not well-behaved but break the security of a signature scheme (e.g., by producing a forgery without knowing whether it is a fresh one), we prefer a stronger security notion and consider *any* adversary rather than restricting our proofs to a class of well-behaved adversaries. For a more detailed discussion on this topic, we refer the reader to [4, Section 2.3].

3 From the Single-Challenge Setting to the Multi-Challenge Setting

In this section, we will describe a generic construction of a reduction in the multi-challenge setting, based on any "canonical" reduction in the single-challenge setting.

3.1 Non-Interactive Computational Assumptions

The following definition of a non-interactive computational assumptions is based on the corresponding definition by Bader et al. [6], which is originally due to Abe et al. [3]. It captures both "search problems", such as CDH, and "decisional problems", such as DDH. We focus on *non-interactive* computational hardness assumptions, for the following reasons. First, these may be considered the most "interesting" hardness assumption when (memory) tightness is considered. Second, it makes the definitions and proofs significantly cleaner, and therefore makes it easier to understand and verify the core technical ideas and approach.

Definition 5. *A* non-interactive computational assumption *is defined as the tuple* $\Lambda = (\mathsf{InstGen}, \mathsf{V}, \mathsf{U})$*, where*

1. $(\phi, \omega) \xleftarrow{\$} \mathsf{InstGen}(1^\lambda)$*:* $\mathsf{InstGen}$ *is the probabilistic instance generation algorithm that takes as input a security parameter* 1^λ*, and outputs a problem instance* ϕ *and a witness* ω*.*
2. $0/1 \coloneqq \mathsf{V}(\phi, \omega, \rho)$*:* V *is the deterministic verification algorithm that takes as input a problem instance* ϕ*, a witness* ω *and a candidate solution* ρ*, and outputs* 0 *or* 1*. We say that* ρ *is a correct solution for* ϕ *if* $\mathsf{V}(\phi, \omega, \rho) = 1$*.*
3. $\rho \xleftarrow{\$} \mathsf{U}(\phi)$*:* U *is a probabilistic algorithm that on input* ϕ *outputs a candidate solution* ρ*.*

We define the advantage of an adversary $\mathcal{R}$ *breaking* Λ *as*

$$\mathsf{Adv}^{\mathsf{NICA}}_{\Lambda,\lambda}(\mathcal{R}) \coloneqq \left| \Pr\left[\mathsf{Exp}^{\mathsf{NICA}}_{\Lambda,\lambda}(\mathcal{R}) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{NICA}}_{\Lambda,\lambda}(\mathsf{U}) = 1\right] \right|$$

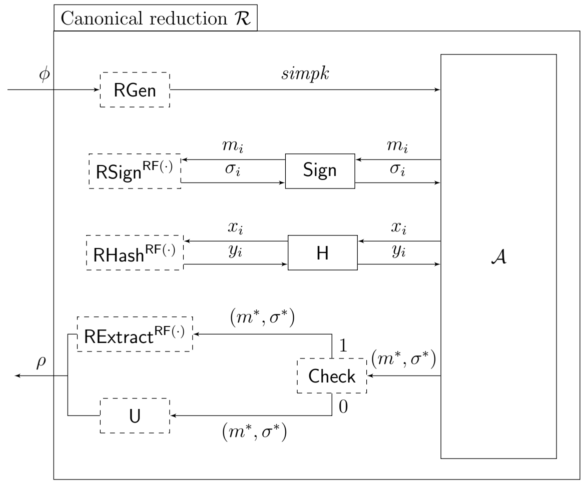

Fig. 1. Canonical reduction $\mathcal{R}$ from sEUF-CMA1-security of a signature scheme Sig to a computational assumption Λ with black-box access to an adversary $\mathcal{A}$. Check is a shorthand defined as $\mathsf{Check}(m^*, \sigma^*) = 1 \iff \mathsf{Sig.Vrfy}(simpk, m^*, \sigma^*) = 1 \wedge \sigma^* \neq \mathsf{RSign}^{\mathsf{RF}(\cdot)}(simsk, m^*)$ determining the algorithm to compute the final solution. For a complete formal definition, see Definition 6.

where the experiment $\mathsf{Exp}^{\mathsf{NICA}}_{\Lambda,\lambda}(\mathcal{A})$ *generates* $(\phi, \omega) \xleftarrow{\$} \mathsf{InstGen}(1^\lambda)$*, runs* $\rho \xleftarrow{\$} \mathcal{A}(\phi)$ *and returns* $\mathsf{V}(\phi, \omega, \rho)$*.*

Intuitively, U can be seen as the "trivial" solution strategy. For example, if Λ is a decisional problem, such as DDH, U usually would output a uniformly random bit such that $\Pr\left[\mathsf{Exp}^{\mathsf{NICA}}_{\Lambda,\lambda}(\mathsf{U}) = 1\right] = \frac{1}{2}$. Then, $\mathsf{Adv}^{\mathsf{NICA}}_{\Lambda,\lambda}(\mathcal{R})$ basically defines the "bit-guessing advantage" against Λ. For a search problem, such as CDH, U would output a constant symbol such that $\Pr\left[\mathsf{Exp}^{\mathsf{NICA}}_{\Lambda,\lambda}(\mathsf{U}) = 1\right] = 0$. Then, $\mathsf{Adv}^{\mathsf{NICA}}_{\Lambda,\lambda}(\mathcal{R})$ corresponds to the probability of $\mathcal{R}$ finding a solution ρ for the given problem instance ϕ.

3.2 Canonical Reductions

We introduce the notion of a *canonical reduction*, which essentially defines an abstract pattern of a reduction which is "compatible" with our approach to prove memory-tight security. Many security proofs of signature schemes can be explained as canonical reductions, we will show some concrete examples below. We focus on reductions from sEUF-CMA1-security to a non-interactive computational assumption Λ (as defined in Section 3.1) in both standard model and random oracle model. For an illustration of a canonical reduction, see Figure 1.

Definition 6. *Let* Sig *be a signature scheme and let* Λ *be a non-interactive computational assumption. Let* (RGen, RF, RSign, RExtract, RHash) *be the following algorithms that are implemented by a canonical reduction:*

1. $(simpk, simsk) \xleftarrow{\$} \mathsf{RGen}(\phi)$*:* RGen *is the probabilistic reduction key generation algorithm that takes as input an instance* ϕ *of* Λ*, and outputs a simulated public key* $simpk$ *and a simulation secret key* $simsk$*.*
2. $(r_{\mathsf{RSign}}, r_{\mathsf{RExtract}}, r_{\mathsf{RHash}}) \xleftarrow{\$} \mathsf{RF}(x)$*:* RF *is a stateful probabilistic algorithm simulating a truly random function with domain* $\{0,1\}^*$ *and range* $\mathsf{Coins}_{\mathsf{RSign}} \times \mathsf{Coins}_{\mathsf{RExtract}} \times \mathsf{Coins}_{\mathsf{RHash}}$ *using a lazily sampled random table, where* $\mathsf{Coins}_{\mathsf{RSign}}$*,* $\mathsf{Coins}_{\mathsf{RExtract}}$*, and* $\mathsf{Coins}_{\mathsf{RHash}}$ *are sets for random coins of* RSign*,* $\mathsf{RExtract}$ *and* RHash*, respectively.*[4]

   Remark 7. Intuitively, RF *has the following purpose. We will below define algorithms* RSign*,* $\mathsf{RExtract}$*, and* RHash*, which are used by the reduction to simulate signatures, extract from a forgery, and possibly to simulate a random oracle (if in the random oracle model), respectively. We require these algorithms to be* stateless *and* deterministic*, since this will be necessary for our construction of a memory-tight reduction. At the same time, we do not want the algorithms* RSign*,* $\mathsf{RExtract}$ *and,* RHash *to be completely independent of each other. For example, the simulation of a signature by* RSign *may have to be consistent with the random oracle implemented by* RHash*. We ensure this consistency by giving all oracles access to the* same *truly random function simulation algorithm* RF*. The algorithms of the canonical reduction are required to achieve consistency by only having access to* RF*. We will show below that this indeed holds for many standard security proofs for signature schemes.*

3. $\sigma := \mathsf{RSign}^{\mathsf{RF}(\cdot)}(simsk, m)$*:* RSign *is the deterministic signature simulation algorithm with access to the algorithm* RF *that takes as input the simulation secret key* $simsk$ *and a message* m*, and outputs a simulated signature* σ*.*[5]
4. $\rho := \mathsf{RExtract}^{\mathsf{RF}(\cdot)}(simsk, (m^*, \sigma^*))$*:* $\mathsf{RExtract}$ *is the deterministic problem solution extraction algorithm with access to the algorithm* RF *that takes as input a forgery* (m^*, σ^*)*, and outputs an extracted solution* ρ*.*
5. $y := \mathsf{RHash}^{\mathsf{RF}(\cdot)}(simsk, x)$*:* RHash *is the deterministic hash simulation algorithm with access to the algorithm* RF *that takes as input an argument* x*, and outputs a simulated hash image* y*.*

We call an algorithm $\mathcal{R}$ *with black-box access to any adversary* $\mathcal{A}$*, write* $\mathcal{R}^{\mathcal{A}}$*, a* (ℓ, δ)-canonical reduction *from* $\mathsf{sEUF\text{-}CMA1}$ *to* Λ *if* $\mathcal{R}$ *satisfies the following properties.*

1. *The reduction* $\mathcal{R}$ *proceeds as follows:*

[4] We note that algorithm RF is part of the canonical reduction. Another option would be providing the canonical reduction with an external random function oracle. We choose the former characterization because it naturally includes the memory consumption of the random table when considering the overall memory consumption of the canonical reduction.

[5] Note that the output signature σ is not necessarily a valid signature of Sig with respect to $simpk$.

(a) *When receiving a problem instance ϕ, the reduction $\mathcal{R}$ uses* $\mathsf{RGen}(\phi)$ *to simulate a public key simpk of* Sig *and generate the simulation secret key simsk, and starts $\mathcal{A}$ on input simpk.*

(b) *Whenever the adversary $\mathcal{A}$ issues a signing query* $\mathsf{Sign}(m)$*, the reduction simulates the signature σ with* $\sigma := \mathsf{RSign}^{\mathsf{RF}(\cdot)}(\mathit{simsk}, m)$ *and returns σ to $\mathcal{A}$. Note that* RSign *is deterministic, so even if* $\mathsf{Sign}(m)$ *is queried multiple times, the adversary always gets the same signature in return.*

(c) *In case the random oracle model (ROM) is considered, the reduction also needs to be able to simulate the random oracle. To this end, the reduction $\mathcal{R}$ answers a random oracle query x by running* $y := \mathsf{RHash}^{\mathsf{RF}(\cdot)}(\mathit{simsk}, x)$ *and returns y.*

(d) *When the adversary $\mathcal{A}$ outputs a candidate forgery (m^*, σ^*), the reduction $\mathcal{R}$ first tests whether it is a valid forgery by checking*

$$\mathsf{Sig.Vrfy}(\mathit{simpk}, m^*, \sigma^*) = 1 \wedge \sigma^* \neq \mathsf{RSign}^{\mathsf{RF}(\cdot)}(\mathit{simsk}, m^*).$$

Intuitively, the second check is the main leverage to "recognize" new signatures. If the checks pass, then we know that (m^*, σ^*) is valid and not *the signature that $\mathcal{R}$ would have simulated. Then $\mathcal{R}$ uses* $\mathsf{RExtract}$ *to extract a solution ρ to the underlying problem Λ with*

$$\rho := \mathsf{RExtract}^{\mathsf{RF}(\cdot)}(\mathit{simsk}, (m^*, \sigma^*)).$$

If the checks fail, $\mathcal{R}$ runs $\rho \stackrel{\$}{\leftarrow} \mathsf{U}(\phi)$*. Finally, $\mathcal{R}$ outputs ρ as the solution to the problem instance ϕ.*

2. *We require that $\mathcal{R}$ is a "valid" reduction from* $\mathsf{sEUF\text{-}CMA1}$*-security to a non-interactive computational assumption Λ. That is, for any adversary $\mathcal{A}$, we have*
$$\mathsf{Adv}^{\mathsf{NICA}}_{\Lambda,\lambda}\left(\mathcal{R}^{\mathcal{A}}\right) \geq \frac{1}{\ell}\mathsf{Adv}^{\mathsf{sEUF\text{-}CMA1}}_{\mathsf{Sig}}(\mathcal{A}) - \delta.$$

Remark 8. If $\mathcal{R}$ is canonical, q_{S} is the upper bound of the number of Sign queries made by the adversary, q_{H} is the upper bound of the number of random oracle queries and q_{RF} is an upper bound of the number of evaluations of RF, then we obtain that

$$\begin{aligned}\mathbf{LocalTime}\left(\mathcal{R}^{\mathcal{A}}\right) \approx \mathbf{LocalTime}(\mathcal{A}) &+ \mathbf{Time}(\mathsf{RGen}) + q_{\mathsf{S}} \cdot \mathbf{Time}(\mathsf{RSign}) \\ &+ q_{\mathsf{H}} \cdot \mathbf{Time}(\mathsf{RHash}) + \mathbf{Time}(\mathsf{Sig.Vrfy}) \\ &+ \max\{\mathbf{Time}(\mathsf{RExtract}), \mathbf{Time}(\mathsf{U})\} + q_{\mathsf{RF}} \cdot \mathbf{Time}(\mathsf{RF}),\end{aligned} \tag{1}$$

and that

$$\begin{aligned}\mathbf{LocalMem}\left(\mathcal{R}^{\mathcal{A}}\right) = \mathbf{LocalMem}(\mathcal{A}) &+ \mathbf{Mem}(\mathsf{RGen}) + \mathbf{Mem}(\mathsf{RSign}) \\ &+ \mathbf{Mem}(\mathsf{RHash}) + \mathbf{Mem}(\mathsf{Sig.Vrfy}) + \mathbf{Mem}(\mathsf{RExtract}) \\ &+ \mathbf{Mem}(\mathsf{U}) + \mathbf{Mem}(\mathsf{RF}).\end{aligned} \tag{2}$$

Note that by design of the canonical reduction the only common state of the algorithms RGen, RSign, RHash and $\mathsf{RExtract}$ is the random table (whose size grows linearly with the number of different queries) in the random function simulation algorithm RF. Otherwise, these algorithms are stateless. This will be the main leverage to achieve memory-tightness, since the random function can be implemented memory-efficiently with a pseudorandom function.

3.3 Multi-Challenge Security for Canonical Reductions

Next, we show how to transform any canonical reduction in the *single*-challenge setting to another reduction in the *multi*-challenge setting. Formally, consider the following theorem.

Theorem 9. *Let* Sig *be a digital signature scheme and let* Λ *be a non-interactive computational assumption. Suppose* $\mathcal{R}$ *is a* (ℓ, δ)*-canonical reduction from the* $\mathsf{sEUF\text{-}CMA1}$*-security of* Sig *to* Λ *and* $\mathsf{PRF}\colon \{0,1\}^\lambda \times \{0,1\}^* \to \mathsf{Coins}_{\mathsf{RSign}} \times \mathsf{Coins}_{\mathsf{RExtract}} \times \mathsf{Coins}_{\mathsf{RHash}}$ *is a pseudorandom function. Using* $\mathcal{R}$ *and* PRF*, we can build another reduction* $\mathcal{R}'$ *from the* $\mathsf{msEUF\text{-}CMA1}$*-security of* Sig *to* Λ *such that for any adversary* $\mathcal{A}'$ *attacking the* $\mathsf{msEUF\text{-}CMA1}$*-security of* Sig*, there exists an adversary* $\mathcal{B}$ *so that*

$$\mathsf{Adv}^{\mathsf{NICA}}_{\Lambda,\lambda}\left(\mathcal{R}'^{\mathcal{A}'}\right) \geq \frac{1}{\ell} \cdot \mathsf{Adv}^{\mathsf{msEUF\text{-}CMA1}}_{\mathsf{Sig}}(\mathcal{A}') - \mathsf{Adv}^{\mathsf{PRF\text{-}sec}}_{\mathsf{PRF}}\left(\mathcal{B}^{\mathcal{A}'}\right) - \delta. \tag{3}$$

Furthermore,

$$\begin{aligned}
\mathbf{LocalTime}(\mathcal{R}'^{\mathcal{A}'}) \approx\ & \mathbf{LocalTime}\,(\mathcal{A}') + \mathbf{Time}(\mathsf{RGen}) + (q_{\mathsf{S}} + q_{\mathsf{F}}) \cdot \mathbf{Time}(\mathsf{RSign}) \\
& + q_{\mathsf{H}} \cdot \mathbf{Time}(\mathsf{RHash}) + q_{\mathsf{F}} \cdot \mathbf{Time}(\mathsf{Sig.Vrfy}) \\
& + \max\{\mathbf{Time}(\mathsf{RExtract}), \mathbf{Time}(\mathsf{U})\} + q_{\mathsf{RF}} \cdot \mathbf{Time}(\mathsf{PRF}), \\
\mathbf{LocalMem}(\mathcal{R}'^{\mathcal{A}'}) =\ & \mathbf{LocalMem}\,(\mathcal{A}') + \mathbf{Mem}(\mathsf{RGen}) + \mathbf{Mem}(\mathsf{RSign}) \\
& + \mathbf{Mem}(\mathsf{RHash}) + \mathbf{Mem}(\mathsf{Sig.Vrfy}) + \mathbf{Mem}(\mathsf{RExtract}) \\
& + \mathbf{Mem}(\mathsf{U}) + \mathbf{Mem}(\mathsf{PRF}) + 1,
\end{aligned} \tag{4}$$

and

$$\begin{aligned}
\mathbf{LocalTime}(\mathcal{B}^{\mathcal{A}'}) \approx\ & \mathbf{LocalTime}\,(\mathcal{A}') + \mathbf{Time}(\mathsf{RGen}) + (q_{\mathsf{S}} + q_{\mathsf{F}}) \cdot \mathbf{Time}(\mathsf{RSign}) \\
& + q_{\mathsf{H}} \cdot \mathbf{Time}(\mathsf{RHash}) + q_{\mathsf{F}} \cdot \mathbf{Time}(\mathsf{Sig.Vrfy}) \\
& + \max\{\mathbf{Time}(\mathsf{RExtract}), \mathbf{Time}(\mathsf{U})\} + \mathbf{Time}(\mathsf{InstGen}) \\
& + \mathbf{Time}(\mathsf{V}), \\
\mathbf{LocalMem}(\mathcal{B}^{\mathcal{A}'}) =\ & \mathbf{LocalMem}\,(\mathcal{A}') + \mathbf{Mem}(\mathsf{RGen}) + \mathbf{Mem}(\mathsf{RSign}) \\
& + \mathbf{Mem}(\mathsf{RHash}) + \mathbf{Mem}(\mathsf{Sig.Vrfy}) + \mathbf{Mem}(\mathsf{RExtract}) \\
& + \mathbf{Mem}(\mathsf{U}) + \mathbf{Mem}(\mathsf{InstGen}) + \mathbf{Mem}(\mathsf{V}).
\end{aligned}$$

where q_{F} *is the number of* Forge *queries made by* $\mathcal{A}'$*,* q_{S} *is the number of* Sign *queries made by* $\mathcal{A}'$*,* q_{H} *is the numbers of queries made to the random oracle*[6]*, and* q_{RF} *is an upper bound of the number of evaluations of* RF*.*

[6] If the reduction is not in the ROM, then $q_{\mathsf{H}} = 0$ holds.

Remark 10. For any $\mathsf{sEUF\text{-}CMA1}$ adversary $\mathcal{A}$ and any $\mathsf{msEUF\text{-}CMA1}$ adversary $\mathcal{A}'$, if we define the memory overhead of $\mathcal{R}'$ ($\mathcal{R}$) as

$$\Delta(\mathcal{R}') := \mathbf{LocalMem}(\mathcal{R}'^{\mathcal{A}'}) - \mathbf{LocalMem}(\mathcal{A}')$$
$$\Delta(\mathcal{R}) := \mathbf{LocalMem}(\mathcal{R}^{\mathcal{A}}) - \mathbf{LocalMem}(\mathcal{A}).$$

Then, from Equations (2) and (4), we have that,

$$\Delta(\mathcal{R}') - \Delta(\mathcal{R}) = \mathbf{Mem}(\mathsf{PRF}) + 1 - \mathbf{Mem}(\mathsf{RF}).$$

More intuitively speaking, this means that reduction $\mathcal{R}'$ does not use memory to keep a random function RF whose random table grows linearly with the number of different queries, but instead it uses some small amount of memory to store a PRF key and run the PRF. Furthermore, the algorithms in $\mathcal{R}'$ (RGen, RSign, RHash, $\mathsf{RExtract}$, $\mathsf{Sig.Vrfy}$, PRF and U) are stateless and their memory usage is independent of the number of queries made by adversary. Thus, the memory overhead of $\mathcal{R}'$, i.e., $\Delta(\mathcal{R}')$ will also be independent of the adversary, especially independent of q_{S}.

Remark 11. Equation (3) is equivalent to

$$\mathsf{Adv}_{\mathsf{Sig}}^{\mathsf{msEUF\text{-}CMA1}}(\mathcal{A}') \leq \ell \cdot \left(\mathsf{Adv}_{\Lambda,\lambda}^{\mathsf{NICA}}\left(\mathcal{R}'^{\mathcal{A}'}\right) + \mathsf{Adv}_{\mathsf{PRF}}^{\mathsf{PRF\text{-}sec}}\left(\mathcal{B}^{\mathcal{A}'}\right) + \delta\right).$$

It shows that the $\mathsf{msEUF\text{-}CMA1}$ security of Sig builds upon both the security of NICA and the pseudorandomness of PRF. If ℓ is a constant, δ is a negligible value which is independent of the number of queries made by the adversary and PRF is memory-tightly secure, then the $\mathsf{msEUF\text{-}CMA1}$ security of Sig is tight in both working factor and memory. (See Section 5 for more discussions about concrete applications.)

Proof (of Theorem 9). Since $\mathcal{R}$ is a canonical reduction, we know that there are algorithms $(\mathsf{RGen}, \mathsf{RSign}, \mathsf{RExtract}, \mathsf{RHash})$. Using these algorithms and a pseudorandom function, we construct another reduction $\mathcal{R}'$ which transfers any $\mathsf{msEUF\text{-}CMA1}$ adversary $\mathcal{A}'$ to a hard problem solver of Λ.

Construction of $\mathcal{R}'$. The reduction $\mathcal{R}'$ receives as input an instance ϕ of Λ and simulates the experiment $\mathsf{Exp}_{\mathsf{Sig}}^{\mathsf{msEUF\text{-}CMA1}}(\mathcal{A}')$ for $\mathcal{A}'$. To this end, it first runs $(simpk, simsk) \xleftarrow{\$} \mathsf{RGen}(\phi)$ to obtain a simulated public key $simpk$ for the signature scheme Sig. Note that this is exactly the same as what $\mathcal{R}$ would do.

In contrast to $\mathcal{R}$, $\mathcal{R}'$ does not simulate a random function with algorithm RF. Instead, it chooses a uniform key $k \xleftarrow{\$} \{0,1\}^\lambda$ for a pseudorandom function $\mathsf{PRF}\colon \{0,1\}^\lambda \times \{0,1\}^* \to \mathsf{Coins}_{\mathsf{RSign}} \times \mathsf{Coins}_{\mathsf{RExtract}} \times \mathsf{Coins}_{\mathsf{RHash}}$ and uses PRF as a replacement.

$\mathcal{A}'$ then receives as input the simulated public key $simpk$ and gets access to the signing oracle Sign, the random oracle (if ROM is considered) and the "forgery attempt" oracle Forge. To simulate these oracles for $\mathcal{A}'$, the reduction $\mathcal{R}'$ does the following:

Sign-oracle. Upon receiving a signature query $\mathsf{Sign}(m)$ for some message $m \in M$, the reduction $\mathcal{R}'$ runs $\mathcal{R}$'s signature simulation algorithm with oracle access to PRF, i.e., $\sigma \coloneqq \mathsf{RSign}^{\mathsf{PRF}(k,\cdot)}(simsk, m)$. Then it returns σ to $\mathcal{A}'$. Note that the same signature will be returned if the same message is queried multiple times since RSign is deterministic.

Random oracle. $\mathcal{R}'$ answers a random oracle query x by running RHash with oracle access to PRF, i.e., $y \coloneqq \mathsf{RHash}^{\mathsf{PRF}(k,\cdot)}(simsk, x)$ and returns y.

Forge-oracle. Upon receiving a forgery attempt (m^*, σ^*), the reduction $\mathcal{R}'$ at first checks whether

$$\mathsf{Sig.Vrfy}(simpk, m^*, \sigma^*) = 1 \quad \text{and} \quad \sigma^* \neq \mathsf{RSign}^{\mathsf{PRF}(k,\cdot)}(simsk, m^*)$$

In case both checks pass, the reduction $\mathcal{R}'$ attempts to extract a solution ρ for the problem instance ϕ from the forgery at hand by running $\rho \coloneqq \mathsf{RExtract}^{\mathsf{PRF}(k,\cdot)}(simsk, (m^*, \sigma^*))$. Then $\mathcal{R}'$ returns ρ and halts.
In case any of the previous two checks failed, $\mathcal{R}'$ continues to simulate $\mathcal{A}'$. If the adversary $\mathcal{A}'$ fails to output any forgery attempt (m^*, σ^*) that can pass the checks throughout the whole simulation process, $\mathcal{R}'$ runs $\rho \xleftarrow{\$} \mathsf{U}(\phi)$ and outputs ρ.

Note that $\mathcal{R}'$ proceeds exactly as $\mathcal{R}$ but it uses a pseudorandom function instead of a truly random function and it needs to handle at most q_{F} forgery attempts as opposed to just one. Therefore, the running time of $\mathcal{R}'$ is the running time of $\mathcal{R}$ as given in Remark 8, replacing $\mathbf{Time}(\mathsf{RF})$ by $\mathbf{Time}(\mathsf{PRF})$ plus the time required to simulate the additional $q_{\mathsf{F}} - 1$ Forge-queries, namely $(q_{\mathsf{F}} - 1) \cdot (\mathbf{Time}(\mathsf{Vrfy}) + \mathbf{Time}(\mathsf{RSign}))$. This yields the time given in Theorem 9.

Similarly, the memory consumption of $\mathcal{R}'$ is the memory consumed by $\mathcal{R}$ as given in Remark 8, but instead of storing the random table in RF, $\mathcal{R}'$ needs to store the function description of PRF and its corresponding key, which again yields the values given in Theorem 9. In particular, note that the memory consumed by $\mathcal{R}'^{\mathcal{A}'}$ is independent of the number of queries made by $\mathcal{A}'$, as the stateful random table is replaced with the stateless keyed PRF PRF.

We complete the proof of Theorem 9 by analyzing the advantage of $\mathcal{R}'$ as follows.

The advantage of $\mathcal{R}'^{\mathcal{A}'}$. In order to analyse the advantage of $\mathcal{R}'^{\mathcal{A}'}$, we first modify the reduction $\mathcal{R}'$ to get a new reduction $\mathcal{R}_1$. More precisely, $\mathcal{R}_1$ is exactly $\mathcal{R}'$ except that it uses a random function RF instead of a pseudorandom function PRF.

We can easily build an adversary $\mathcal{B}$ and show that

$$\mathsf{Adv}^{\mathsf{NICA}}_{\Lambda,\lambda}\left(\mathcal{R}'^{\mathcal{A}'}\right) \geq \mathsf{Adv}^{\mathsf{NICA}}_{\Lambda,\lambda}\left(\mathcal{R}_1^{\mathcal{A}'}\right) - \mathsf{Adv}^{\mathsf{PRF\text{-}sec}}_{\mathsf{PRF}}\left(\mathcal{B}^{\mathcal{A}'}\right). \tag{5}$$

The construction of $\mathcal{B}$ is straightforward. It generates the problem instance together with its witness using $(\phi, \omega) \xleftarrow{\$} \mathsf{InstGen}(1^\lambda)$. Then it simulates the above reductions and interacting with $\mathcal{A}'$ by forwarding all the input to RF/PRF to its own challenger. If the reduction outputs a solution ρ, $\mathcal{B}$ runs the algorithm V and outputs $\mathsf{V}(\phi, \omega, \rho)$. Thus, Equation (5) holds and the running time and memory consumption of $\mathcal{B}$ follows the equations in Theorem 9.

Next we modify $\mathcal{R}_1$ again to get $\mathcal{R}_2$. $\mathcal{R}_2$ is identical to $\mathcal{R}_1$ except that it logs all the chosen message queries with their respective signatures in the set $\mathcal{Q}$ and it replaces the check in Forge-oracle from

$$\mathsf{Sig.Vrfy}(\mathit{simpk}, m^*, \sigma^*) = 1 \wedge \sigma^* \neq \mathsf{RSign}^{\mathsf{RF}(\cdot)}(\mathit{simsk}, m^*)$$

to the check

$$\mathsf{Sig.Vrfy}(\mathit{simpk}, m^*, \sigma^*) = 1 \wedge \sigma^* \neq \mathsf{RSign}^{\mathsf{RF}(\cdot)}(\mathit{simsk}, m^*) \wedge (m^*, \sigma^*) \notin \mathcal{Q}.$$

Note that the added check $(m^*, \sigma^*) \notin \mathcal{Q}$ is redundant because every (m, σ) pair in $\mathcal{Q}$ has the property that $\sigma = \mathsf{RSign}^{\mathsf{PRF}(\cdot)}(\mathit{simsk}, m)$. Thus, we have that

$$\mathsf{Adv}^{\mathsf{NICA}}_{\Lambda,\lambda}\left(\mathcal{R}_1^{\mathcal{A}'}\right) = \mathsf{Adv}^{\mathsf{NICA}}_{\Lambda,\lambda}\left(\mathcal{R}_2^{\mathcal{A}'}\right). \tag{6}$$

Next, we construct a single-challenge $\mathsf{sEUF\text{-}CMA1}$-adversary $\widetilde{\mathcal{A}}$ that combines the multi-challenge $\mathcal{A}'$ with the check $\mathsf{Sig.Vrfy}(pk, m^*, \sigma^*) = 1 \wedge (m^*, \sigma^*) \notin \mathcal{Q}$. More precisely, after getting the public key pk, $\widetilde{\mathcal{A}}$ simulates $\mathcal{A}'$ and keep log of the set $\mathcal{Q}$ itself. Whenever $\mathcal{A}'$ submits a Forge-query, $\widetilde{\mathcal{A}}$ checks whether $\mathsf{Sig.Vrfy}(pk, m^*, \sigma^*) = 1 \wedge (m^*, \sigma^*) \notin \mathcal{Q}$. If the check does not pass, $\widetilde{\mathcal{A}}$ continues the simulation of $\mathcal{A}'$. And $\widetilde{\mathcal{A}}$ outputs the first forgery that can pass this check as its own forgery attempt. After that, $\widetilde{\mathcal{A}}$ terminates. Note that $\widetilde{\mathcal{A}}$ can perform the checks efficiently because it knows the public key pk and can log the set $\mathcal{Q}$ itself.

We can obtain an important observation on $\widetilde{\mathcal{A}}$: the game that is played between $\mathcal{R}_2$ and the multi-challenge adversary $\mathcal{A}'$ distributes identically with the game that is played between the canonical reduction $\mathcal{R}$ and the single-challenge adversary $\widetilde{\mathcal{A}}$. Thus, we have that

$$\mathsf{Adv}^{\mathsf{NICA}}_{\Lambda,\lambda}(\mathcal{R}_2^{\mathcal{A}'}) = \mathsf{Adv}^{\mathsf{NICA}}_{\Lambda,\lambda}(\mathcal{R}^{\widetilde{\mathcal{A}}}).$$

Furthermore, we know that $\widetilde{\mathcal{A}}$ wins the (single-challenge) $\mathsf{sEUF\text{-}CMA1}$ game if and only if $\mathcal{A}'$ wins the (multi-challenge) $\mathsf{msEUF\text{-}CMA1}$ game because of the check $\mathsf{Sig.Vrfy}(\mathit{simpk}, m^*, \sigma^*) = 1 \wedge (m^*, \sigma^*) \notin \mathcal{Q}$. So, we have that

$$\mathsf{Adv}^{\mathsf{sEUF\text{-}CMA1}}_{\mathsf{Sig}}(\widetilde{\mathcal{A}}) = \mathsf{Adv}^{\mathsf{msEUF\text{-}CMA1}}_{\mathsf{Sig}}(\mathcal{A}').$$

Since $\mathcal{R}$ is canonical, we have that

$$\begin{aligned} \mathsf{Adv}^{\mathsf{NICA}}_{\Lambda,\lambda}(\mathcal{R}_2^{\mathcal{A}'}) = \mathsf{Adv}^{\mathsf{NICA}}_{\Lambda,\lambda}(\mathcal{R}^{\widetilde{\mathcal{A}}}) &\geq \frac{1}{\ell}\mathsf{Adv}^{\mathsf{sEUF\text{-}CMA1}}_{\mathsf{Sig}}(\widetilde{\mathcal{A}}) - \delta \\ &= \frac{1}{\ell}\mathsf{Adv}^{\mathsf{msEUF\text{-}CMA1}}_{\mathsf{Sig}}(\mathcal{A}') - \delta. \end{aligned} \tag{7}$$

Combining Equations (5) to (7), we have that

$$\mathsf{Adv}^{\mathsf{NICA}}_{\Lambda,\lambda}(\mathcal{R}'^{\mathcal{A}'}) \geq \frac{1}{\ell}\mathsf{Adv}^{\mathsf{msEUF\text{-}CMA1}}_{\mathsf{Sig}}(\mathcal{A}') - \delta - \mathsf{Adv}^{\mathsf{PRF\text{-}sec}}_{\mathsf{PRF}}(\mathcal{B}^{\mathcal{A}'}),$$

and the theorem follows. $\square$

4 From msEUF-CMA1 Security to msEUF-CMA Security

So far we have shown how any signature scheme that can be proven sEUF-CMA1-secure (i.e., single-challenge and one-signature-per-message) via a canonical reduction to some computational problem, can be proven msEUF-CMA1-secure (i.e., *multi-challenge* and one-signature-per-message) in a memory-tight way. In this section, we extend our approach and present a generic transform, which "memory-tightly lifts" *any* signature scheme from msEUF-CMA1 security (i.e., multi-challenge and one-signature-per-message) to the desired msEUF-CMA security (i.e., multi-challenge and *many-signatures-per-message*).

Intuition. The core idea of this transform is to sign a message together with some randomly-chosen nonce n. Intuitively, this nonce "expands" the set of valid signatures for a given message. While this transform is straightforward, we see value to make it explicit.

Transform. Let $\lambda \in \mathbb{N}$ and let $\mathsf{Sig}' = (\mathsf{Gen}', \mathsf{Sign}', \mathsf{Vrfy}')$ be a signature scheme. We construct a new signature scheme $\mathsf{Sig} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ as follows:

Key generation. Gen behaves exactly like Gen'.

Signing. Sign takes as input the secret key sk and a message m. It samples a nonce $n \xleftarrow{\$} \{0,1\}^\lambda$, computes $\sigma' \xleftarrow{\$} \mathsf{Sign}'(sk, m \,\|\, n)$, and returns $\sigma = (\sigma', n)$.

Verification. Vrfy takes as input a public key pk, a message m, and a signature $\sigma = (\sigma', n)$. It computes and returns $\mathsf{Sig}'.\mathsf{Vrfy}(pk, m \,\|\, n, \sigma')$.

Theorem 12. *From each adversary $\mathcal{A}$ breaking the* msEUF-CMA*-security of the above signature scheme* Sig *(with q_s signing queries), we can construct an adversary $\mathcal{B}$ such that* $\mathsf{Adv}^{\mathsf{msEUF\text{-}CMA}}_{\mathsf{Sig}}(\mathcal{A}) \le \mathsf{Adv}^{\mathsf{msEUF\text{-}CMA1}}_{\mathsf{Sig}'}(\mathcal{B}) + \frac{q_s^2}{2^\lambda}$ *and*

$$\mathbf{LocalTime}(\mathcal{B}) \approx \mathbf{LocalTime}(\mathcal{A}) \quad \textit{and} \quad \mathbf{LocalMem}(\mathcal{B}) = \mathbf{LocalMem}(\mathcal{A}).$$

The proof of Theorem 12 is straightforward and we provide it in the full version [21].

5 Applications

In this section, we present how the results of Sections 3 and 4 can be used to yield memory-tight strongly unforgeable signatures in the multi-challenge and many-signatures-per-message setting. In Section 5.1, we present a construction based on lossy identification schemes (similar to the construction by Abdalla et al. [1]) and prove its memory-tight security using our results. Then, in Section 5.2, we show how existing signature schemes such as RSA-FDH [10] benefit from our result and evade the existing impossibility results of [4, 51]. In the full version of this paper [21], we show similar results for the Boneh, Lynn, and Shacham signature scheme [14, 15].

We note that, a pseudorandom function is required when applying our results of Sections 3 and 4. In the standard model, we are aware of several pseudorandom functions that achieve almost tight security based on standard assumptions [40, 45, 47]. In the random oracle model, such a pseudorandom function exists unconditionally.

5.1 Memory-Tight Signatures from Lossy Identification Schemes

In this section, we present how to construct memory-tight strongly unforgeable signatures in the multi-challenge and many-signatures-per-message setting based on lossy identification schemes. To this end, we first present a formal definition of lossy identification schemes.

Lossy Identification Schemes. We adapt the definition of a *lossy identification scheme* [1, 2].

Definition 13. *A* lossy identification scheme LID *is a tuple of algorithms*

$$\mathsf{LID} = (\mathsf{LID.Gen}, \mathsf{LID.LossyGen}, \mathsf{LID.Prove}, \mathsf{LID.Vrfy}, \mathsf{LID.Sim})$$

with the following properties.

- $(pk, sk) \xleftarrow{\$} \mathsf{LID.Gen}(1^\lambda)$ *is the normal key generation algorithm. It takes as input the security parameter and outputs a public verification key* pk *and a secret key* sk.
- $pk \xleftarrow{\$} \mathsf{LID.LossyGen}(1^\lambda)$ *is a lossy key generation algorithm that takes the security parameter and outputs a lossy verification key* pk.
- $\mathsf{LID.Prove}$ *is the prover algorithm that is split into two algorithms:*
  - $(\mathsf{cmt}, \mathsf{st}) \xleftarrow{\$} \mathsf{LID.Prove}_1(sk)$ *is a probabilistic algorithm that takes as input the secret key and returns a commitment* cmt *and a state* st.
  - $\mathsf{resp} \xleftarrow{\$} \mathsf{LID.Prove}_2(sk, \mathsf{cmt}, \mathsf{ch}, \mathsf{st})$ *is a deterministic algorithm*[7] *that takes as input the secret key, a commitment* cmt*, a challenge* ch*, a state* st*, and returns a response* resp.
- $\mathsf{LID.Vrfy}(pk, \mathsf{cmt}, \mathsf{ch}, \mathsf{resp}) \in \{0, 1\}$ *is a deterministic verification algorithm that takes a public key, and a conversation transcript (i.e., a commitment, a challenge, and a response) as input and outputs a bit, where* 1 *indicates that the proof is "accepted" and* 0 *"rejected".*

We assume that a public key pk *implicitly defines two sets, the challenge set* CSet *and the response set* RSet.

[7] As far as we know, all the instantiations of lossy identification schemes have a deterministic $\mathsf{LID.Prove}_2$ algorithm. However, if a new instantiation requires randomness, then it can be "forwarded" from $\mathsf{LID.Prove}_1$ in the state variable st. Therefore the requirement that $\mathsf{LID.Prove}_2$ is deterministic is without loss of generality, and only made to simplify our security analysis.

Definition 14. *Let* $\mathsf{LID} = (\mathsf{LID.Gen}, \mathsf{LID.LossyGen}, \mathsf{LID.Prove}, \mathsf{LID.Vrfy}, \mathsf{LID.Sim})$ *defined as above. We call* LID lossy *when the following properties hold:*

- Completeness of normal keys. *Let* $(pk, sk) \xleftarrow{\$} \mathsf{LID.Gen}(1^\lambda)$ *be a key pair and let* $(\mathsf{cmt}, \mathsf{ch}, \mathsf{resp})$ *be an honest transcript (i.e.,* $(\mathsf{cmt}, \mathsf{st}) \xleftarrow{\$} \mathsf{LID.Prove}_1(sk)$*,* $\mathsf{ch} \xleftarrow{\$} \mathsf{CSet}$*, and* $\mathsf{resp} \xleftarrow{\$} \mathsf{LID.Prove}_2(sk, \mathsf{cmt}, \mathsf{ch}, \mathsf{st})$*). We call* LID ρ-complete*, if*

  $$\Pr[\mathsf{LID.Vrfy}(pk, \mathsf{cmt}, \mathsf{ch}, \mathsf{resp}) = 1] \geq \rho(\lambda),$$

  where ρ *is a non-negligible function in* λ*. We call* LID perfectly-complete*, if it is 1-complete.*
- Simulatability of transcripts. *Let* $(pk, sk) \xleftarrow{\$} \mathsf{LID.Gen}(1^\lambda)$ *be a key pair. We call* LID ε_s-simulatable *if* LID.Sim *taking public key* pk*, a challenge* $\mathsf{ch} \in \mathsf{CSet}$ *and a response* $\mathsf{resp} \in \mathsf{RSet}$ *as input, deterministically generates a commitment* cmt *such that* $(\mathsf{cmt}, \mathsf{ch}, \mathsf{resp})$ *is a valid transcript (i.e.,* $\mathsf{LID.Vrfy}(pk, \mathsf{cmt}, \mathsf{ch}, \mathsf{resp}) = 1$*). Furthermore, if* $(\mathsf{ch}, \mathsf{resp})$ *is chosen uniformly random from* $\mathsf{CSet} \times \mathsf{RSet}$*, the distribution of the transcript* $(\mathsf{cmt}, \mathsf{ch}, \mathsf{resp})$ *is statistically indistinguishable (up to an upper bound* ε_s*) from honestly generated transcripts. If* $\varepsilon_s = 0$*, we call* LID perfectly simulatable.
- Indistinguishability of keys. *We define the advantage of an adversary* $\mathcal{A}$ *to break the key-indistinguishability of* LID *as*

  $$\mathsf{Adv}^{\mathsf{IND\text{-}KEY}}_{\mathsf{LID}}(\mathcal{A}) := \left|\Pr\left[\mathcal{A}(pk) = 1\right] - \Pr\left[\mathcal{A}(pk') = 1\right]\right|,$$

  where $(pk, sk) \xleftarrow{\$} \mathsf{LID.Gen}(1^\lambda)$ *and* $pk' \xleftarrow{\$} \mathsf{LID.LossyGen}(1^\lambda)$*, is negligible in* λ*.*
- Lossiness. *Consider the following security experiment* $\mathsf{Exp}^{\mathsf{IMPERSONATE}}_{\mathsf{LID}}(\mathcal{A})$ *described below, played between a challenger and an adversary* $\mathcal{A}$*:*
  1. *The challenger generates a lossy verification key* $pk \xleftarrow{\$} \mathsf{LID.LossyGen}(1^\lambda)$ *and sends it to the adversary* $\mathcal{A}$*.*
  2. *The adversary* $\mathcal{A}$ *may now compute a commitment* cmt *and send it to the challenger. The challenger responds with a random challenge* $\mathsf{ch} \xleftarrow{\$} \mathsf{CSet}$*.*
  3. *Eventually, the adversary* $\mathcal{A}$ *outputs a response* resp*. The challenger outputs* $\mathsf{LID.Vrfy}(pk, \mathsf{cmt}, \mathsf{ch}, \mathsf{resp})$*.*

  We call LID ε_ℓ*-lossy if no computationally unrestricted adversary* $\mathcal{A}$ *wins the above security game with probability*

  $$\Pr[\mathsf{Exp}^{\mathsf{IMPERSONATE}}_{\mathsf{LID}}(\mathcal{A}) = 1] \geq \varepsilon_\ell.$$

Definition 15. *A lossy identification scheme*

$$\mathsf{LID} = (\mathsf{LID.Gen}, \mathsf{LID.LossyGen}, \mathsf{LID.Prove}, \mathsf{LID.Vrfy}, \mathsf{LID.Sim})$$

is commitment-recoverable *if* $\mathsf{LID.Vrfy}(pk, \mathsf{cmt}, \mathsf{ch}, \mathsf{resp})$ *first recomputes* $\mathsf{cmt}' = \mathsf{LID.Sim}(pk, \mathsf{ch}, \mathsf{resp})$ *and then outputs 1 if and only if* $\mathsf{cmt}' = \mathsf{cmt}$*.*

Remark 16. We are aware of five different lossy identification scheme instantiations and they are based on DDH [44], DSDL, Ring-LWE, Subset Sum [1,2] and RSA [35]. As far as we know, all of them are commitment-recoverable. And the schemes based on DDH, DSDL and RSA assumption are perfectly complete and perfectly simulatable.

Memory-Tight Signatures from Lossy Identification Schemes. In the following, we present the construction of the signature scheme based on lossy identification scheme. This construction is slightly different from the construction by Abdalla et al. in [1, 2] and can be seen as a variant of the Fiat-Shamir transform [25]. We show that this construction can be proven strongly unforgeable in the single challenge and one-message-per-signature setting (in the sense of $\mathsf{sEUF\text{-}CMA1}$, see Definition 3) in Theorem 17. This result is not yet memory-tight, but work-factor-tight, as the reduction still needs to do book-keeping for a random function, but does not need to store the set of queried messages and there respective signatures in the set $\mathcal{Q}$ anymore. Based this result, we show how to apply Theorems 9 and 12 to yield strong unforgeability in the multi-challenge and many-signatures-per-message setting (in the sense of $\mathsf{msEUF\text{-}CMA}$), which then will be fully tight, i.e., both work-factor- and memory-tight. [1, 2].

Let $\mathsf{LID} = (\mathsf{LID.Gen}, \mathsf{LID.LossyGen}, \mathsf{LID.Prove}, \mathsf{LID.Vrfy})$ be a lossy identification scheme and let $\mathsf{H} : \{0,1\}^* \to \mathsf{CSet}$. Consider the following digital signature scheme $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$.

Key generation. Algorithm Gen samples a key pair $(pk, sk) \xleftarrow{\$} \mathsf{LID.Gen}(1^\lambda)$.

Signing. The signing algorithm Sign takes as input sk and a message $m \in \{0,1\}^*$. Then, it computes $(\mathsf{cmt}, \mathsf{st}) \xleftarrow{\$} \mathsf{LID.Prove}_1(sk)$, $\mathsf{ch} \coloneqq \mathsf{H}(m, \mathsf{cmt})$ and $\mathsf{resp} \coloneqq \mathsf{LID.Prove}_2(sk, \mathsf{ch}, \mathsf{cmt}, \mathsf{st})$, and outputs the signature $\sigma \coloneqq (\mathsf{ch}, \mathsf{resp})$.

Verification. The verification algorithm Vrfy takes as input a public key pk, message $m \in \{0,1\}^*$, and a signature $\sigma = (\mathsf{ch}, \mathsf{resp})$. It runs the check $\mathsf{LID.Vrfy}(pk, \mathsf{cmt}, \mathsf{ch}, \mathsf{resp})$. More precisely, it first recovers

$$\mathsf{cmt} \coloneqq \mathsf{LID.Sim}(pk, \mathsf{ch}, \mathsf{resp})$$

and then computes $\mathsf{ch}' \coloneqq \mathsf{H}(m, \mathsf{cmt})$ Finally, the reduction outputs 1 if and only if ch equals ch'.

Compared to the signature scheme by Abdalla et al. [1, 2], signature of the above scheme is a pair $(\mathsf{ch}, \mathsf{resp})$ whereas signature in [1, 2] is a pair $(\mathsf{cmt}, \mathsf{resp})$ for a transcript $(\mathsf{cmt}, \mathsf{ch}, \mathsf{resp})$ of the lossy identification scheme. For a concrete instantiation based on DDH assumption, this yields a shorter signature.

Theorem 17. *Let* $\mathsf{H}\colon \{0,1\}^* \to \mathsf{CSet}$ *be modeled as a random oracle and let* LID *be a lossy identification scheme that is commitment-recoverable, perfectly complete,* ε_s*-simulatable and* ε_ℓ*-lossy.*

Then, from each adversary $\mathcal{A}$ *breaking the* $\mathsf{sEUF\text{-}CMA1}$ *security of the above signature scheme, we can construct an adversary* $\mathcal{B}$ *such that*

$$\mathsf{Adv}^{\mathsf{sEUF\text{-}CMA1}}_{\mathsf{Sig}}(\mathcal{A}) \leq \mathsf{Adv}^{\mathsf{IND\text{-}KEY}}_{\mathsf{LID}}(\mathcal{B}) + \frac{1}{|\mathsf{CSet}|} + \frac{1}{|\mathsf{RSet}|} + q_{\mathsf{S}} \cdot \varepsilon_s + q_{\mathsf{H}} \cdot \varepsilon_\ell$$

and

$$\begin{aligned}\mathbf{LocalTime}(\mathcal{B}) \leq{} & \mathbf{LocalTime}(\mathcal{A}) + \mathbf{Time}(\mathsf{LID.LossyGen}) \\ & + (q_s + q_{\mathsf{H}} + 1) \cdot \mathbf{Time}(\mathsf{RF}) + \mathbf{Time}(\mathsf{Sig.Vrfy}), \\ \mathbf{LocalMem}(\mathcal{B}) ={} & \mathbf{LocalMem}(\mathcal{A}) + \mathbf{Mem}(\mathsf{LID.LossyGen}) + \mathbf{Mem}(\mathsf{RF}) \\ & + \mathbf{Mem}(\mathsf{Sig.Vrfy}),\end{aligned}$$

where q_{S} *is the number of* Sign-*queries issued by* $\mathcal{A}$*,* q_{F} *is the number of* Forge-*queries issued by* $\mathcal{A}$ *and* q_{H} *is the number of hash queries throughout the game.*

The proof of Theorem 17 is similar to the proof by Abdalla et al. in [1, 2]. One technical difference it that, in our proof, we need to memory-tightly switch the winning condition in the sEUF-CMA1 game into the checks that a canonical reduction would do. For completeness, we provide the full proof in the full version [21].

Applying Theorem 9. Here, we show how to apply Theorem 9 to lift the security of the LID-based signature scheme to work-factor-tight *and* memory-tight security in the *multi*-challenge and one-per-message setting. To apply the theorem, we show that the adversary $\mathcal{B}$ in Theorem 17 can be "translated" into a canonical reduction $\mathcal{R}_{\mathsf{LID}}$ which satisfies Definition 6.

To this end, we define the canonical reduction $\mathcal{R}_{\mathsf{LID}}$ from sEUF-CMA1-security to the indistinguishability of keys IND-KEY to be the tuple $(\mathsf{RGen}, \mathsf{RF}, \mathsf{RSign}, \mathsf{RExtract}, \mathsf{RHash})$ as follows.

RGen: On input $\phi = pk$, RGen return $(pk, \emptyset)$ where $\emptyset$ denotes the empty word in this context.

RF: On input any string $x \in \{0,1\}^*$, RF simulates a random function using a lazily sampled random table. In the following, we will omit this table and view RF as a random function. Further, for $(r_{\mathsf{RSign}}, r_{\mathsf{RHash}}) := \mathsf{RF}(x)$, we define the short-hands $r_{\mathsf{RSign}} =: \mathsf{RF}(\texttt{"sim"} \,\|\, x)$ and $r_{\mathsf{RHash}} =: \mathsf{RF}(\texttt{"hash"} \,\|\, x)$.

$\mathsf{RSign}^{\mathsf{RF}(\cdot)}$: On input $simsk = \emptyset$ and m, RSign outputs $\sigma = (\mathsf{ch}, \mathsf{resp})$ with $(\mathsf{ch}, \mathsf{resp}) := \mathsf{RF}(\texttt{"sim"} \,\|\, m)$.

$\mathsf{RExtract}^{\mathsf{RF}(\cdot)}$: On input $simsk = \emptyset$ and (m^*, σ^*), RExtract outputs solution $\rho = 1$. Note that by definition $\mathcal{R}_{\mathsf{LID}}$ runs RExtract only if $\mathsf{Vrfy}(pk, m^*, \sigma^*) = 1$ and $\sigma^* = (\mathsf{ch}^*, \mathsf{resp}^*) \neq \mathsf{RSign}(simsk, m^*) = \mathsf{RF}(\texttt{"sim"} \,\|\, m^*)$. Hence, if RExtract is run the queried forgery is valid.

$\mathsf{RHash}^{\mathsf{RF}(\cdot)}$: On input $simsk = \emptyset$ and x, RHash works as follows:
- If x cannot be parsed as $x = m \,\|\, \mathsf{cmt}$, then it returns $\mathsf{RF}(\texttt{"hash"} \,\|\, x)$.
- Otherwise, it parses $m \,\|\, \mathsf{cmt} := x$ and runs $(\mathsf{ch}, \mathsf{resp}) := \mathsf{RF}(\texttt{"sim"} \,\|\, m)$ and then $\mathsf{cmt}' := \mathsf{LID.Sim}(\mathsf{ch}, \mathsf{resp})$.
  - If $\mathsf{cmt} = \mathsf{cmt}'$, then it returns ch.
  - Otherwise, it returns $\mathsf{RF}(\texttt{"hash"} \,\|\, x)$.

According to the results of Theorem 17, we have

$$\mathsf{Adv}^{\mathsf{IND\text{-}KEY}}_{\mathsf{LID}}(\mathcal{R}^{\mathcal{A}}_{\mathsf{LID}}) \geq \mathsf{Adv}^{\mathsf{sEUF\text{-}CMA1}}_{\mathsf{Sig}}(\mathcal{A}) - \frac{1}{|\mathsf{CSet}|} - \frac{1}{|\mathsf{RSet}|} - q_{\mathsf{S}} \cdot \varepsilon_s - q_{\mathsf{H}} \cdot \varepsilon_\ell$$

where all quantities are defined as in Theorem 17 and $\mathsf{Adv}^{\mathsf{IND\text{-}KEY}}_{\mathsf{LID}}(\mathcal{R}^{\mathcal{A}}_{\mathsf{LID}}) = \mathsf{Adv}^{\mathsf{IND\text{-}KEY}}_{\mathsf{LID}}(\mathcal{B})$. Thus, $\mathcal{R}_{\mathsf{LID}}$ fulfills Definition 6, Property 2 with $\ell = 1$ and $\delta = \frac{1}{|\mathsf{CSet}|} + \frac{1}{|\mathsf{RSet}|} + q_{\mathsf{S}} \cdot \varepsilon_s + q_{\mathsf{H}} \cdot \varepsilon_\ell$.

Applying Theorem 12. It remains to lift the security of the LID-based signature scheme from the one-signature-per-message setting to the many-signatures-per-message-setting. This can easily be done, by applying the transform presented in Section 4. As the reduction presented in Theorem 12 preserves the memory-tightness of the one-per-message scheme Sig', we have that the transformed LID-based signature scheme is memory-tightly strongly unforgeable in the multi-challenge and many-signatures-per-message setting.

5.2 On the Memory-Tightness of RSA-FDH

Auerbach et al. [4] show that RSA-FDH can be proven memory-tightly unforgeable in the single-challenge and many-signatures-per-message setting under the RSA assumption. However, due to the existing tightness lower bounds, they did not achieve work-factor-tightness. In this subsection, we first show that RSA-FDH can be proven memory-tightly unforgeable in the *multi-challenge* setting because the reduction by Auerbach et al. satisfies our definition of a canonical reduction. Furthermore, we additionally show that with one extra random bit in the signature, we are able to achieve both memory and working factor tightness together with strong security.

We briefly recall the RSA assumption in the form of a non-interactive computational assumption.

Definition 18. *Let* GenRSA *be an algorithm that takes as input the security parameter* 1^λ *and returns* $(N = pq, e, d)$*, where* p *and* q *are distinct primes of bit length* $\lambda/2$ *and* e, d *are integers such that* $ed = 1 \bmod \phi(N)$*. The RSA assumption with respect to* GenRSA *is a non-interactive computational assumption* $\Lambda_{\mathsf{RSA}} = (\mathsf{InstGen}_{\mathsf{RSA}}, \mathsf{V}_{\mathsf{RSA}}, \mathsf{U}_{\mathsf{RSA}})$ *where*

1. $\mathsf{InstGen}_{\mathsf{RSA}}(1^\lambda)$ *runs* $(N, e, d) \xleftarrow{\$} \mathsf{GenRSA}(1^\lambda)$*, selects* $x \xleftarrow{\$} \mathbb{Z}_N$*, computes* $y = x^e \bmod N$ *and outputs a problem instance* $\phi = (N, e, y)$ *and a witness* $\omega = x$*.*
2. $\mathsf{V}_{\mathsf{RSA}}(\phi, \omega, \rho)$ *returns* 1 *if and only if* $\rho = \omega$*.*
3. $\mathsf{U}_{\mathsf{RSA}}(\phi)$ *returns a failure symbol* $\perp$*.*

Recall the RSA-FDH signature scheme [10] $\mathsf{Sig} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ as follows.

- Gen runs $(N, e, d) \xleftarrow{\$} \mathsf{GenRSA}(1^\lambda)$ and returns $pk = (N, e), sk = (N, d)$.

– $\mathsf{Sign}(sk, m)$ returns $\sigma = H(m)^d \bmod N$ where $H : \{0,1\}^* \to \mathbb{Z}_n$ is a hash function.
– $\mathsf{Vrfy}(pk, m, \sigma)$ returns 1 if and only if $\sigma^e = H(m) \bmod N$.

The scheme provides existential unforgeability under chosen message attacks, which can be reduced to the RSA assumption in the random oracle model as shown by [11,18]. However, these proofs are neither work-factor-tight (an inherent loss linear in the number of signature queries) nor memory-tight (implementing the random oracle). Auerbach et al. [4] were able to improve those results by proving RSA-FDH memory-tight in the single-challenge setting, based on the RSA assumption in the random oracle model. We show how to further improve this result with our techniques.

We proceed as in Section 5.1. That is, we first argue that RSA-FDH is strongly unforgeable under an chosen message attack in the single-challenge and one-signature-per-message setting (sEUF-CMA1-secure) under the RSA assumption in the random oracle model. From this result, we then construct the canonical reduction to show multi-challenge security. The transform presented in Section 4 then finally gives us many-signatures-per-message security again.

We will omit a full proof of sEUF-CMA1 security of RSA-FDH but only provide a brief sketch. The proof is very similar to the proof of EUF-CMA security presented by Auerbach et al. [4]. Note that RSA-FDH scheme is a unique signature scheme. That is, for every message m there is exactly one valid signature, namely $\sigma = \mathsf{H}(m)^d \mod N$. Thus, whenever $\mathsf{Sign}(m)$ is queried it will always return the same signature σ and the adversary will always see exactly one signature per message. Moreover, given a valid message-signature pair (m^*, σ^*), there exists no second valid signature $\sigma \neq \sigma^*$. Hence,

$$\mathsf{Adv}^{\mathsf{sEUF\text{-}CMA1}}_{\text{RSA-FDH}}(\mathcal{A}) \leq \mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\text{RSA-FDH}}(\mathcal{A}). \tag{8}$$

As we need a memory-tight reduction for RSA-FDH up to a truly random function RF, we adapt the result [4, Thm. 5] by Auerbach et al. slightly. Namely, we do not implement the random sampling with a PRF as they are doing, but by a truly random function RF that is maintained with an explicit look-up table. By standard arguments, it is easy to verify that with this adaptation it follows from [4, Thm. 5] and Equation (8) that

$$\mathsf{Adv}^{\mathsf{sEUF\text{-}CMA1}}_{\text{RSA-FDH}}(\mathcal{A}) \leq \exp(1) \cdot q_{\mathsf{S}} \cdot \mathsf{Adv}^{\mathsf{NICA}}_{\Lambda_{\mathsf{RSA}},\lambda}(\mathcal{B}) \tag{9}$$

where q_{S} denotes the number of signature queried by $\mathcal{A}$ and where $\mathcal{B}$ is identical to $\mathcal{B}_2$ in the proof of [4, Thm. 5] except that $\mathcal{B}$ uses a random function RF with a explicitly stored look-up table instead of a PRF. We have

$$\begin{aligned}
&\mathbf{LocalTime}(\mathcal{B}) \approx \mathbf{LocalTime}(\mathcal{A}) + (q_{\mathsf{H}} + q_{\mathsf{S}}) \cdot \mathbf{Time}(\mathsf{RF}),\\
&\mathbf{LocalMem}(\mathcal{B}) = \mathbf{LocalMem}(\mathcal{A}) + \mathbf{Mem}(\mathsf{RF}) + 3
\end{aligned}$$

where q_{H} is the number of random oracle queries and q_{S} the number of signature queries made by $\mathcal{A}$.

We define the canonical reduction $\mathcal{R}_{\mathsf{RSA}}$ from $\mathsf{sEUF\text{-}CMA1}$-security to the RSA assumption as tuple $(\mathsf{RGen}, \mathsf{RSign}, \mathsf{RExtract}, \mathsf{RHash})$ as follows. In essence, $\mathcal{R}_{\mathsf{RSA}}$ works exactly as $\mathcal{B}$. Let $\mathsf{RF}\colon \{0,1\}^* \to \{0,1\} \times \mathbb{Z}_N$ with $\mathsf{Coins}_{\mathsf{RSign}} = \mathsf{Coins}_{\mathsf{RExtract}} = \emptyset$ and $\{0,1\} \times \mathbb{Z}_N = \mathsf{Coins}_{\mathsf{RHash}}$. Further, for $(b, r) := \mathsf{RF}(x)$, we define the short-hands $b =: \mathsf{RF}_1(x)$ and $r =: \mathsf{RF}_2(x)$. We view RF_1 as an $(1/q_{\mathsf{S}})$-biased random function similar to the biased coin used by Coron [18], i.e., $\Pr[\mathsf{RF}_1(x) = 1] = 1/q_{\mathsf{S}}$, where q_{S} is the number of signature queries issued by the adversary.

RGen**:** Given an RSA instance $\phi = (N, e, y)$, RGen returns $(\mathit{simpk}, \mathit{simsk}) = ((N, e), (N, e, y))$.

$\mathsf{RHash}^{\mathsf{RF}(\cdot)}$**:** Given $\mathit{simsk} = (N, e, y)$ and x, RHash returns $\mathsf{RF}_2(x)^e \cdot y$ if $\mathsf{RF}_1(x) = 1$. Otherwise, it returns $\mathsf{RF}_2(x)^e$.

$\mathsf{RSign}^{\mathsf{RF}(\cdot)}$**:** Given $\mathit{simsk} = (N, e, y)$ and m, RSign outputs a signature $\sigma = \mathsf{RF}_2(m)$ if $\mathsf{RF}_1(m) = 0$. Otherwise, the reduction aborts and terminates by outputting the failure symbol $\perp$.

$\mathsf{RExtract}^{\mathsf{RF}(\cdot)}$**:** Given $\mathit{simsk} = (N, e, y)$ and (m^*, σ^*), $\mathsf{RExtract}$ outputs a solution $\rho = \sigma^*/\mathsf{RF}_2(m)$. Note that by definition $\mathcal{R}_{\mathsf{RSA}}$ runs $\mathsf{RExtract}$ only if $\mathsf{Vrfy}(\mathit{simpk}, m^*, \sigma^*) = 1$ and $\sigma^* \neq \mathsf{RSign}(\mathit{simsk}, m^*)$. The validity of the signature implies that $(\sigma^*)^e = \mathsf{RHash}(\mathit{simsk}, m^*)$ and since we have $\sigma^* \neq \mathsf{RSign}(\mathit{simsk}, m^*)$, we also know that $\mathsf{RF}_1(m^*) = 1$.

Reduction $\mathcal{R}_{\mathsf{RSA}}$ works basically as $\mathcal{B}$, we have due to Equation (9)

$$\mathsf{Adv}^{\mathsf{NICA}}_{\Lambda_{\mathsf{RSA}},\lambda}(\mathcal{R}^{\mathcal{A}}_{\mathsf{RSA}}) \geq \frac{1}{\exp(1) \cdot q_{\mathsf{S}}} \cdot \mathsf{Adv}^{\mathsf{sEUF\text{-}CMA1}}_{\text{RSA-FDH}}(\mathcal{A}).$$

That is, $\mathcal{R}_{\mathsf{RSA}}$ is a $(\ell, 0)$-canonical reduction for RSA-FDH with value $\ell = 1/(\exp(1) \cdot q_{\mathsf{S}})$. The local time of $\mathcal{R}^{\mathcal{A}}_{\mathsf{RSA}}$ is $\mathbf{LocalTime}(\mathcal{R}^{\mathcal{A}}_{\mathsf{RSA}}) \approx \mathbf{LocalTime}(\mathcal{A})$ $+ \mathbf{Time}(\mathsf{Sig.Vrfy}) + (q_{\mathsf{H}} + q_{\mathsf{S}} + 1) \cdot \mathbf{Time}(\mathsf{RF})$, and the local memory is

$$\mathbf{LocalMem}(\mathcal{R}^{\mathcal{A}}_{\mathsf{RSA}}) = \mathbf{LocalMem}(\mathcal{A}) + \mathbf{Mem}(\mathsf{RF}) + \mathbf{Mem}(\mathsf{Sig.Vrfy}) + 3.$$

Now, we can use Theorem 9 to lift the security of RSA-FDH to the multi-challenge in a memory-tight way. To this end, we can construct a reduction $\mathcal{R}'_{\mathsf{RSA}}$ from $\mathsf{msEUF\text{-}CMA1}$-security of RSA-FDH to the RSA assumption as presented in the proof Theorem 9. This implies that we can construct an adversary $\mathcal{B}'$ such that

$$\mathsf{Adv}^{\mathsf{NICA}}_{\Lambda_{\mathsf{RSA}},\lambda}((\mathcal{R}'_{\mathsf{RSA}})^{\mathcal{A}'}) \geq \frac{1}{\exp(1) \cdot q_{\mathsf{S}}} \cdot \mathsf{Adv}^{\mathsf{msEUF\text{-}CMA1}}_{\text{RSA-FDH}}(\mathcal{A}') - \mathsf{Adv}^{\mathsf{PRF\text{-}sec}}_{\mathsf{PRF}}(\mathcal{B}')$$

where $\mathsf{PRF}\colon \{0,1\}^\lambda \times \{0,1\}^* \to \{0,1\} \times \mathbb{Z}_N$ is a keyed function. Moreover, it holds that

$$\begin{aligned}
\mathbf{LocalTime}((\mathcal{R}'_{\mathsf{RSA}})^{\mathcal{A}'}) &\approx \mathbf{LocalTime}(\mathcal{A}') + \mathbf{Time}(\mathsf{RGen}) \\
&\quad + (q_{\mathsf{S}} + q_{\mathsf{F}} + q_{\mathsf{H}}) \cdot \mathbf{Time}(\mathsf{PRF}) + q_{\mathsf{F}} \cdot \mathbf{Time}(\mathsf{Sig.Vrfy}) \\
\mathbf{LocalMem}((\mathcal{R}'_{\mathsf{RSA}})^{\mathcal{A}'}) &= \mathbf{LocalMem}(\mathcal{A}') + 4 + \mathbf{Mem}(\mathsf{Sig.Vrfy}) \\
&\quad + \mathbf{Mem}(\mathsf{PRF}).
\end{aligned}$$

Thus, the reduction $\mathcal{R}'_{\mathsf{RSA}}$ is a memory-tight, but not work-factor-tight, reduction from msEUF-CMA1-security to the RSA assumption.

Note that since RSA-FDH is a unique signature scheme, the one-signature-per-message security automatically implies the many-signatures-per-message security. Thus, we do not need to apply our theorem form Section 4. At first glance, this result seems to contradict the memory lower bound for unique signatures established by Wang *et al.* [51, Theorem 3]. However, this is not the case as our reduction does not meet the criteria for their impossibility result to hold.[8] So we evade their lower bound and achieve memory tightness for RSA-FDH.

On the Overall Tightness of RSA-FDH. In the previous section, we have shown how RSA-FDH can be proven memory-tight in the multi-challenge and many-signatures-per-message setting. As already explained above, due to existing tightness lower bounds, plain RSA-FDH cannot be proven work-fact-tight. However, when considering a slight variant of RSA-FDH, which was proposed by Katz and Wang [44], we can apply our techniques to prove this variant fully tight. In essence, we still consider RSA-FDH, but choose a uniformly random bit b and sign $b \,\|\, m$ instead of only m. We call this scheme RSA-FDH+ and we can prove the following theorem.

Theorem 19. *For any adversary $\mathcal{A}'$, there exists a reduction $\mathcal{R}'_{\mathsf{RSA+}}$ and an adversary $\mathcal{B}'$ such that*

$$\mathsf{Adv}^{\mathsf{msEUF\text{-}CMA1}}_{\text{RSA-FDH+}}(\mathcal{A}') \leq 2\mathsf{Adv}^{\mathsf{NICA}}_{\Lambda_{\mathsf{RSA}},\lambda}((\mathcal{R}'_{\mathsf{RSA+}})^{\mathcal{A}'}) + 2\mathsf{Adv}^{\mathsf{PRF\text{-}sec}}_{\mathsf{PRF}}(\mathcal{B}').$$

where $\mathsf{PRF}\colon \{0,1\}^\lambda \times \{0,1\}^* \to \{0,1\} \times \mathbb{Z}_N \times \mathbb{Z}_N$ *is a keyed PRF. Moreover, it holds that*

$$\begin{aligned}\mathbf{LocalTime}((\mathcal{R}'_{\mathsf{RSA+}})^{\mathcal{A}'}) &\approx \mathbf{LocalTime}(\mathcal{A}') + \mathbf{Time}(\mathsf{RGen}) \\ &\quad + (q_{\mathsf{S}} + q_{\mathsf{F}} + q_{\mathsf{H}}) \cdot \mathbf{Time}(\mathsf{PRF}) + q_{\mathsf{F}} \cdot \mathbf{Time}(\mathsf{Sig.Vrfy}) \\ \mathbf{LocalMem}((\mathcal{R}'_{\mathsf{RSA+}})^{\mathcal{A}'}) &= \mathbf{LocalMem}(\mathcal{A}') + \mathbf{Mem}(\mathsf{Sig.Vrfy}) + \mathbf{Mem}(\mathsf{PRF}) + 4.\end{aligned}$$

Hence, $\mathcal{R}'_{\mathsf{RSA+}}$ is a fully tight reduction (i.e., work-factor-tight and memory-tight), from msEUF-CMA1-security of RSA-FDH+ to the RSA assumption. Applying the transform of Section 4 and adding an additional nonce that is signed along with the message, we can further lift this result to achieve msEUF-CMA-security under the RSA assumption.

The proof of Theorem 19 follows the Katz-Wang approach. We provide the formal description of scheme RSA-FDH+ and the proof of Theorem 19 in the full version [21].

[8] More precisely, Wang *et al.* [51] define two parameters c_r and c_g, where c_r captures the work-factor loss of the reduction and c_g captures the trivial winning probability of the assumption. They require $c_g < 1/2$ and $c_r + c_g > 1/2$ for their lower bound to hold. However, we have $c_g = 0$ for the RSA assumption and $c_r = 1/(\exp(1) \cdot q_{\mathsf{S}})$ for our reduction, implying $c_r + c_g < 1/2$, which does not fall into the realm of Theorem 3 in [51].

References

1. Abdalla, M., Fouque, P.A., Lyubashevsky, V., Tibouchi, M.: Tightly-secure signatures from lossy identification schemes. In: EUROCRYPT 2012. LNCS, vol. 7237, pp. 572–590. Springer, Heidelberg (Apr 2012)
2. Abdalla, M., Fouque, P.A., Lyubashevsky, V., Tibouchi, M.: Tightly secure signatures from lossy identification schemes. Journal of Cryptology **29**(3), 597–631 (Jul 2016)
3. Abe, M., Groth, J., Ohkubo, M.: Separating short structure-preserving signatures from non-interactive assumptions. In: ASIACRYPT 2011. LNCS, vol. 7073, pp. 628–646. Springer, Heidelberg (Dec 2011)
4. Auerbach, B., Cash, D., Fersch, M., Kiltz, E.: Memory-tight reductions. In: CRYPTO 2017, Part I. LNCS, vol. 10401, pp. 101–132. Springer, Heidelberg (Aug 2017)
5. Bader, C., Hofheinz, D., Jager, T., Kiltz, E., Li, Y.: Tightly-secure authenticated key exchange. In: TCC 2015, Part I. LNCS, vol. 9014, pp. 629–658. Springer, Heidelberg (Mar 2015)
6. Bader, C., Jager, T., Li, Y., Schäge, S.: On the impossibility of tight cryptographic reductions. In: EUROCRYPT 2016, Part II. LNCS, vol. 9666, pp. 273–304. Springer, Heidelberg (May 2016)
7. Bellare, M., Boldyreva, A., Micali, S.: Public-key encryption in a multi-user setting: Security proofs and improvements. In: EUROCRYPT 2000. LNCS, vol. 1807, pp. 259–274. Springer, Heidelberg (May 2000)
8. Bellare, M., Ristenpart, T.: Simulation without the artificial abort: Simplified proof and improved concrete security for Waters' IBE scheme. In: EUROCRYPT 2009. LNCS, vol. 5479, pp. 407–424. Springer, Heidelberg (Apr 2009)
9. Bellare, M., Ristenpart, T.: Simulation without the artificial abort: Simplified proof and improved concrete security for waters' IBE scheme. Cryptology ePrint Archive, Report 2009/084 (2009), `http://eprint.iacr.org/2009/084`
10. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: ACM CCS 93. pp. 62–73. ACM Press (Nov 1993)
11. Bellare, M., Rogaway, P.: The exact security of digital signatures: How to sign with RSA and Rabin. In: EUROCRYPT'96. LNCS, vol. 1070, pp. 399–416. Springer, Heidelberg (May 1996)
12. Bhattacharyya, R.: Memory-tight reductions for practical key encapsulation mechanisms. In: PKC 2020, Part I. LNCS, vol. 12110, pp. 249–278. Springer, Heidelberg (May 2020)
13. Blazy, O., Kiltz, E., Pan, J.: (Hierarchical) identity-based encryption from affine message authentication. In: CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 408–425. Springer, Heidelberg (Aug 2014)
14. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the Weil pairing. In: ASIACRYPT 2001. LNCS, vol. 2248, pp. 514–532. Springer, Heidelberg (Dec 2001)
15. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the Weil pairing. Journal of Cryptology **17**(4), 297–319 (Sep 2004)
16. Chen, J., Wee, H.: Fully, (almost) tightly secure IBE and dual system groups. In: CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 435–460. Springer, Heidelberg (Aug 2013)
17. Cohn-Gordon, K., Cremers, C., Gjøsteen, K., Jacobsen, H., Jager, T.: Highly efficient key exchange protocols with optimal tightness. In: CRYPTO 2019, Part III. LNCS, vol. 11694, pp. 767–797. Springer, Heidelberg (Aug 2019)

18. Coron, J.S.: On the exact security of full domain hash. In: CRYPTO 2000. LNCS, vol. 1880, pp. 229–235. Springer, Heidelberg (Aug 2000)
19. Coron, J.S.: Optimal security proofs for PSS and other signature schemes. In: EUROCRYPT 2002. LNCS, vol. 2332, pp. 272–287. Springer, Heidelberg (Apr / May 2002)
20. Davis, H., Günther, F.: Tighter proofs for the sigma and TLS 1.3 key exchange protocols. Cryptology ePrint Archive, Report 2020/1029 (2020), `https://eprint.iacr.org/2020/1029`
21. Diemert, D., Gellert, K., Jager, T., Lyu, L.: Digital signatures with memory-tight security in the multi-challenge setting. Cryptology ePrint Archive, Report 2021/1220 (2021), `https://ia.cr/2021/1220`
22. Diemert, D., Gellert, K., Jager, T., Lyu, L.: More efficient digital signatures with tight multi-user security. In: Public-Key Cryptography – PKC 2021. pp. 1–31. Springer International Publishing, Cham (2021)
23. Diemert, D., Jager, T.: On the tight security of TLS 1.3: Theoretically-sound cryptographic parameters for real-world deployments. Cryptology ePrint Archive, Report 2020/726; to appear in the Journal of Cryptology (2020), `https://eprint.iacr.org/2020/726`
24. Fersch, M., Kiltz, E., Poettering, B.: On the one-per-message unforgeability of (EC)DSA and its variants. In: TCC 2017, Part II. LNCS, vol. 10678, pp. 519–534. Springer, Heidelberg (Nov 2017)
25. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: CRYPTO'86. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (Aug 1987)
26. Fleischhacker, N., Jager, T., Schröder, D.: On tight security proofs for Schnorr signatures. In: ASIACRYPT 2014, Part I. LNCS, vol. 8873, pp. 512–531. Springer, Heidelberg (Dec 2014)
27. Fleischhacker, N., Jager, T., Schröder, D.: On tight security proofs for Schnorr signatures. Journal of Cryptology **32**(2), 566–599 (Apr 2019)
28. Garg, S., Bhaskar, R., Lokam, S.V.: Improved bounds on security reductions for discrete log based signatures. In: CRYPTO 2008. LNCS, vol. 5157, pp. 93–107. Springer, Heidelberg (Aug 2008)
29. Gay, R., Hofheinz, D., Kiltz, E., Wee, H.: Tightly CCA-secure encryption without pairings. In: EUROCRYPT 2016, Part I. LNCS, vol. 9665, pp. 1–27. Springer, Heidelberg (May 2016)
30. Ghoshal, A., Jaeger, J., Tessaro, S.: The memory-tightness of authenticated encryption. In: Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part I. Lecture Notes in Computer Science, vol. 12170, pp. 127–156. Springer (2020)
31. Ghoshal, A., Tessaro, S.: On the memory-tightness of hashed ElGamal. In: EUROCRYPT 2020, Part II. LNCS, vol. 12106, pp. 33–62. Springer, Heidelberg (May 2020)
32. Gjøsteen, K., Jager, T.: Practical and tightly-secure digital signatures and authenticated key exchange. In: CRYPTO 2018, Part II. LNCS, vol. 10992, pp. 95–125. Springer, Heidelberg (Aug 2018)
33. Goldwasser, S., Micali, S., Rivest, R.L.: A digital signature scheme secure against adaptive chosen-message attacks. SIAM Journal on Computing **17**(2), 281–308 (Apr 1988)

34. Gueron, S., Lindell, Y.: Better bounds for block cipher modes of operation via nonce-based key derivation. In: ACM CCS 2017. pp. 1019–1036. ACM Press (Oct / Nov 2017)
35. Hasegawa, S., Isobe, S.: Lossy identification schemes from decisional RSA. In: International Symposium on Information Theory and its Applications, ISITA 2014, Melbourne, Australia, October 26-29, 2014. pp. 143–147. IEEE (2014)
36. Hoang, V.T., Tessaro, S.: The multi-user security of double encryption. In: EUROCRYPT 2017, Part II. LNCS, vol. 10211, pp. 381–411. Springer, Heidelberg (Apr / May 2017)
37. Hofheinz, D., Jager, T.: Tightly secure signatures and public-key encryption. In: CRYPTO 2012. LNCS, vol. 7417, pp. 590–607. Springer, Heidelberg (Aug 2012)
38. Hofheinz, D., Jager, T., Knapp, E.: Waters signatures with optimal security reduction. In: PKC 2012. LNCS, vol. 7293, pp. 66–83. Springer, Heidelberg (May 2012)
39. Jager, T., Kakvi, S.A., May, A.: On the security of the PKCS#1 v1.5 signature scheme. In: ACM CCS 2018. pp. 1195–1208. ACM Press (Oct 2018)
40. Jager, T., Kurek, R., Pan, J.: Simple and more efficient PRFs with tight security from LWE and matrix-DDH. In: ASIACRYPT 2018, Part III. LNCS, vol. 11274, pp. 490–518. Springer, Heidelberg (Dec 2018)
41. Jager, T., Stam, M., Stanley-Oakes, R., Warinschi, B.: Multi-key authenticated encryption with corruptions: Reductions are lossy. In: TCC 2017, Part I. LNCS, vol. 10677, pp. 409–441. Springer, Heidelberg (Nov 2017)
42. Kakvi, S.A., Kiltz, E.: Optimal security proofs for full domain hash, revisited. In: EUROCRYPT 2012. LNCS, vol. 7237, pp. 537–553. Springer, Heidelberg (Apr 2012)
43. Kakvi, S.A., Kiltz, E.: Optimal security proofs for full domain hash, revisited. Journal of Cryptology **31**(1), 276–306 (Jan 2018)
44. Katz, J., Wang, N.: Efficiency improvements for signature schemes with tight security reductions. In: ACM CCS 2003. pp. 155–164. ACM Press (Oct 2003)
45. Lewko, A.B., Waters, B.: Efficient pseudorandom functions from the decisional linear assumption and weaker variants. In: ACM CCS 2009. pp. 112–120. ACM Press (Nov 2009)
46. Liu, X., Liu, S., Gu, D., Weng, J.: Two-pass authenticated key exchange with explicit authentication and tight security. In: ASIACRYPT 2020, Part II. LNCS, vol. 12492, pp. 785–814. Springer, Heidelberg (Dec 2020)
47. Naor, M., Reingold, O.: Number-theoretic constructions of efficient pseudo-random functions. In: 38th FOCS. pp. 458–467. IEEE Computer Society Press (Oct 1997)
48. Paillier, P., Vergnaud, D.: Discrete-log-based signatures may not be equivalent to discrete log. In: ASIACRYPT 2005. LNCS, vol. 3788, pp. 1–20. Springer, Heidelberg (Dec 2005)
49. Schäge, S.: Tight proofs for signature schemes without random oracles. In: EUROCRYPT 2011. LNCS, vol. 6632, pp. 189–206. Springer, Heidelberg (May 2011)
50. Seurin, Y.: On the exact security of Schnorr-type signatures in the random oracle model. In: EUROCRYPT 2012. LNCS, vol. 7237, pp. 554–571. Springer, Heidelberg (Apr 2012)
51. Wang, Y., Matsuda, T., Hanaoka, G., Tanaka, K.: Memory lower bounds of reductions revisited. In: EUROCRYPT 2018, Part I. LNCS, vol. 10820, pp. 61–90. Springer, Heidelberg (Apr / May 2018)