On Tightly Secure Primitives in the Multi-Instance Setting

Dennis Hofheinz[1], Ngoc Khanh Nguyen[2,†]

[1] Karlsruhe Institute of Technology, Germany
[2] IBM Research Zurich, Switzerland

Abstract. We initiate the study of *general* tight reductions in cryptography. There already exist a variety of works that offer tight reductions for a number of cryptographic tasks, ranging from encryption and signature schemes to proof systems. However, our work is the first to provide a universal definition of a tight reduction (for arbitrary primitives), along with several observations and results concerning primitives for which tight reductions have not been known.
Technically, we start from the general notion of reductions due to Reingold, Trevisan, and Vadhan (TCC 2004), and equip it with a quantification of the respective reduction loss, and a canonical multi-instance extension to primitives. We then revisit several standard reductions whose tight security has not yet been considered. For instance, we revisit a generic construction of signature schemes from one-way functions, and show how to tighten the corresponding reduction by assuming collision-resistance from the used one-way function. We also obtain tightly secure pseudorandom generators (by using suitable rerandomisable hard-core predicates), and tightly secure lossy trapdoor functions.

1 Introduction

Motivation. To argue for the security of a cryptographic scheme, we usually employ a security reduction (or simply reduction). A reduction formalises that the only way to break the scheme is to solve an underlying computational problem (such as factoring a large integer). More specifically, a reduction turns any adversary A on the scheme into a problem solver B. Hence, if the problem is hard to solve, then the scheme must be secure.

Most existing reductions are however *loose*, in the sense that B's success is much lower (or its runtime much higher) than that of A. For instance, for most existing encryption schemes, the best known reduction in a multi-user, multi-ciphertext scenario yields Bs whose success degrades linearly in the number of users and/or ciphertexts. Hence, in a large-scale setting, this reduction loss can easily be in the order of, say, 2^{30}.

In contrast, a *tight* reduction yields problem solvers B which have the same success (and running time) as A.[1] A loose reduction gives quantitatively lower

[†]This work was carried out when the author was doing an internship at the KIT.

[1]Of course, there are other interesting properties (such as memory usage [4]) of a given adversary A one would want a reduction to preserve.

guarantees than a tight one. For instance, suppose one would like to give a key length recommendation for a scheme based on the currently best attacks on the underlying computational problem. In this case, loose reductions lead to larger key length recommendations, and thus to a (perhaps substantially) less efficient scheme.

In this work, we are interested in tight reductions, in particular in a setting in which the scheme or primitive is used multiple times.

State of the art. The tightness of reductions (in particular for schemes in a multi-instance scenario) has first been considered by Bellare, Boldyreva, and Micali [6]. Since their work, a variety of tightly secure constructions for concrete cryptographic building blocks (such as encryption [27,1,34,35,17,26,18], identity-based encryption [11,7,28,3,22], digital signatures [34,35,25,2], or zero-knowledge proofs [27,17]) have been proposed.

On the other hand, the notion of a reduction has been formalised early on in cryptography (e.g., in the context of black-box separations such as [31,45,43]).[2] We note that these works were mostly interested in the (non-)existence of reductions for certain types of schemes, and do not take into account reduction loss. Hence, currently there is no *general* (i.e., formal but primitive-independent) definition of a tight reduction.

Our contribution. We provide the first general definition of a tight reduction, and revisit several classical reductions (with an emphasis on their tightness). We obtain the following results:

- We obtain a new (and tighter) security reduction for the classical construction of signatures from one-way functions [44,32,23].
- We also obtain tightly secure pseudorandom generators by instantiating the classical construction of Blum and Micali [8] with a suitable (i.e., rerandomisable) hard-core predicate.
- Finally, we show that the DDH-based lossy trapdoor functions of Peikert and Waters [42] are tightly secure in a multi-instance scenario.

In the following, we will outline our definition and results.

Our new definition. Our definition of tight security adapts the general definition of reductions due to Reingold, Trevisan, and Vadhan [43]. First, we will consider the tightness of a reduction as an additional property of that reduction. Additionally, we will formalise the "multi-instance version" of a given primitive (taking into account a suitably modified multi-instance security game and potential global parameters).

Perhaps most interestingly, this modification allows to define the notion of a "tightly extensible primitive". Intuitively, a primitive X is tightly extensible relative to another primitive Y if the multi-instance version of X can be tightly reduced to Y. For instance, it is easy to see that one-way functions are tightly

[2] We also remark that other formalisations of cryptographic *assumptions* exist, e.g., [38,14].

extensible relative to collision-resistant hash functions (CRHFs). In fact, a simple extension of an argument of Damgård [12] shows that any compressing CRHF h already is a one-way function in the multi-instance setting: suppose an algorithm A successfully finds a preimage x_i' for one of potentially many given images $h(x_i)$. Since h is compressing, we have $x_i' \neq x_i$ with probability at least $1/2$, so that (x_i, x_i') forms a collision. This holds even if we require "adaptive" one-wayness[3], in the sense that A may get selected preimages x_i upon request (and then loses the inversion game in those instances of course).

Our results. We now outline the results mentioned above.

First, we revisit the classical construction of signatures from one-way functions from [44,32,23]. This construction uses the one-time signature scheme of Lamport [33], which in turn uses many (i.e., $L = O(\lambda)$, where λ is the security parameter) instances of a given one-way function f. Each forged (one-time) signature implies an inversion of one instance of f. The problem here is that it is not clear a priori *which* instance is inverted. Hence, the corresponding security reduction for the one-time signature scheme (as formalized, e.g., in [20]) loses a factor of L, which of course is inherited by the security reduction of the overall signature scheme.

This loss of L can essentially be avoided if we assume that f is collision-resistant. Concretely, recall our observation above that f (when viewed as an adaptive one-way function) is tightly extensible relative to itself (when viewed as a CRHF). In particular, an adversary that inverts one out of many f-instances can be turned into a collision finder for f with a reduction loss of only 2. Hence, any forged one-time signature can be converted into an f-collision with probability of at least $1/2$, and we can save a factor of $L/2$ in the overall reduction.

Next, we consider pseudorandom generators (PRGs) G that are tightly extensible (relative to themselves). In other words, we are looking for a G such that the pseudorandomness of many $G(x_i)$ instances (for independently chosen seeds x_i) can be tightly reduced to the pseudorandomness of a single $G(x)$. This property leads to tighter reductions whenever G is used multiple times (e.g., in one or many instances of a larger scheme).

Note that an almost trivial solution to this problem can be found under the DDH assumption (assuming groups with dense representations, such that random group elements are random bitstrings). Namely, recall that the DDH assumption states that for a generator g and random exponents a, b, c, the tuple (g^a, g^b, g^{ab}) is computationally indistinguishable from (g^a, g^b, g^c). Now the DDH assumption is known to be rerandomisable (e.g., [6, Lemma 1]), in the sense a distinguisher between many $(g^{a_i}, g^{b_i}, g^{a_i b_i})$ and many $(g^{a_i}, g^{b_i}, g^{c_i})$ can be converted into a DDH distinguisher, with (almost) no reduction loss. Hence, defining $G(a, b) = g^a || g^b || g^{ab}$ yields a tightly extensible PRG.

Here, however, we are interested in constructions from (potentially) weaker assumptions. To this end, we revisit the PRG of Blum and Micali [8]. This PRG

[3] This notion is not related to the notion of adaptive one-way functions from [40] in which an adversary gets access to a full inversion oracle for the function.

assumes a one-way permutation f with a hard-core predicate b, and defines $G(x) = f(x)||b(x)$.[4] We set $f(x) = g^x$ (which also means we require a group with dense representations), and $b(x)$ to be the Legendre symbol $(\frac{x}{p})$ of x modulo the group order p.[5] Under a suitable computational assumption (that appears to lie in between the CDH and DDH assumptions), b is indeed a hard-core predicate of f. Most importantly, and unlike with other hard-core predicates, f and b are rerandomisable: given $f(x)$ and $b(x)$, it is easy to compute $f(ax)$ and $b(ax)$ (for a known random a). Hence, by rerandomising PRG images, we can show the tight extensibility of this G.[6]

Finally, we consider the tight extensibility of lossy trapdoor functions (LTDFs) relative to themselves. Our motivation to consider LTDFs is that they form an abstract tool which is already known to imply tightly (IND-CPA-)secure encryption in the single-user (but multi-ciphertext) setting [42]. A tightly extensible LTDF can be additionally useful in settings with many instances (e.g., users). Here, our main result is that the DDH-based LTDF construction of Peikert and Waters [42] is already tightly extensible. The corresponding argument is somewhat technical, but relies on the rerandomisability of the DDH assumption (as outlined already above).

We note that this last result does not yield interesting new tightly secure encryption schemes. In fact, already the ElGamal scheme is tightly IND-CPA-secure under the DDH assumption [6]. Rather, we view our last result as conceptual: it shows that an abstract building block (that was already known to enable "partially tight" reductions) is tightly secure even in a multi-instance setting.

There are areas of cryptography where we have not looked at applying tight extensibility. For example, a natural question would be if we can build tightly extensible symmetric encryption schemes or even more complicated protocols (e.g. zero-knowledge). More importantly, it is an open question whether this notion can be used in constructing more efficient and more secure primitives which could not be shown using current state-of-the-art methods.

2 Preliminaries

In this section we review standard notation and cryptographic definitions we use in later sections. We also provide relevant background related to formal notion of black-box reductions.

2.1 Notation

Let $\mathbb{N}$ be the set of natural numbers and $\mathbb{Z}_n$ be the set of integers modulo n. We denote the security parameter by $\lambda \in \mathbb{N}$ and assume that it is implicitly given to

[4] For simplicity, we only consider a PRG that stretches its seed x by one bit.

[5] Slightly simplifying, we ignore the unlikely case $x = 0$ and treat $(\frac{x}{p})$ as a bit.

[6] We note that the Legendre symbol has already been considered as a hard-core predicate by Damgård [13], although, to the best of our knowledge, its rerandomisability has not been investigated before.

all algorithms in the unary representation 1^λ, unless stated otherwise. An algorithm here is defined as a stateless Turing machine. Algorithms are randomised and PPT means "probabilistic polynomial time" in the (unary) security parameter λ. For a randomised algorithm $\mathcal{A}$, we denote $\mathbf{T}(\mathcal{A})$ for the worst-case runtime of $\mathcal{A}$, parametrized over λ. Also, we describe $(y_1, \dots) \leftarrow_\$ \mathcal{A}(1^\lambda, x_1, \dots; r)$ as an event when $\mathcal{A}$ gets $(1^\lambda, x_1, \dots)$ as input, uses fresh random coins r and outputs $(y_1, \dots)$. If $\mathcal{A}$ is determininstic then we simply write $(y_1, \dots) \leftarrow \mathcal{A}(1^\lambda, x_1, \dots)$. Let us write $\mathcal{A}^B$ to denote that $\mathcal{A}$ has black-box access to algorithm B, meaning it sees only its input-output behaviour. We define $\mathsf{queries}(\mathcal{A}^B, B)$ to be the worst-case number of messages/queries sent by B to $\mathcal{A}$ (parametrized by λ). On the other hand, $\mathcal{A}^{(\cdot)}$ means that $\mathcal{A}$ expects a black-box access to some other algorithm.

For a finite set S, we denote its cardinality by $|S|$ and write $s \leftarrow_\$ S$ meaning that we choose an element s from S uniformly at random. For a function $f : A \to B$ and $C \subset A$, we define $f|_C : C \to B$ as $f|_C(x) = f(x)$. We write $\mathsf{poly}(\lambda)$ to denote the set of polynomials in λ. A function $v : \mathbb{N} \to \mathbb{R}^{\geq 0}$ is negligible if for any $c \in \mathbb{N}$, $\lim_{\lambda \to \infty} v(\lambda)\lambda^c = 0$. We let $\mathsf{negl}(\lambda)$ denote an unspecified negligible function in λ. Throughout the paper, $\perp$ denotes an error symbol.

Denote $\mathcal{X} = \{X_\lambda\}_{\lambda \in \mathbb{N}}, \mathcal{Y} = \{Y_\lambda\}_{\lambda \in \mathbb{N}}$ as ensembles of random variables over some countable set S indexed by λ. Then, $\mathcal{X}$ and $\mathcal{Y}$ are statistically indistinguishable ($\mathcal{X} \overset{s}{\approx} \mathcal{Y}$) if $\Delta(X_\lambda, Y_\lambda) = \frac{1}{2} \sum_{s \in S} |\Pr[X = s] - \Pr[Y = s]| = \mathsf{negl}(\lambda)$. Moreover, $\mathcal{X}$ and $\mathcal{Y}$ are computationally indistinguishable ($\mathcal{X} \overset{c}{\approx} \mathcal{Y}$) if for every PPT algorithm $\mathcal{A}$:

$$|\Pr[1 \leftarrow_\$ \mathcal{A}(1^\lambda, X_\lambda)] - \Pr[1 \leftarrow_\$ \mathcal{A}(1^\lambda, Y_\lambda)]| = \mathsf{negl}(\lambda).$$

2.2 Cryptographic Primitives

One-Way Functions. Intuitively, we say that a function is one-way (OWF) if it is easy to compute but hard to invert. Using our notation, we formalise it as follows:

Definition 1. *A function $f : \{0,1\}^* \to \{0,1\}^*$ is one-way if:*

- *There exists a deterministic polynomial-time algorithm f so that*

$$\forall \lambda \in \mathbb{N}, \Pr[f(x) \leftarrow \mathsf{f}(1^\lambda, x) : x \leftarrow_\$ \{0,1\}^\lambda] = 1.$$

- *For every PPT algorithm $\mathcal{A}$,*

$$\Pr[f(y) = f(x) : x \leftarrow_\$ \{0,1\}^\lambda, y \leftarrow_\$ \mathcal{A}(1^\lambda, f(x))] = \mathit{negl}(\lambda).$$

Most of the time, we do not deal with a single one-way function but rather with a collection of one-way functions. That is, we consider a set $\mathcal{F} = \{f\}$ of functions f that each have a finite domain, but may be parameterized over the security parameter (and other public parameters such as a fixed group). In that case, we choose f at random for the currently given security parameter (and potentially other parameters), and sample x uniformly at random from f's domain.

In the case of a family $\mathcal{F}$ of parameterized one-way functions f (with finite domain $\mathcal{D}_f$), we say that $\mathcal{F}$ is a family of one-way permutations if f is bijective and $f(\mathcal{D}_f) = \mathcal{D}_f$.

We also recall the notion of a hard-core predicate introduced by Goldreich et al. [21].

Definition 2. *Let $b : \{0,1\}^* \to \{0,1\}$ be a function and $u \leftarrow_{\$} \{0,1\}$. Then, b is a hardcore-bit predicate for function $f : \{0,1\}^* \to \{0,1\}^*$ if*

$$(f(x), b(x)) \overset{c}{\approx} (f(x), u)$$

Goldreich et al. provide in [21] a construction of a one-way function with hard-core predicate from any given one-way function.

Lossy Trapdoor Functions. We say that a function f is a trapdoor function if it is easy to compute $f(x)$ and also easy to invert if we know some "special information" (called trapdoor) but hard to invert without trapdoor. The notion of a lossy trapdoor function was introduced by Peikert et al. [41]. A tuple of PPT algorithms $(S_{inj}, S_{loss}, F, F^{-1})$ is called a collection of $(n,k)-$lossy trapdoor functions (LTDF) if:

- S_{inj} outputs (s,t) where s is a function index and t its trapdoor, $F(s,\cdot)$ computes a deterministic injective function f over the domain $\{0,1\}^n$, and $F^{-1}(t,\cdot)$ computes f^{-1},
- S_{loss} outputs $(s,\perp)$ and $F(s,\cdot)$ computes a deterministic function f over the domain $\{0,1\}^n$ whose image has size at most 2^{n-k},
- For $(s_1,t_1) \leftarrow_{\$} S_{inj}$ and $(s_2,\perp) \leftarrow_{\$} S_{loss}$,

$$s_1 \overset{c}{\approx} s_2.$$

Peikert et al. also define all-but-one trapdoor functions in order to construct an IND-CCA encryption scheme. In this paper we concentrate more on LTDFs but our results can be easily generalised to the second notion.

Pseudorandom Generators. A function $G : \{0,1\}^k \to \{0,1\}^l$, where $k < l$, is a pseudorandom generator (PRG) if given random x from $\{0,1\}^k$, no PPT adversary can distinguish $G(x)$ and a random element from $\{0,1\}^l$.

Definition 3. *Let $G : \{0,1\}^k \to \{0,1\}^l$ be a function where $k < l$, and let $x \leftarrow_{\$} \{0,1\}^k$ and $u \leftarrow_{\$} \{0,1\}^l$ be uniformly chosen. Then, G is a pseudorandom generator if*

$$G(x) \overset{c}{\approx} u.$$

Håstad et al.[24] show that one can construct a pseudorandom generator from any one-way function. Unfortunately, their security reduction has a large loss, i.e., is far from tight. Since then, more efficient constructions of PRGs from OWFs with much tighter reductions have been discovered [29] but they still suffer from a large reduction loss. On the other hand, Blum and Micali [8] provide a construction of a PRG from a one-way permutation which loses a factor of l.

HASHING. A family of functions $\mathcal{H} = \{h_i : A \to B\}$ is universal if for every distinct $x, x' \in A, \Pr_{h \leftarrow_\$ \mathcal{H}}[h(x) = h(x')] = 1/|B|$. Moreover, we say that $\mathcal{H}$ is pairwise independent if, for every distinct $x, x' \in A$ and every $y, y' \in B$, $\Pr_{h \leftarrow_\$ \mathcal{H}}[h(x) = y \wedge h(x') = y'] = 1/|B|^2$.

A hash function $H : \{0,1\}^k \to \{0,1\}^l$, where $k > l$, is collision resistant if for any PPT adversary $\mathcal{A}$, $\Pr[(x, y) \leftarrow_\$ \mathcal{A}(1^\lambda, h) : h(x) = h(y)] = \mathsf{negl}(\lambda)$. Similarly we can define a collection of collision-resistant hash functions.

2.3 Cryptographic Assumptions

We briefly state the most common computational and decisional problems in public-key cryptography. Let G be a cyclic group (that may depend on the security parameter λ) of order p where p is a λ-bit prime. Also, let $g \in G$ be a generator of G. We denote $\langle G \rangle$ to be the description of G.

- Discrete Logarithm Problem (DLOG) - we say that the discrete logarithm problem is hard in G if for every PPT algorithm $\mathcal{A}$:

$$\Pr[x \leftarrow_\$ \mathcal{A}(\langle G \rangle, p, g, g^x) : x \leftarrow_\$ \mathbb{Z}_p] = \mathsf{negl}(\lambda).$$

- Computational Diffie-Hellman Problem (CDH) - we say that the computational Diffie-Hellman problem is hard in G if for every PPT algorithm $\mathcal{A}$:

$$\Pr[g^{xy} \leftarrow_\$ \mathcal{A}(\langle G \rangle, p, g, g^x, g^y) : x, y \leftarrow_\$ \mathbb{Z}_p] = \mathsf{negl}(\lambda).$$

- Decisional Diffie-Hellman Problem (DDH) - we say that the decisional Diffie-Hellman problem is hard in G if for $z \leftarrow_\$ \mathbb{Z}_p$:

$$(\langle G \rangle, p, g, g^x, g^y, g^{xy}) \stackrel{c}{\approx} (\langle G \rangle, p, g, g^x, g^y, g^z).$$

2.4 Public Key Schemes

PUBLIC-KEY ENCRYPTION. A public-key encryption scheme for a given security parameter λ is a triple of PPT algorithms $(\mathsf{Gen}; \mathsf{Enc}; \mathsf{Dec})$ such that:

- $(pk, sk) \leftarrow_\$ \mathsf{Gen}(1^\lambda)$ is the key generation algorithm which takes a security parameter λ and outputs a pair (pk, sk) where pk and sk are called public and secret keys respectively,
- $c \leftarrow_\$ \mathsf{Enc}(pk, m)$ is the encryption algorithm which takes a public key pk, a message m and returns a ciphertext c,
- $m \leftarrow \mathsf{Dec}(sk, c)$ is the decryption algorithm which takes a secret key sk, ciphertext c and returns a message m or $\perp$ if given ciphertext is invalid.

A public-key encryption must satisfy the correctness condition, meaning that for every message m and every security parameter λ, if $(pk, sk) \leftarrow_\$ \mathsf{Gen}(1^\lambda)$ then $\mathsf{Dec}(sk, \mathsf{Enc}(pk, m)) = m$.

We recall basic notions of security for public-key encryption schemes. We say that an encryption scheme $\mathcal{E} = (\mathsf{Gen}; \mathsf{Enc}; \mathsf{Dec})$ is IND-CPA secure (has indistinguishable ciphertexts under chosen plaintext attack) if there exists no PPT adversary $\mathcal{A}$ which wins the following game with non-negligible probability:

1. Challenger C generates $(pk, sk) \leftarrow_{\$} \mathsf{Gen}(1^\lambda)$ and sends pk to $\mathcal{A}$.
2. Adversary $\mathcal{A}$ sends messages (m_0, m_1) to C. Challenger then selects a bit b uniformly at random and returns $c \leftarrow_{\$} \mathsf{Enc}(pk, m_b)$.
3. At the end, $\mathcal{A}$ sends a bit b' to the challenger. Then, $\mathcal{A}$ wins if $b = b'$ and loses otherwise.

Similarly, we define IND-CCA security (indistinguishability under chosen ciphertext attack). We say that $\mathcal{E}$ is IND-CCA secure if there exists no PPT adversary $\mathcal{A}$ which wins a similar game to the one described above (with non-negligible probability), but this time $\mathcal{A}$ also has access to decryption oracle $\mathcal{O}_{\mathsf{Dec}}$ which on input c' returns $\perp$ if $c = c'$ and $\mathsf{Dec}(sk, c')$ otherwise.

One-Time Signatures. A signature scheme consists of a tuple of PPT algorithms $(\mathsf{Gen}; \mathsf{Sign}; \mathsf{Ver})$ satisfying the following conditions:

- $(vk, sk) \leftarrow_{\$} \mathsf{Gen}(1^\lambda)$ is the key generation algorithm, which takes a security parameter λ and outputs a pair (vk, sk) where vk and sk are called verification and signing keys respectively,
- $c \leftarrow_{\$} \mathsf{Sign}(sk, m)$ is the signing algorithm which takes the signing key sk, a message m and returns a signature σ,
- $b \leftarrow \mathsf{Ver}(vk, m, \sigma)$ is the verification algorithm which takes the verification key vk, message m and signature σ and returns a bit b.

Any signature scheme must satisfy the correctness condition meaning that for every message m and every security parameter λ, if $(vk, sk) \leftarrow_{\$} \mathsf{Gen}(1^\lambda)$ then $\mathsf{Ver}(vk, m, \mathsf{Sign}(sk, m)) = 1$.

We now define a security notion for signature schemes called existential unforgeability under a one-time chosen message attack (EUF-OTCMA). We say that the signature scheme $\mathcal{E} = (\mathsf{Gen}; \mathsf{Sign}; \mathsf{Ver})$ is EUF-OTCMA secure if there is no PPT adversary $\mathcal{A}$ which wins the following game with non-negligible probability:

1. Challenger C generates $(vk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$ and sends vk to $\mathcal{A}$.
2. Adversary $\mathcal{A}$ sends message a m to C. Challenger then returns $\sigma = \mathsf{Sign}(sk, m)$.
3. Finally, $\mathcal{A}$ outputs a pair (m', σ') b'. Then, $\mathcal{A}$ wins if $\mathsf{Ver}(vk, m', \sigma') = 1$ and $m' \neq m$ and loses otherwise.

Strong unforgeable one-time signatures can be constructed from a one-way function as well as collision-resistant hash functions.

2.5 Fully Black-Box Reductions

We review the framework of Reingold et al. [43] on security reductions. For simplicity, we only consider fully black-box reductions against uniform adversaries. There are many formal definitions of reductions (such as [5,16,30,43]) but in this paper we focus on work by Reingold et al. [43], mainly due to its simplicity and the ease to modify their framework to suit our needs. Using their notation, primitive P is a pair $\langle F_P, R_P \rangle$ where F_P is a set of functions $f : \{0,1\}^* \rightarrow \{0,1\}^*$

and R_P is a relation over pairs (f, M) for $f \in F_P$ and machine M. One can think of F_P as implementations of primitive P and R_P as security conditions on F_P. For example, if we think of P as a one-way function, the set of implementations could be a set of one-way functions. On the other hand, R_P would be the standard one-wayness game. Depending on the application, it might be useful to define F_P such that it only corresponds to *efficient* (e.g., realizable through PPT machines) implementations.

There is a fully-BB reduction from a primitive $P = \langle F_P, R_P \rangle$ to $Q = \langle F_Q, R_Q \rangle$ if there exist PPT machines G, S such that:

- for every function $f \in F_Q$, $G^f \in F_P$,
- for every function $f \in F_Q$ and every adversary $\mathcal{A}$, $(G^f, \mathcal{A}) \in R_P \implies (f, S^{\mathcal{A}}) \in R_Q$.

As mentioned in [43], this definition of reduction does not apply to non-uniform or information-theoretic notions of security. They also define different types of reductions such as semi-black-box or relativizing reductions.

3 Notion of Tight Reduction

In this section we formalise the notion of tight reduction by adapting the framework of Reingold et al. (RTV) [43]. Roughly speaking, we represent security conditions as a security game instead of a set of relations. Thus, we could formally define what we mean by "breaking one primitive with about the same success as the other primitive" in terms of probabilities. Then, we define what a multi-instance version of a primitive is. At the end, we give a few examples of cryptographic primitives which satisfy our framework.

3.1 Primitives and Reductions

We start by stating what a primitive is and what it means for it to be secure.

Definition 4. *A primitive P is a tuple $\langle \mathbb{P}, S_P, F_P, R_P, \sigma \rangle$ where:*

- *$\mathbb{P}$ is a triple of sets (A, B, C) where $C \subset A$,*
- *F_P is a subset of $\{f : A \to B\}$,*
- *S_P is a PPT setup algorithm which sends parameters $(r_1, \ldots)$ to R_P,*
- *$R_P^{(\cdot,\cdot)}$ is a PPT security algorithm which gets parameters $(r_1, \ldots)$ from S_P.*
- *$\sigma : \mathbb{N} \to \mathbb{R}$ is a security threshold.*

We say that f is an implementation of P if $f \in F_P$.

There are three main differences between this definition and the one proposed by Reingold et al. Firstly, $\mathbb{P} = (A, B, C)$ is a triple of sets which describe the domain, co-domain and the challenge space respectively. Indeed, an implementation f is a function from A to B and the security game R_P can call f only on inputs in C (e.g. $A = \{0,1\}^*$ and $C = \{0,1\}^\lambda$). This modification enables us

to characterize implementations which are defined on more abstract mathematical models (e.g. groups, rings) rather than on $\{0,1\}^*$. Secondly, R_P is now an algorithm which expects black-box access to both an implementation f and an adversary $\mathcal{A}$. One can think of R_P as a security game, e.g. one-wayness game or IND-CCA game. Here, we want to associate for each pair $(f, \mathcal{A})$ a value in $[0,1]$ which corresponds to the probability of $\mathcal{A}$ winning the R_P game against f (see Definition 5). This adjustment helps us introduce the notion of a security loss. Eventually, we introduce a setup algorithm S_P which sends some values to R_P. This machine could as well send nothing or just provide fresh random coins which R_P would use in its game. However, this addition will be very useful in defining the multi-instance setting of a primitive. Informally, we can define a new security game R' which represents the security of P in the multi-user setting as follows: given parameters $(r_1, \ldots)$ from S_P, run n independent copies of R_P and send $(r_1, \ldots)$ as setup parameters to each of them. Then, R' returns a bit depending on what the n copies returned earlier. This idea is formally defined in Subsection 3.2.

Definition 5. *Let $P = \langle \mathbb{P}, S_P, F_P, R_P, \sigma \rangle$ be a primitive and $\mathbb{P} = (A, B, C)$. Take $f \in F_P$ and any algorithm $\mathcal{A}$. We define the advantage of $\mathcal{A}$ in breaking f as*

$$\mathsf{Adv}^{\mathrm{p}}_{f,\mathcal{A}}(\lambda) \coloneqq |\Pr[R_P^{f|_C,\mathcal{A}} = 1] - \sigma(\lambda)|$$

where the probability is defined over random coins in the system. We say that $\mathcal{A}$ $P-$breaks f if $\mathsf{Adv}^{\mathrm{p}}_{f,\mathcal{A}}(\lambda)$ is non-negligible. Primitive P is called secure if there exists an implementation f of P such that there are no PPT algorithms $\mathcal{A}$ that $P-$break f.

From the definition above one observes that we do not assign each pair $(f, \mathcal{A})$ a binary value (that would indicate, e.g., whether it satisfies a relation or not), but a probability. Therefore, the notion of a primitive from [43] is a generalisation of our definition. Indeed, any primitive in our sense can be easily transformed into a primitive from RTV definition: let $P = \langle \mathbb{P}, S_P, F_P, R_P, \sigma \rangle$ be a primitive in our sense and define a primitive $P' = \langle F'_P, R'_P \rangle$ such that $F'_P = F_P$ and $R'_P = \{(f, \mathcal{A}) | \mathcal{A}\ P\text{-breaks}\ f\}$. Then, primitives P and P' are equivalent. On the other hand, it is not clear if implication in the opposite direction holds. It is unknown if given relation set $\mathcal{R}$ we can construct a PPT algorithm R which could be equivalent to $\mathcal{R}$, meaning that $(f, \mathcal{A}) \in \mathcal{R} \iff |\Pr[R^{f,\mathcal{A}} = 1] - \sigma(\lambda)|$ is not negligible. An obvious brute force solution would be to check all elements of R but this could take exponential time. Despite the fact that our definition of a primitive is less general, it allows us to spot relations between two distinct advantages and consequently, to formally define a security loss.

Using our previous definitions we formalise the notion of a (tight) fully black-box reduction.

Definition 6 (C-tightness). *Let $P = \langle \mathbb{P}_1, S_P, F_P, R_P, \sigma \rangle$ and $Q = \langle \mathbb{P}_2, S_Q, F_Q, R_Q, \tau \rangle$ be primitives. Then, there is a fully black-box reduction from P to Q if there exist algorithms $G^{(\cdot)}, S^{(\cdot)}$ such that:*

- *for every implementation f of Q, G^f is an implementation of P,*
- *for every implementation f of Q and every (unbounded) algorithm $\mathcal{A}$, if $\mathcal{A}$ P-breaks G^f then $S^{\mathcal{A}}$ Q-breaks f.*

We require that G^f is PPT for every $f \in F_Q$, and that $S^{\mathcal{A}}$ is PPT for every PPT $\mathcal{A}$. Let $C\colon \mathbb{N} \to \mathbb{R}$ be a function. We say that the reduction is C-tight (and write $P \overset{C}{\hookrightarrow} Q$) if:

- *for every algorithm $\mathcal{A}$, $\boldsymbol{T}(S^{\mathcal{A}}) = \boldsymbol{T}(\mathcal{A}) + \mathsf{queries}(S^{\mathcal{A}}, \mathcal{A}) \cdot n_1(\lambda) + n_2(\lambda)$ for some $n_1, n_2 \in \mathsf{poly}(\lambda)$ that do not depend on $\mathcal{A}$*[7,8]*,*
- *for every implementation f of Q and every algorithm $\mathcal{A}$:*

$$\mathsf{Adv}^{\mathrm{p}}_{G^f,\mathcal{A}}(\lambda) \leq C(\lambda) \cdot \mathsf{Adv}^{\mathrm{q}}_{f,S^{\mathcal{A}}}(\lambda) + \mathit{negl}(\lambda). \tag{1}$$

*In particular, we say that the reduction is **fully-tight** if $C = 1$, **tight** if $C = a$ for $a \in \mathbb{N}$ and **almost-tight** if $C \in \mathsf{poly}(\lambda)$.*

We say that G is a generic construction of P from Q and S is an actual reduction. The first condition for a tight reduction states that the runtime of $S^{\mathcal{A}}$ should be about the same as the runtime of the adversary $\mathcal{A}$. This prevents the reduction S from running many copies of $\mathcal{A}$. An alternative way to formalise this condition would be to use the definition of a *time-success ratio* from [24] and combine it with the security loss C. However, in this paper we do not calculate exactly the runtime of $S^{\mathcal{A}}$ [9] and thus, we omit such formalities. Note that we allow a tight reduction to do some *small enough* amount of work proportional to the number of queries it gets from $\mathcal{A}$. Hence, some reductions, which are commonly considered as tight (e.g. ElGamal encryption scheme to DDH), would be also classified as tight by our definition. Further, the second condition from the definition of tight reduction assures that the success of an adversary $\mathcal{A}$ breaking the primitive is always about as large as the success of the reduction $S^{\mathcal{A}}$ breaking the other one.

We note that reductions with security loss L, which depends on the number of queries made by $\mathcal{A}$, are still almost-tight as long as $L = \mathsf{poly}(\lambda)$. This observation includes recent identity-based encryption (IBE) schemes [10,19] with security loss $O(\log Q)$, where Q is the number of IBE secret key queries. In these reductions, we have that $Q \leq 2^{\lambda}$ and consequently, they are still almost-tight in our definition.

We can get some simple but useful properties of tight reductions from the definition above. For example, they satisfy the transitivity property.

Lemma 1. *Let P, Q, R be primitives such that $P \overset{C}{\hookrightarrow} Q$ and $Q \overset{D}{\hookrightarrow} R$, where $C, D \in \mathsf{poly}(\lambda)$. Then, $P \overset{E}{\hookrightarrow} R$, where $E(\lambda) = C(\lambda) \cdot D(\lambda)$.*

[7] Recall that $\mathsf{queries}(S^{\mathcal{A}}, \mathcal{A})$ denotes the worst-case number of queries/messages from $\mathcal{A}$ to S.

[8] Runtime of $\mathcal{A}$ is included in $\mathbf{T}(S^{\mathcal{A}})$.

[9] As long as it is similar to the runtime of $\mathcal{A}$.

Proof. Let (G, S) be a tight reduction from P to Q and (G', S') be a tight reduction from Q to R. Define:

$$\bar{G}^{(\cdot)} = G^{G'^{(\cdot)}}, \bar{S}^{(\cdot)} = S'^{S^{(\cdot)}}.$$

We claim that $(\bar{G}, \bar{S})$ gives a reduction from P to R.

Take $f \in F_R$. Then, G'^f is an implementation of Q. Therefore, $\bar{G}^f = G^{G'^f}$ is an implementation of P. Now, take any $f \in F_R$ and algorithm $\mathcal{A}$. Note that there are some negligible functions $\mathsf{negl}_1(\lambda), \mathsf{negl}_2(\lambda)$ and $\mathsf{negl}(\lambda)$ such that:

$$\begin{aligned}
\mathsf{Adv}^{\mathrm{p}}_{\bar{G}^f,\mathcal{A}}(\lambda) &\leq C(\lambda) \cdot \mathsf{Adv}^{\mathrm{q}}_{G'^f,S^{\mathcal{A}}}(\lambda)| + \mathsf{negl}_1(\lambda) \\
&\leq C(\lambda) \cdot (D(\lambda) \cdot \mathsf{Adv}^{\mathrm{r}}_{f,\bar{S}^{\mathcal{A}}}(\lambda) + \mathsf{negl}_1(\lambda)) + \mathsf{negl}_2(\lambda) \\
&= C(\lambda)D(\lambda) \cdot \mathsf{Adv}^{\mathrm{r}}_{f,\bar{S}^{\mathcal{A}}}(\lambda) + \mathsf{negl}(\lambda) \\
&= E(\lambda) \cdot \mathsf{Adv}^{\mathrm{r}}_{f,\bar{S}^{\mathcal{A}}}(\lambda) + \mathsf{negl}(\lambda)
\end{aligned} \tag{2}$$

by the definition of the almost-tight reduction. Therefore, we have shown that $(\bar{G}, \bar{S})$ is a $E-$tight reduction from P to R. □

The notion of computational indistinguishability ($\approx$, see Section 2) can also be recast in our definitional framework. Let Ω_1, Ω_2, S be finite sets and $X_\lambda : \Omega_1 \to S, Y_\lambda : \Omega_2 \to S$ be random variables (parametrized over λ). We define $f_{X,Y} : \Omega_1 \times \Omega_2 \to S \times S$ as $f_{X,Y}(u) = (X_\lambda(u), Y_\lambda(u))$. Then, the $[X_\lambda; Y_\lambda]$ primitive is a tuple $\langle \mathbb{P}, S_{X,Y}, F_{X,Y}, R_{X,Y}, \frac{1}{2} \rangle$, where $\mathbb{P} = (\Omega_1 \times \Omega_2, S \times S, \Omega_1 \times \Omega_2)$, $F_{X,Y} = \{f_{X,Y}\}$, $S_{X,Y}$ sends $b \leftarrow\!\!\$\, \{0,1\}$ to $R_{X,Y}$, and $R_{X,Y}$ is the following game: generate $(u_0, u_1) \leftarrow\!\!\$\, \Omega_1 \times \Omega_2$ and send (u_0, u_1) to an implementation $f \in F_{X,Y}$. Then, get back (v_1, v_2) from f and take b from $S_{X,Y}$. Output v_b to an adversary $\mathcal{A}$ and eventually get back b' from $\mathcal{A}$. Then, the adversary $\mathcal{A}$ wins if $b = b'$.

3.2 Multi-instance Setting

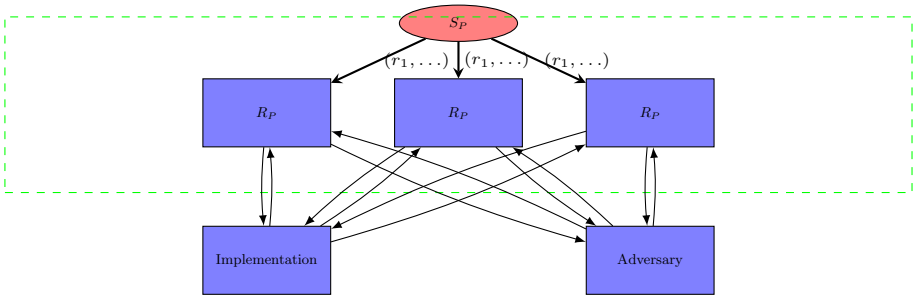

Fig. 1. New security game for the three-instance version of primitive $P = \langle \mathbb{P}, S_P, F_P, R_P, \sigma \rangle$ is described as the green dashed box.

Using notions from Subsection 3.1 we define the multi-instance setting of a primitive. Informally, this means that an adversary now interacts with (polynomially) many independent copies of the security game, so it gets more information. However, the winning condition would be almost the same as in the single-instance case, e.g. returning a preimage or guessing a bit (see Fig. 1). We define it formally for a primitive $P = \langle \mathbb{P}, S_P, F_P, R_P, \sigma \rangle$ in terms of the security algorithm R_P. We, however, provide two definitions of the multi-instance setting due to the differences between computational and decisional problems.

$\mathcal{R}_{\exists}^{f|_C,\mathcal{A}}(1^\lambda, r_1, \ldots)$

1 : Initialise $n(\lambda)$ ind. copies of $R_P^{f|_C,\mathcal{A}}$
2 : send $(r_1, \ldots)$ to each of them
3 : as the setup parameters
4 : **if** one of the copies returns 1
5 : **return** 1
6 : **else return** 0

$\mathcal{R}_{\forall}^{f|_C,\mathcal{A}}(1^\lambda, r_1, \ldots)$

1 : Initialise $n(\lambda)$ ind. copies of $R_P^{f|_C,\mathcal{A}}$
2 : send $(r_1, \ldots)$ to each of them
3 : as the setup parameters
4 : **if** all of the copies return 1
5 : **return** 1
6 : **else return** 0

Fig. 2. Security algorithms for $\exists\mathsf{MI}_n(P)$ (on the left) and $\forall\mathsf{MI}_n(P)$ (on the right). They get as input the security parameter λ (in unary) and parameters $(r_1, \ldots)$ from the setup algorithm S_Q. Here, we assume that $\mathcal{R}$ has black-box access to implementation f (restricted to the domain C, where $\mathbb{P} = (A, B, C)$) and adversary $\mathcal{A}$.

Definition 7 ($\exists/\forall$-Multi-instance Setting). *Let $n(\lambda)$ be a polynomial in λ and $P = \langle \mathbb{P}, S_P, F_P, R_P, \sigma \rangle$ be a primitive. Then, the $\exists\mathsf{MI}_n(P)$ primitive (resp. $\forall\mathsf{MI}_n(P)$) is a primitive $\langle \mathbb{P}, \mathcal{S}, \mathcal{F}, \mathcal{R}_{\exists}, \sigma \rangle$ (resp. $\langle \mathbb{P}, \mathcal{S}, \mathcal{F}, \mathcal{R}_{\forall}, \sigma \rangle$) such that $\mathcal{F} = F_P$, $\mathcal{S} = S_P$ and $\mathcal{R}_{\exists}$ (resp. $\mathcal{R}_{\forall}$) is defined in Fig. 2 (left) (resp. right).*

Now we provide examples of cryptographic primitives in the multi-instance setting using definitions above.

Example 1. Let $\mathsf{OWF} = \langle \mathbb{P}_{\mathsf{OWF}}, S_{\mathsf{OWF}}, F_{\mathsf{OWF}}, R_{\mathsf{OWF}}, 0 \rangle$ be the primitive of a one-way function. That is, $\mathbb{P}_{\mathsf{OWF}} = (A, B, C)$, F_{OWF} is a collection of functions $f : A \to B$, and R_{OWF} is a standard one-wayness game (i.e. R_{OWF} generates x uniformly at random from C, gets $f(x)$ by calling an implementation f and returns 1 only if adversary can guess the preimage of $f(x)$). For simplicity, when we write $\mathsf{OWF}_{p(\lambda)}$, where $p = \mathsf{poly}(\lambda)$, we mean OWF with $\mathbb{P}_{\mathsf{OWF}} = (\{0,1\}^*, \{0,1\}^*, \{0,1\}^{p(\lambda)})$ which is closer to the standard definition of a one-way function. The security game for $\exists\mathsf{MI}_n(\mathsf{OWF})$ would be as follows: it generates $x_1, \ldots, x_n$ independently and uniformly at random from C, then it gets $f(x_1), \ldots, f(x_n)$ by calling f and sends to adversary. The winning condition is that adversary returns a preimage of *one* of the values it was given i.e. x satisfying $f(x) = f(x_i)$ for some i. On the other hand, in $\forall\mathsf{MI}_n(\mathsf{OWF}_{p(\lambda)})$, the

adversary wins if it returns preimages for *all* values $f(x_1), \ldots, f(x_n)$. In practice, the $\exists\mathsf{MI}_n(\mathsf{OWF}_{p(\lambda)})$ setting is more common and therefore we focus on the former case.

Example 2. Let us define a primitive $\mathsf{PRG} = \langle \mathbb{P}_{\mathsf{PRG}}, S_{\mathsf{PRG}}, F_{\mathsf{PRG}}, R_{\mathsf{PRG}}, \frac{1}{2} \rangle$ of a pseudorandom generator where $\mathbb{P}_{\mathsf{PRG}} = (A, B, C)$ such that $C = A$ and $|B| > |A|$. Let F_{PRG} be a collection of functions $G : A \to B$ and let S_{PRG} generate a bit $b \leftarrow_{\$} \{0, 1\}$. We also define R_{PRG}, given bit b, to generate random $x \in C$ and output the image $G(x)$ of an implementation G if $b = 0$ and uniformly random value from B otherwise. The adversary wins if it can guess the bit b. Then, the security game for $\forall\mathsf{MI}_n(\mathsf{PRG})$ would be as follows: given bit b from the setup algorithm, generate and send $G(x_1), \ldots, G(x_n)$ for some $x_1, \ldots, x_n \in A$ if $b = 0$ or n uniformly random elements from B if $b = 1$. The winning condition here is that adversary guesses the bits for *all* of the n subgames which is equivalent to guessing the bit b.

Now, consider $\exists\mathsf{MI}_n(\mathsf{PRG})$. We claim that it is not secure. Note that in this case adversary wins if it guesses the bit b for *one* of the n subgames. In other words, it has n chances to guess b. Thus, an adversary which just sends 1 to a random subgame and 0 to another one will always win one of these two games (because $b \in \{0, 1\}$) and consequently, break $\exists\mathsf{MI}_n(\mathsf{PRG})$. Therefore, we only analyse the security of $\forall\mathsf{MI}_n(\mathsf{PRG})$.

Example 3. We define a primitive corresponding to an IND-CPA secure public-key encryption scheme as $\mathsf{PKE} = \langle \mathbb{P}_{\mathsf{PKE}}, S_{\mathsf{PKE}}, F_{\mathsf{PKE}}, R_{\mathsf{PKE}}, \frac{1}{2} \rangle$ where, as before, S_{PKE} does the sampling $b \leftarrow_{\$} \{0, 1\}$, R_{PKE} is the IND-CPA game and F_{PKE} contains encryption schemes. Note that $\exists\mathsf{MI}_n(\mathsf{PKE})$ is not secure due to the same reasons as the $\exists$-multi instance pseudorandom generator. On the other hand, $\forall\mathsf{MI}_n(\mathsf{PKE})$ yields the definition of an encryption scheme in the multi-user setting by Bellare et al. [6]. In a similar fashion we can define IND-CCA secure PKE schemes.

One observes that one can slightly change the definition of a primitive in order to get a definition of "multi-ciphertext setting". If we give S_{PKE} black-box access to an implementation and let it also generate keys (pk, sk) instead of R_{PKE} then the security game for $\forall\mathsf{MI}_n(\mathsf{PKE})$ is indeed an IND-CPA game with many ciphertexts. However, we do not consider the multi-ciphertext security in this paper so we omit defining it formally here.

We also introduce the notion of a primitive being *tightly extensible*, meaning that a reduction from its multi-instance setting admits the same security loss as in the single-instance case.

Definition 8. *Let P be a primitive and $C : \mathbb{N} \to \mathbb{R}$ be a function. Then, P is $(C, \forall)$-tightly extensible (resp. $(C, \exists)$-tightly extensible) with respect to primitive Q if:*

- $P \overset{C}{\hookrightarrow} Q$,
- $\forall n \in \mathsf{poly}(\lambda)$, $\forall\mathsf{MI}_n(P) \overset{C}{\hookrightarrow} Q$ *(resp.* $\exists\mathsf{MI}_n(P) \overset{C}{\hookrightarrow} Q$*)*.

Based on what we have already defined, we can formally state the main problem of this paper:

Problem. Suppose that $P \overset{C}{\hookrightarrow} Q$. Show that P is $(C, \exists(\text{ or } \forall))$-tightly extensible w.r.t. Q.

There are two standard approaches to show that P is tightly extensible w.r.t. Q. Namely, (i) somehow tightly reduce the multi-instance primitive to the single case, (ii) modify the former reduction and apply re-randomisation/self-reducibility techniques to eventually obtain the same security loss, or (iii) hide the factor of n in the statistical difference. In Section 4 we discuss these methods used in practical examples. When we apply (i), we use the following simple lemma.

Lemma 2. *Let n be a polynomial in λ and let P and Q be primitives such that $P = \langle \mathbb{P}_1, S_P, F_P, R_P, \sigma \rangle \overset{C}{\hookrightarrow} Q = \langle \mathbb{P}_2, S_Q, F_Q, R_Q, \tau \rangle$ and let (G, S) be such a reduction. Define P/Q to be the primitive $\langle \mathbb{P}_1, S_P, \mathcal{F}, R_P, \sigma \rangle$ such that $\mathcal{F} = \{G^f : f \in F_Q\}$. Then, if $\exists \mathsf{MI}_n(P) \overset{D}{\hookrightarrow} P/Q$ (resp. $\forall \mathsf{MI}_n(P) \overset{D}{\hookrightarrow} P/Q$) then $\exists \mathsf{MI}_n(P) \overset{C \cdot D}{\hookrightarrow} Q$ (resp. $\forall \mathsf{MI}_n(P) \overset{C \cdot D}{\hookrightarrow} Q$). In particular, if $D = 1$ then P is $(C, \exists)$ (resp. $(C, \forall)$)-tightly extensible w.r.t. Q.*

Proof. Using the notation above, it is easy to see that $P \overset{C}{\hookrightarrow} Q$ implies $P/Q \overset{C}{\hookrightarrow} Q$. The result holds by this simple observation and by Lemma 1. □

4 Tightly Extensible Primitives

We provide a few constructions of tightly extensible primitives from more general primitives. In principle, we first take a tight reduction from the single-instance primitive and see if we can extend it (in a tight way) to the multi-instance setting or otherwise, use the Lemma 2. In the first subsection, we demonstrate the use of definitions from Section 3, and derive formal proofs. In the later subsections, however, we focus more on showing novel techniques to extend reductions to the multi-instance setting.

4.1 One-wayness of Collision-Resistant Hash Functions

It is well-known that signatures schemes can be constructed from one-way functions [44,32,23]. Concretely, we can use the Lamport construction [33] of a one-time signature scheme from one-way functions, and then extend these one-time signatures to full signatures using Merkle trees [37]. The corresponding reduction is far from tight, and not known to scale well to the multi-user setting. Here, we consider the same construction under a slightly stronger assumption (namely, collision-resistance) of the used one-way function. We will show that this stronger assumption enables a much tighter security reduction.

We start by defining collision-resistant hash functions w.r.t. the definition of a primitive from Section 3. Define $\mathsf{CRHF} = \langle \mathbb{P}_{\mathsf{CRHF}}, S_{\mathsf{CRHF}}, F_{\mathsf{CRHF}}, R_{\mathsf{CRHF}}, 0 \rangle$ as

follows: $\mathbb{P}_{\mathsf{CRHF}} = (\{0,1\}^*, \{0,1\}^\lambda, \{0,1\}^*)$, S_{CRHF} returns no parameters, $F_{\mathsf{CRHF}} = \{h : \{0,1\}^* \to \{0,1\}^\lambda\}$ and R_{CRHF} is the collision resistance game, i.e. it waits until it gets (x, x') from adversary, checks if $x \neq x'$ and then calls the implementation h to check if $h(x) = h(x')$.

Let us define an *adaptive one-way function*. Specifically, we denote $\mathsf{AOWF} = \langle \mathbb{P}_{\mathsf{OWF}}, S_{\mathsf{OWF}}, F_{\mathsf{OWF}}, R_{\mathsf{AOWF}}, 0 \rangle$, where $\mathbb{P}_{\mathsf{OWF}}, S_{\mathsf{OWF}}, F_{\mathsf{OWF}}$ are defined in Example 1. Also, R_{AOWF} is defined identically as R_{OWF}, but here adversary can also send a message `lose`, in which case R_{AOWF} outputs back the challenge preimage x and automatically returns 0. Similarly as in Example 1, we use the notation $\mathsf{AOWF}_{p(\lambda)}$ when $\mathbb{P}_{\mathsf{OWF}} = (\{0,1\}^*, \{0,1\}^*, \{0,1\}^{p(\lambda)})$. Clearly, security of OWF implies (tightly) the AOWF security. Interestingly, we cannot conclude the same for $\exists\mathsf{MI}_n(\mathsf{OWF})$ and $\exists\mathsf{MI}_n(\mathsf{AOWF})$.

Damgård [12] showed that h, when considered as a function with domain $\{0,1\}^{\lambda+1}$, is also a (adaptive) one-way function. Indeed, if there exists an adversary $\mathcal{A}$ which can find preimage of $h(x)$ for uniformly random x, then we can construct adversary $S^{\mathcal{A}}$ which breaks the collision-resistance of h as follows: given h, choose random x and send $h(x)$ to $\mathcal{A}$. Let x' be the output of $\mathcal{A}$. Then, return the pair (x, x'). Note that with non-negligible probability (over x and x'), we have $x \neq x'$. Hence, adversary $S^{\mathcal{A}}$ wins the collision-resistance game and additionally, the reduction itself is clearly tight.

Damgård's argument in fact nicely extends to the multi-instance setting:

Theorem 1. *Let CRHF be the primitive of a collision-resistant hash function and $\mathsf{AOWF}_{2\lambda}$ be the primitive of a one-way function defined in Example 1. Then, $\mathsf{AOWF}_{2\lambda}$ is $(2, \exists)$-tightly extensible w.r.t. CRHF.*

Proof. We first reprove that $\mathsf{AOWF}_{2\lambda} \overset{2}{\hookrightarrow} \mathsf{CRHF}$. Let us define PPT algorithms $G^{\cdot}$ and $S^{\cdot}$ as in Fig. 3. Clearly, both G and S run in polynomial time. One can observe that G is a generic construction. Indeed, G only forwards all the queries from/to an implementation and hence, $\forall h \in F_{\mathsf{CRHF}}$, $G^h = h$. In particular, $G^h : \{0,1\}^* \to \{0,1\}^\lambda$ is a function, so $G^h \in F_{\mathsf{OWF}}$. Now, suppose that there

$G^h(1^\lambda)$

1 : **if** G is queried on x:
2 : send x to h
3 : get y from h
4 : **return** y

$S^{\mathcal{A}}(1^\lambda)$

1 : $x \leftarrow\!\!\$\ \{0,1\}^{2\lambda}$
2 : send $h(x)$ to $\mathcal{A}$
3 : if $\mathcal{A}$ outputs `lose`, send x to $\mathcal{A}$ and abort
4 : Othewise, get x' from $\mathcal{A}$ and **return** (x, x')

Fig. 3. PPT Algorithms G and S.

exists an algorithm $\mathcal{A}$ which $\mathsf{OWF}_{2\lambda}$-breaks G^h. We want to prove that $S^{\mathcal{A}}$

CRHF-breaks h. Using the variables x and x' from Fig. 3, one observes that:

$$\begin{aligned}\mathsf{Adv}^{\mathsf{crhf}}_{h,S^{\mathcal{A}}}(\lambda) &= \Pr[R^{h,S^{\mathcal{A}}}_{\mathsf{CRHF}} = 1] \\ &= \Pr[x \neq x' \wedge h(x) = h(x')] \\ &\geq \Pr[x \neq x' \wedge h(x) = h(x') \mid |h^{-1}(h(x))| \geq 2] \cdot \Pr[|h^{-1}(h(x))| \geq 2],\end{aligned} \tag{3}$$

where $h^{-1}(u) = \{v \in \{0,1\}^{2\lambda} : h(v) = u\}$. Clearly, $|h^{-1}(h(x))| \geq 1$. Note that $\Pr[x \neq x' \mid h(x) = h(x') \wedge |h^{-1}(h(x))| \geq 2] \geq \frac{1}{2}$ since adversary $\mathcal{A}$ does not know, given $h(x)$, if S chose exactly x or some other element in $h^{-1}(h(x))$ (it exists $h^{-1}(h(x)) \geq 2$). Hence, if we denote $X = |h^{-1}(h(x))|$, then we eventually have $\frac{X-1}{X} \geq 1/2$. Using this observation and the fact that $h(x)$ is generated by S with the same distribution as the challenge by R_{OWF} , we deduce that:

$$\begin{aligned}\mathsf{Adv}^{\mathsf{crhf}}_{h,S^{\mathcal{A}}}(\lambda) &\geq \frac{1}{2}\Pr[h(x) = h(x') \mid |h^{-1}(h(x))| \geq 2] \cdot \Pr[|h^{-1}(h(x))| \geq 2] \\ &\geq \frac{1}{2}\Pr[h(x) = h(x')] - \frac{1}{2}\Pr[|h^{-1}(h(x))| = 1] \\ &\geq \frac{1}{2}\mathsf{Adv}^{\mathsf{owf}}_{G^h,\mathcal{A}}(\lambda) - \frac{1}{2}\Pr[|h^{-1}(h(x))| = 1].\end{aligned} \tag{4}$$

The only thing to compute here is $\Pr[|h^{-1}(h(x))| = 1]$. Let $a_1, \ldots, a_m \in \{0,1\}^{2\lambda}$ be the bit-strings such that $|h^{-1}(h(a_i))| = 1$ for $i = 1, \ldots, m$. Clearly, we have that $h(a_1), \ldots, h(a_m)$ are pairwise distinct. Also $\{h(a_1), \ldots, h(a_m)\} \subset \{0,1\}^{\lambda}$, and therefore $m \leq 2^{\lambda}$. Thus, $\Pr[|h^{-1}(h(x))| = 1] = \frac{m}{2^{2\lambda}} \leq \frac{2^{\lambda}}{2^{2\lambda}} = \frac{1}{2^{\lambda}}$. By substituting this result into Equation 4 and reordering both sides we get:

$$2\mathsf{Adv}^{\mathsf{crhf}}_{h,S^{\mathcal{A}}}(\lambda) + \frac{1}{2^{\lambda}} \geq \mathsf{Adv}^{\mathsf{aowf}}_{G^h,\mathcal{A}}(\lambda),$$

which concludes that $\mathsf{AOWF}_{2\lambda} \overset{2}{\hookrightarrow} \mathsf{CRHF}$.

Now, we have to prove that $\exists\mathsf{MI}_n(\mathsf{AOWF}_{2\lambda}) \overset{2}{\hookrightarrow} \mathsf{CRHF}$ for any $n \in \mathsf{poly}(\lambda)$. Denote $n = n(\lambda)$. We define the reduction $(\bar{G}, \bar{S})$ as in the Fig. 4. One observes that $\bar{G}$ and $\bar{S}$ are both PPT algorithms, and also $\bar{G} = G$ is a generic construction of $\exists\mathsf{MI}_n(\mathsf{AOWF}_{2\lambda})$ from CRHF. Suppose that there exists an algorithm $\mathcal{A}$ which $\mathsf{OWF}_{2\lambda}$-breaks G^h for some $h \in F_{\mathsf{CRHF}}$. We extend the previous argument for many instances as follows (the probabilities are calculated over $x_1, \ldots, x_n, x'$ which are defined in Fig. 4):

$$\begin{aligned}\mathsf{Adv}^{\mathsf{crhf}}_{h,\bar{S}^{\mathcal{A}}}(\lambda) &= \Pr[R^{h,\bar{S}^{\mathcal{A}}}_{\mathsf{CRHF}} = 1] \\ &= \Pr[x_i \neq x' \wedge h(x_i) = h(x')] \\ &\geq \Pr[x_i \neq x' \wedge h(x_i) = h(x') \mid E] \cdot \Pr[E],\end{aligned} \tag{5}$$

$\bar{G}^h(1^\lambda)$

1 : **if** G is queried on x:
2 : send x to h
3 : get y from h
4 : **return** y

$\bar{S}^{\mathcal{A}}(1^\lambda)$

1 : $x_1, \ldots, x_n \leftarrow\!\!\$\, \{0,1\}^{2\lambda}, U = \emptyset$
2 : send $h(x_1), \ldots, h(x_n)$ to $\mathcal{A}$
3 : if $\mathcal{A}$ returns `lose` in the i-th subgame:
4 : send back x_i and set $U \leftarrow U \cup \{i\}$
5 : if $\mathcal{A}$ returns some x' in the i-th subgame:
6 : **if** $h(x') = h(x_i)$, **return** (x', x_i)

Fig. 4. PPT Algorithms G and S.

where $E = \bigwedge_{i=1}^{n} |h^{-1}(h(x_i))| \geq 2$. In the similar fashion as before, we have that $\Pr[x_i \neq x' \mid h(x_i) = h(x') \wedge E] \geq \frac{1}{2}$. Hence,

$$\begin{aligned} \mathsf{Adv}^{\mathsf{crhf}}_{h,\bar{S}^{\mathcal{A}}}(\lambda) &\geq \frac{1}{2} \Pr[h(x_i) = h(x') \mid E] \cdot \Pr[E] \\ &\geq \frac{1}{2}\mathsf{Adv}^{\mathsf{mi_n(aowf)}}_{\bar{G}^h,\mathcal{A}}(\lambda) - \frac{1}{2}\Pr[\neg E]. \end{aligned} \tag{6}$$

By the union bound, we compute:

$$\Pr[\neg E] = \Pr[\bigvee_{i=1}^{n} |h^{-1}(h(x_i))| = 1] \leq \sum_{i=1}^{n} \Pr[|h^{-1}(h(x_i))| = 1] = \frac{n}{2^\lambda}. \tag{7}$$

Eventually, by reordering both sides of Equation 6 we get that $\exists \mathsf{MI}_n(\mathsf{AOWF}_{2\lambda}) \overset{2}{\hookrightarrow} \mathsf{CRHF}$:

$$2\mathsf{Adv}^{\mathsf{crhf}}_{h,\bar{S}^{\mathcal{A}}}(\lambda) + \frac{n}{2^\lambda} \geq \mathsf{Adv}^{\mathsf{mi_n(aowf)}}_{\bar{G}^h,\mathcal{A}}(\lambda).$$

□

Even though this proof is not complicated, the theorem itself can be useful in constructing secure one-time signature schemes (OTS) with small security loss. Let us first define OTS using the definition of a primitive from Section 3. That is, $\mathsf{OTS} = \langle \mathbb{P}_{\mathsf{OTS}}, S_{\mathsf{OTS}}, F_{\mathsf{OTS}}, R_{\mathsf{OTS}}, 0 \rangle$, where $\mathbb{P} = (\{0,1\}^*, \{0,1\}^*, \{0,1\}^*)$, S_{OTS} does not send any global parameters, F_{OTS} is the set of all signature schemes and R_{OTS} represents the EUF-OTCMA game.

Let f be a one-way function and let n be a polynomial in λ. Consider the Lamport's one-time signature scheme [33] $(\mathsf{Gen}_L; \mathsf{Sign}_L; \mathsf{Ver}_L)$ for messages of length $n = n(\lambda)$, meaning:

- $\mathsf{Gen}_L(\lambda)$: generate $2n$ random values $x_{i,j}$ and compute $y_{i,j} = f(x_{i,j})$ for $j \in \{0,1\}, i \in \{1, \ldots, n\}$. Then, set $sk = \{x_{1,0}, \ldots, x_{n,0}, x_{1,1}, \ldots, x_{n,1}\}$ and $vk = \{y_{1,0}, \ldots, y_{n,0}, y_{1,1}, \ldots, y_{n,1}\}$.
- $\mathsf{Sign}_L(m, sk)$: for message $m = m_1 m_2 \ldots m_n$, output $\sigma = \sigma_1 \sigma_2 \ldots \sigma_n$ where $\sigma_i = x_{i,m_i}$.

- $\mathsf{Ver}_L(vk, m, \sigma)$: for signature $\sigma = \sigma_1\sigma_2 \dots \sigma_n$ and message $m = m_1 m_2 \dots m_n$, check if $f(\sigma_i) = y_{i,m_i}$ for all $i = 1, \dots, n$. If so, return 1 and 0 otherwise.

Lamport proved that this scheme is EUF-OTCMA secure by giving a reduction to the one-wayness of f which admits the security loss of $2n$. We show how to tightly reduce it to the adaptive one-wayness of f in the multi-instance setting.

Theorem 2. *Let $n \in \mathsf{poly}(\lambda)$ and OTS and $\mathsf{AOWF}_{2\lambda}$ be the primitives of a one-time signature scheme and one-way function respectively. Then, there exists a fully-tight fully black-box reduction from OTS to $\exists\mathsf{MI}_{2n}(\mathsf{AOWF}_{2\lambda})$.*

Proof. As usual, let us define PPT algorithms G and S as in Fig. 5. We can

$G^f(1^\lambda)$
if G is queried on $\mathsf{Gen}(1^\lambda)$:
run $\mathsf{Gen}_L(\lambda)$ by calling f
if G is queried on $\mathsf{Sign}(m, sk)$:
run $\mathsf{Sign}_L(m, sk)$
if G is queried on $\mathsf{Ver}(vk, m, \sigma)$:
run $\mathsf{Ver}_L(vk, m, \sigma)$

$S^{\mathcal{A}}(1^\lambda)$
receive $(y_{1,0}, \dots, y_{n,0}, y_{1,1}, \dots, y_{n,1})$
$vk = (y_{1,0}, \dots, y_{n,1})$
send vk to $\mathcal{A}$
if $\mathcal{A}$ requests a signature on m':
for $i = 1, ..., n$
send `lose` to $i + m'_i \cdot n$-th subgame
get back x_{i,m'_i}
send $\sigma = x_{1,m'_1} ... x_{n,m'_n}$ to $\mathcal{A}$
if $\mathcal{A}$ outputs $(m, \sigma = \sigma_1, \dots, \sigma_n)$:
find i so that $m_i \neq m'_i$
send σ_i to $i + m_i \cdot n$-th subgame

Fig. 5. PPT Algorithms G and S. Here, $(\mathsf{Gen}_L; \mathsf{Sign}_L; \mathsf{Ver}_L)$ is the Lamport signature scheme.

see that G represents the Lamport construction of a one-time signature scheme. In particular, G is a signature scheme and also, G is a generic construction of OTS from $\exists\mathsf{MI}_{2n}(\mathsf{AOWF}_{2\lambda})$. Now, let us consider S and suppose that S is given $f(x_{1,0}), \dots, f(x_{n,1})$ from $R_{\exists\mathsf{MI}_{2n}(\mathsf{AOWF}_{2\lambda})}$. If $\mathcal{A}$ requests a signature on $m' = m'_1 ... m'_n$ then we abort $1 + m'_1 \cdot n$-th,...,$n + m'_n \cdot n$-th subgames of $R_{\exists\mathsf{MI}_{2n}(\mathsf{AOWF}_{2\lambda})}$ and consequently, get back preimages $x_{i,m'_i}, ..., x_{n,m'_n}$. Eventually, when $\mathcal{A}$ outputs a valid forgery (m, σ), then we must have $m \neq m'$. So, there exists some index i such that $m_i \neq m'_i$[10]. This means that $f(\sigma_i) = y_{i,m_i}$ and therefore, S wins the $i + m_i \cdot n$-th subgame of $R_{\exists\mathsf{MI}_{2n}(\mathsf{AOWF}_{2\lambda})}$. Hence, we end up with a tight reduction which admits security loss 1. □

Combining the previous two results we get that it is possible to construct a one-time signature scheme, which can be reduced to a collision-resistant hash

[10] If $\mathcal{A}$ has not requested a signature before, then we just set $i = 1$.

function with the loss of 2. Thus, we managed to eliminate the factor n and if one wants to apply Merkle trees, the overall reduction to CRHF from a secure signature scheme would have the security loss of $O(l)$, where l is the number of signing queries.

4.2 A Rerandomisable Hard-core Predicate

Blum and Micali [8] provided a construction of a pseudorandom generator from a one-way permutation. Let f be a one-way permutation and b be its hard-core predicate. Then, the function $G(x) = f(x)||b(x)$ is a pseudo-random generator. Now, our aim is to construct, given f and b, a tightly extensible pseudo-random generator w.r.t. some certain mathematical assumption. We find suitable f and b such that we can apply Lemma 2. Note that in order to do this, we need a rerandomisation property from these functions. For instance, given $b(x)$ and a value a, we should somehow be able to compute $b(ax)$. We will choose a one-way permutation on a group where the discrete logarithm problem is hard and use properties of Legendre symbol to construct a rerandomisable hard-core predicate.
BACKGROUND. Let $a \in \mathbb{Z}$ be an integer and p be a prime number such that $p \nmid a$. We say that a is a quadratic residue mod p if there exists $x \in \mathbb{Z} \setminus \{0\}$ so that $x^2 \equiv a(\mod p)$. If there is no such x, then a is a quadratic non-residue mod p. It is a well-known fact that in $\{1, \ldots, p-1\}$ there are exactly $(p-1)/2$ quadratic residues (and also $(p-1)/2$ quadratic non-residues) mod p. Clearly, the quadratic residues form a subgroup of $\mathbb{Z}_p^*$.

The Legendre symbol is defined as follows:

$$(\frac{a}{p}) = \begin{cases} 0 & \text{if } p|a \\ 1 & \text{if } a \text{ is a quadratic residue mod } p \\ -1 & \text{if } a \text{ is a quadratic non-residue mod } p \end{cases}$$

One of useful properties of the Legendre symbol is that it is homomorphic, meaning $(\frac{a}{p})(\frac{b}{p}) = (\frac{ab}{p})$ for any a, b. Moreover, by the Euler's criterion, the Legendre symbol can be computed efficiently.
THE LGR PROBLEM. We propose a new computational problem, called Legendre Problem (LGR).

Definition 9. *Let p be a λ-bit prime number and let G be a group of order p with generator g. We say that the Legendre Problem (LGR) is hard in G if for all PPT algorithms $\mathcal{A}$,*

$$\mathsf{Adv}^{lgr}_{G,\mathcal{A}}(\lambda) := \Pr[b \leftarrow_\$ \mathcal{A}(\langle G \rangle, p, g, g^x) : x \leftarrow_\$ \mathbb{Z}_p, b = \frac{1 + (\frac{x}{p})}{2}] = \frac{1}{2} + \mathsf{negl}(\lambda).$$

Equivalently, define a primitive $\mathsf{LGR}_G = \langle \mathbb{P}_{LGR}, S_{LGR}, F_{LGR}, R_{LGR}, \frac{1}{2} \rangle$ such that $\mathbb{P}_{LGR} = (\mathbb{Z}_p, G, \mathbb{Z}_p)$, S_{LGR} sends parameters (G, p, g) to R_P and $F_{LGR} = \{f_g\}$, where f_g is defined by $a \mapsto g^a$. The security game R_{LGR} first chooses random $x \leftarrow_\$ \mathbb{Z}_p$, sends x to an implementation $f \in F_{LGR}$ of LGR_G and gets back y. Then, it outputs y to adversary $\mathcal{A}$. When it receives $b \in \{0, 1\}$ from $\mathcal{A}$, it returns a value of the statement $b = (1 + (\frac{x}{p}))/2$.

One observes that the Legendre Problem is not harder than the discrete logarithm problem. Indeed, given g^x, solving the discrete logarithm problem yields x, from which one can directly compute $(\frac{x}{p})$. A more interesting question is whether LGR is as hard as DLOG. We first show that LGR is at least as hard as DDH.

Lemma 3. *Let G be a group of prime order p and suppose that DDH is hard in G. Then, the Legendre Problem is also hard in G.*

Proof. Suppose there exists a PPT algorithm $\mathcal{A}$ which solves LGR with a non-negligible advantage. We construct a PPT algorithm $S^{\mathcal{A}}$ which wins the DDH game as follows. S first generates random $a, b, c \leftarrow_{\$} \mathbb{Z}_p$. Next, given g^x, g^x, g^z, where $z = xy$ or z is random, S sends g^{ax}, g^{by}, g^{cz} to $\mathcal{A}$ and gets $(\frac{ax}{p}), (\frac{by}{p}), (\frac{cz}{p})$ respectively. Then, S simply extracts $(\frac{x}{p}), (\frac{y}{p}), (\frac{z}{p})$ (e.g. by $(\frac{x}{p})(\frac{a}{p}) = (\frac{ax}{p})$) and returns the value of statement $(\frac{x}{p})(\frac{y}{p}) = (\frac{z}{p})$. Note that if $z = xy$ then this will always be true. On the other hand, if z is random, then the probability that $(\frac{z}{p}) = b$ for $b \in \{-1, 1\}$ is $1/2$. All in all, S wins the DDH game with non-negligible probability. The reduction itself, however, is far from tight [11]. □

Legendre's symbol has already been used in building pseudorandom generators [13,36,46]. For example, Damgård [13] applied specific subsequences of the sequence of Legendre symbols modulo a prime to obtain a pseudorandom generator. Security of such constructions, however, rely on empirical results or additional unproven conjectures.

Our Construction. We build a tightly extensible pseudorandom generator with respect to the Legendre assumption. Let G be a *densely presentable* group with generator g of prime order p, i.e. a group which satisfies the property that for $x \in \mathbb{Z}_p$, the map $x \mapsto g^x$ is a permutation (e.g. [9]). We define PRG_G to be the primitive from Example 2 with $\mathbb{P} = (\mathbb{Z}_p, G \times \{0,1\}, \mathbb{Z}_p)$. Then, the construction is presented as follows.

Theorem 3. *Let G be a densely presentable group of prime order p where the Legendre problem is hard. Denote $g \in G$ to be a generator of G. Then, PRG_G is $(2, \forall)$-tightly extensible w.r.t. LGR_G.*

Proof. Define $f : \mathbb{Z}_p \to G$ as $x \mapsto g^x$, $b : \mathbb{Z}_p \to \{0,1\}$ as $x \mapsto (1 + (\frac{x}{p}))/2$ and eventually, $F(x) \coloneqq f(x)||b(x)$. Clearly, if the Legendre Problem is hard in G then f is a one-way permutation and b is a hard-core predicate for f.

Blum and Micali [8] showed that F is a pseudorandom generator. In this case, the generic reduction is fully-tight. Thus, by Lemma 2 it is enough to reduce the multi-instance setting of F to the single-instance with security loss 2. Suppose that there exists an adversary $\mathcal{A}$ which can win the n-multi-instance game for the pseudorandom generator F. We construct an adversary $S^{\mathcal{A}}$ that wins a single-instance game as follows. Given $(u||v)$ $(u \in G, v \in \{0,1\})$, toss a fair coin n times. For the i-th trial, where $i = 1, \ldots, n$, if we get heads - generate random

[11] On the other hand, one can actually derive a simple tight reduction from $\forall\mathsf{MI}_3(\mathsf{LGR})$ to DDH.

$x_i \leftarrow_\$ \mathbb{Z}_p$ and set $y_i = F(x_i)$. On the other hand, if the coin comes out tails, we choose random $a_i \leftarrow_\$ \mathbb{Z}_p$ and set $y_i = u^{a_i}||v_i$ where $v_i = (1 + (\frac{a_i}{p})(2v - 1))/2$. Then, send $y_1, \ldots, y_n$ to adversary $\mathcal{A}$ and eventually output what $\mathcal{A}$ returns.

Let us assume that $x \neq 0$ [12]. In order to analyse correctness of this reduction, we have to consider two cases. First, suppose that $u = g^x$ for some x. Then, $v = (1 + (\frac{x}{p}))/2$ or $v = (1 - (\frac{x}{p}))/2$.

Case 1: $v = (1+(\frac{x}{p}))/2$. Let us fix i and consider the i-th trial of flipping the coin. If it comes out heads, then $y_i = F(x_i)$ for randomly chosen x_i. Otherwise, we get that $v_i = (1 + (\frac{a_i}{p})(\frac{x}{p}))/2 = (1 + (\frac{a_i x}{p}))/2$ and thus $y_i = F(x_i)$ where $x_i = a_i x$ for uniformly random a_i. Therefore, for each i we have $y_i = F(x_i)$ and also $x_1, \ldots, x_n$ are independently, uniformly random in $\mathbb{Z}_p$.

Case 2: $v = (1 - (\frac{x}{p}))/2$. Let us denote $y_i = g^{s_i}||t_i$ where $s_i \in \mathbb{Z}_p, t \in \{0, 1\}$. We need to show that for every $\alpha_1, \ldots, \alpha_n \in \mathbb{Z}_p$ and $\beta_1, \ldots, \beta_n \in \{0, 1\}$ we have

$$\Pr[s_1 = \alpha_1, \ldots, s_n = \alpha_n, t_1 = \beta_1, \ldots, t_n = \beta_n] = \frac{1}{2^n p^n}.$$

This is the same as showing $\Pr[t_1 = \beta_1, \ldots, t_n = \beta_n | s_1 = \alpha_1, \ldots, s_n = \alpha_n] \cdot \Pr[s_1 = \alpha_1, \ldots, s_n = \alpha_n] = \frac{1}{2^n p^n}$. Note that $\Pr[s_1 = \alpha_1, \ldots, s_n = \alpha_n] = 1/p^n$ because $x \neq 0$ and $s_i = a_i x$ or $s_i = x_i$ for random a_i, x_i. Now, consider $\Pr[t_1 = \beta_1, \ldots, t_n = \beta_n | s_1 = \alpha_1, \ldots, s_n = \alpha_n]$. Clearly, this is the same as $\Pr[t_1 = \beta_1 | s_1 = \alpha_1, \ldots, s_n = \alpha_n]^n$. Firstly, assume that $\beta_1 = (1 + (\frac{\alpha_1}{p}))/2$ and let X denote the output of tossing a coin for the first time (and say H - heads, T - tails). Then, $\Pr[t_1 = \beta_1 | X = H, s_1 = \alpha_1, \ldots, s_n = \alpha_n] = 1$ because if $S^{\mathcal{A}}$ gets heads then it generates fresh $F(x_1)$ and in this case $x_1 = s_1 = \alpha_1$ so $t_1 = (1 + (\alpha_1/p))/2 = \beta_1$. On the other hand, if the coin comes out tails then we have $\alpha_i = s_i = a_i x$. Also, $t_1 = v_1 = (1 - (\frac{a_i}{p})(\frac{x}{p})/2 = (1 - (\frac{\alpha_1}{p}))/2 \neq \beta_1$ and hence $\Pr[t_1 = \beta_1 | X = T, s_1 = \alpha_1, \ldots, s_n = \alpha_n] = 0$. Consequently, we get $\Pr[t_1 = (1+(\frac{\alpha_1}{p}))/2 | s_1 = \alpha_1, \ldots, s_n = \alpha_n] = (1+0)/2 = 1/2$. Using a similar argument it can be shown that $\Pr[t_1 = (1 - (\frac{\alpha_1}{p}))/2 | s_1 = \alpha_1, \ldots, s_n = \alpha_n] = 1/2$. Thus, $\Pr[t_1 = \beta_1, \ldots, t_n = \beta_n | s_1 = \alpha_1, \ldots, s_n = \alpha_n] = 1/2^n$ and the result holds. In particular, if $v = (1 - (\frac{x}{p}))/2$ then all the values y_i sent to $\mathcal{A}$ look independently and uniformly random.

In conclusion, we obtain the following results:

$$\Pr[1 \leftarrow_\$ S^{\mathcal{A}}(u||v) \mid (u||v) \leftarrow_\$ F] = \Pr[1 \leftarrow_\$ \mathcal{A}(y_1, ..., y_n) \mid (y_1, ..., y_n) \leftarrow_\$ F],$$

and also $\Pr[1 \leftarrow_\$ S^{\mathcal{A}}(u||v) \mid (u||v) \leftarrow_\$ G \times \{0, 1\}] = \alpha$, where

$$\begin{aligned} \alpha &\geq \frac{1}{2} \Pr[1 \leftarrow_\$ S^{\mathcal{A}}(u||v) \mid (u||v) \leftarrow_\$ G \times \{0, 1\} \wedge v = (1 - (\frac{x}{p}))/2] \\ &= \frac{1}{2} \Pr[1 \leftarrow_\$ \mathcal{A}(y_1, ..., y_n) \mid (y_1, ..., y_n) \leftarrow_\$ G \times \{0, 1\}]. \end{aligned} \tag{8}$$

Thus, we get a tight fully black-box reduction from $\forall \mathsf{MI}_n(\mathsf{PRG}_G)$ to $\mathsf{PRG}_G/\mathsf{LGR}_G$ which admits the security loss of 2. Hence, by Lemma 2 the result holds. □

[12] This occurs with an overwhelming probability.

A rerandomisable hard-core predicate can be potentially also very useful in constructing a tightly extensible encryption scheme out of a computational problem (rather than a decisional one). However, it is not known how it can concretely be used, for example, because we do not have any information about functions related to DLOG being trapdoor one-way functions.

4.3 Tightly Extensible Lossy Trapdoor Functions

Our aim is to construct IND-CPA secure encryption schemes in the multi-user setting from lossy trapdoor functions in a tight way. In order to do so, we introduce tightly secure LTDFs in the multi-instance setting. Let $\mathsf{LTDF} = \langle \mathbb{P}_{\mathsf{LTDF}}, S_{\mathsf{LTDF}}, F_{\mathsf{LTDF}}, R_{\mathsf{LTDF}}, \frac{1}{2} \rangle$ be a primitive of lossy trapdoor functions, i.e. $\mathbb{P}_{\mathsf{LTDF}}$ defines the domain, codomain and challenge space, F_{LTDF} is a collection of LTDFs, S_{LTDF} provides a random bit to R_{LTDF} and R_{LTDF} runs a game where the goal is to distinguish a lossy function from an injective one. As before, let PKE be a public-key encryption scheme with its security game being IND-CPA. For exact same reasons as in PRGs or PKE cases, it is sensible only to consider $\forall\mathsf{MI}_n(\mathsf{LTDF})$ for the multi-instance setting. Also, denote DDH as a primitive representing the DDH assumption (formal definition is not required here). Peikert et al. [41] showed that:

$$\mathsf{PKE} \overset{2}{\hookrightarrow} \mathsf{LTDF} \overset{\lambda}{\hookrightarrow} \mathsf{DDH}.$$

Using these results, we show the following:

$$\forall\mathsf{MI}_n(\mathsf{PKE}) \overset{2}{\hookrightarrow} \forall\mathsf{MI}_n(\mathsf{LTDF}) \overset{1}{\hookrightarrow} \mathsf{LTDF/DDH} \overset{\lambda}{\hookrightarrow} \mathsf{DDH}. \tag{9}$$

Primitive $\mathsf{LTDF/DDH}$ is a construction of a lossy trapdoor function from a DDH group by Peikert et al.

Clearly, $\mathsf{LTDF} \overset{2}{\hookrightarrow} \mathsf{DDH}$ implies $\mathsf{LTDF/DDH} \overset{2}{\hookrightarrow} \mathsf{DDH}$ and so we concentrate on proving the first two reductions.

Theorem 4. *Let $n \in \mathsf{poly}(\lambda)$. Then, $\forall MI_n(PKE) \overset{2}{\hookrightarrow} \forall MI_n(LTDF)$.*

Proof. We use the construction of Peikert et al. Let $(S_{inj}, S_{loss}, F, F^{-1})$ be a collection of (m, k)-lossy trapdoor functions. Let $\mathcal{H}$ be a family of pairwise independent hash functions from $\{0,1\}^m$ to $\{0,1\}^l$ for $l \leq k - 2\log(1/\epsilon)$ where $\epsilon = \mathtt{negl}(\lambda)$. The message space is $\{0,1\}^l$. Define the encryption scheme $\mathcal{E} = (\mathsf{Gen}; \mathsf{Enc}; \mathsf{Dec})$ where:

- $\mathsf{Gen}(1^\lambda)$ takes an injective trapdoor function $(s,t) \leftarrow_\$ S_{inj}$ and a hash function $h \leftarrow_\$ \mathcal{H}$. Then, it sets $pk = (s,h)$ and $sk = (t,h)$.
- $\mathsf{Enc}(pk, m)$ first generates random $x \leftarrow_\$ \{0,1\}^m$. Then, it sets $c_1 = F(s,x)$ and $c_2 = m \oplus h(x)$. It outputs $c = (c_1, c_2)$.
- $\mathsf{Dec}(sk, c)$ computes $x = F^{-1}(t, c_1)$ and returns $c_2 \oplus h(x)$.

Peikert et al. proved that $\mathcal{E}$ is an IND-CPA secure scheme. We show that if $(S_{inj}, S_{loss}, F, F^{-1})$ is a LTDF in the multi-instance setting then $\mathcal{E}$ is a IND-CPA tightly secure scheme in the multi-user setting. We do it using the same technique as Peikert et al. Consider the following variables:

– Variable X_0: choose $(s_1, t_1), \ldots, (s_n, t_n) \leftarrow_\$ S_{inj}, x_1, \ldots, x_n \leftarrow_\$ \{0,1\}^m$ and also $h_1, \ldots, h_n \in \mathcal{H}$. Then the value of X_0 is

$$(s_1, \ldots, s_n, h_1, \ldots, h_n, F(s_1, x_1), \ldots, F(s_n, x_n), h(x_1), \ldots, h(x_n)).$$

– Variable X_1: choose $(s_1, t_1), \ldots, (s_n, t_n) \leftarrow_\$ S_{loss}, x_1, \ldots, x_n \leftarrow_\$ \{0,1\}^m$ and also $h_1, \ldots, h_n \in \mathcal{H}$. Then the value of X_1 is

$$(s_1, \ldots, s_n, h_1, \ldots, h_n, F(s_1, x_1), \ldots, F(s_n, x_n), h(x_1), \ldots, h(x_n)).$$

– Variable X_2: choose $(s_1, t_1), \ldots, (s_n, t_n) \leftarrow_\$ S_{loss}, x_1, \ldots, x_n \leftarrow_\$ \{0,1\}^m$ and also $r_1, \ldots, r_n \in \{0,1\}^l$. Then the value of X_2 is

$$(s_1, \ldots, s_n, h_1, \ldots, h_n, F(s_1, x_1), \ldots, F(s_n, x_n), r_1, \ldots, r_n).$$

– Variable X_3: choose $(s_1, t_1), \ldots, (s_n, t_n) \leftarrow_\$ S_{inj}, x_1, \ldots, x_n \leftarrow_\$ \{0,1\}^m$ and also $r_1, \ldots, r_n \in \{0,1\}^l$. Then the value of X_3 is

$$(s_1, \ldots, s_n, h_1, \ldots, h_n, F(s_1, x_1), \ldots, F(s_n, x_n), r_1, \ldots, r_n).$$

Lemma 4 ([41], generalized). *Let X_0, X_1, X_2, X_3 be random variables defined as above. Then, $\{X_0\} \overset{c}{\approx} \{X_1\} \overset{s}{\approx} \{X_2\} \overset{c}{\approx} \{X_3\}$.*

Proof. Note that X_0 and X_1 are computationally indistinguishable because of the multi-setting indistinguishability property of LTDFs. Identical argument works for X_2 and X_3. In order to show that X_1 and X_2 are statistically indistinguishable we use the result by Peikert et al. ([41], Lemma 3.4) for the single-instance case. Then, by the standard hybrid argument we get $\Delta(X_1, X_2) \leq n \cdot \epsilon(\lambda)$ which is still negligible. □

One observes that, by Lemma 4, the encryption scheme $\mathcal{E}$ is indeed IND-CPA in the multi-user setting. Moreover, the reduction is still tight because only the statistical difference between X_1 and X_2 is dependent on the number of instances n. This completes the proof. □

In a similar way we can define the multi-instance All-But-One LTDFs and use them to construct IND-CCA secure scheme in the multi-user setting. One could take the construction provided by Peikert et al. and extend the proof of security to many instances. This approach would give us a reduction with security loss $O(q_{\mathsf{dec}})$, the same as in the single-instance case. However, we omit the formal proof here.

Let us consider the construction of a pseudorandom generator provided by Peikert et al., i.e. define $G(x) = (h_1(F(s, x)), h_2(x))$, for $h_1, h_2 \leftarrow_\$ \mathcal{H}$. By Lemma 3.4 in [41] and the Leftover Hash Lemma (e.g. [15]), if $(S_{inj}, S_{loss}, F, F^{-1})$ is a collection of lossy trapdoor functions then G is a pseudorandom generator. One observes that this result can be easily extended to the multi-instance setting thanks to Lemma 4. This example and Theorem 4 show that multi-instance

LTDFs can be useful in constructing more general primitives in the multi-instance setting. We now present how to build such primitives from a standard assumption, namely DDH.

Constructing Tightly Extensible LTDFs. We focus on proving the second reduction in (2) which involves encrypting matrices in a way similar to ElGamal encryption scheme. Suppose we work with a group G of prime order p and generator g. For simplicity, we write $[x] = g^x$ for $x \in \mathbb{Z}_p$. We write small bold letters (e.g. $\mathbf{x}$,$\mathbf{y}$) for column vectors and capital bold letters (e.g. $\mathbf{A}$,$\mathbf{U}$) for matrices. We denote $\mathbf{A}^t$ to be the transpose of $\mathbf{A}$. For simplicity, we write $[x] = g^x$ for $x \in \mathbb{Z}_p$ and similarly $[\mathbf{x}] = ([x_1], \ldots, [x_m])$ for $\mathbf{x} = (x_1, \ldots, x_m)$. We use identical notation $[\mathbf{A}]$ also for matrix $\mathbf{A}$.

For a matrix $\mathbf{A}$ with at least 2 rows, we write $\mathtt{WLR}(\mathbf{A})$ for a matrix $\mathbf{A}$ without last row. Similarly, we define $\mathtt{WLC}(\mathbf{A})$ for $\mathbf{A}$ without last column. We will use the following simple observation.

Observation 1 *Let $m, n, k \geq 2$ and $\boldsymbol{A}, \boldsymbol{B}$ be $m \times n$ and $n \times k$ matrices in G respectively. Then, $\mathtt{WLR}(\boldsymbol{AB}) = \mathtt{WLR}(\boldsymbol{A})\boldsymbol{B}$ and $\mathtt{WLC}(\boldsymbol{AB}) = \boldsymbol{A}\mathtt{WLC}(\boldsymbol{B})$*

We briefly recall the method for encrypting matrix $\mathbf{M} \in \mathbb{Z}_p^{m \times m}$ by Peikert et al. Firstly, we generate secret keys $\mathbf{z} = (z_1, \ldots, z_m) \in \mathbb{Z}_p^m$ and set $sk = \mathbf{z}, pk = [\mathbf{z}]$. Also, denote $h_i = [z_i]$. Then, choose uniformly random $r_1, \ldots, r_m \in \mathbb{Z}_p$. The encryption of $\mathbf{M}$ is a matrix $\mathbf{C} = (C_{i,j})$ where $C_{i,j} = ([r_i], [m_{i,j}][z_i]^{r_i})$. The construction of a LTDF $(S_{inj}, S_{loss}, F, F^{-1})$ from a DDH group looks as follows.

- S_{inj} first selects group parameters (G, p, g). Then, it returns $(\mathbf{C}, t)$ where $\mathbf{C}$ is a matrix encryption of the identity $\mathbf{I}$ and t consists of secret keys $\mathbf{z}$.
- S_{loss} selects group parameters (G, p, g) and returns $(\mathbf{C}, \perp)$ where $\mathbf{C}$ is a matrix encryption of zero matrix $\mathbf{0}$.
- F takes as input $(\mathbf{C}, \mathbf{x})$, where $\mathbf{C}$ is a function index and $\mathbf{x} \in \{0,1\}^m$, and returns $\mathbf{y} = \mathbf{x}\mathbf{C}$.
- F^{-1} takes as input $\mathbf{y} = ((y_{1,0}, y_{1,1}), \ldots, (y_{m,0}, y_{m,1}))$ and the trapdoor $\mathbf{z} = (z_1, \ldots, z_m)$. Then, it returns $\mathbf{x} = (x_1, \ldots, x_m)$ where $x_i = \log_g(y_{i,1}/y_{i,0}^{z_i})$ (it can be efficiently computed since x_i is a bit).

Security of this lossy trapdoor function relies heavily on the fact that the matrix encryption scheme described above gives indistinguishable ciphertexts if the DDH assumption holds [39,41]. The key observation is that if DDH is hard in G, then for randomly chosen $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_p^m$, $[\mathbf{x}\mathbf{y}^t]$ is indistinguishable from a uniformly random chosen matrix $\mathbf{U} \leftarrow_\$ G^{m \times m}$. We claim that this is also true for many instances, i.e. for randomly chosen $\mathbf{x}_i, \mathbf{y}_i \in \mathbb{Z}_p^m$ where $i = 1, \ldots, n$, $([\mathbf{x}_1\mathbf{y}_1^t], \ldots, [\mathbf{x}_n\mathbf{y}_n^t])$ is indistinguishable from a uniformly random chosen matrix $([\mathbf{U}_1], \ldots, [\mathbf{U}_n])$ where $U_i \leftarrow_\$ \mathbb{Z}_p^{m \times m}$. We write it formally as follows.

Theorem 5. *Let n be a polynomial in λ and $m \geq 2$. Define primitive $P_m = \langle \mathbb{P}, S_P, F_P R_P, \frac{1}{2} \rangle$ where:*

- $\mathbb{P} = (\mathbb{Z}_p^m \times \mathbb{Z}_p^m, \mathbb{Z}_p^{m \times m}, \mathbb{Z}_p^m \times \mathbb{Z}_p^m)$,
- *S_P sends group information (G, p, g) to R_P,*

- *F_P contains only a function f defined by $f(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{x}\boldsymbol{y}^t$,*
- *R_P first generates $\boldsymbol{x}, \boldsymbol{y} \leftarrow_\$ \mathbb{Z}_p^m$, calls f to get $\boldsymbol{x}\boldsymbol{y}^t$, samples $b \leftarrow_\$ \{0,1\}$ and sets $U = \boldsymbol{x}\boldsymbol{y}^t$ if $b = 0$ and $U \leftarrow_\$ \mathbb{Z}_p^{m\times m}$ if $b = 1$. Finally, it sends $([\boldsymbol{x}], [\boldsymbol{y}], [\boldsymbol{U}])$ along with (G, p, g) to adversary $\mathcal{A}$. Security game R_P returns 1 if $\mathcal{A}$ guesses the bit b.*

Then, $\forall \mathsf{MI}_n(P_m) \overset{1}{\hookrightarrow} P_m$.

Proof. Assume first that there exists a PPT algorithm $\mathcal{C}_1$, which given a triple $([\mathbf{x}], [\mathbf{y}], [\mathbf{U}])$ sent by R_P, returns another triple $([\mathbf{x}'], [\mathbf{y}'], [\mathbf{U}'])$ such that: (i) $\mathbf{y}' = \mathbf{y}$, (ii) $\mathbf{x}'$ is uniformly random, (iii) if $b = 0$ then $\mathbf{U}' = \mathbf{x}'\mathbf{y}'^t$ and if $b = 1$ then $\mathbf{U}'$ is uniformly random with probability $1 - \mathtt{negl}(\lambda)$. In a similar fashion we can define a PPT algorithm $\mathcal{C}_2$ which does the same thing as $\mathcal{C}_1$ but it fixes $\mathbf{x}$ instead of $\mathbf{y}$. Note that if $\mathcal{C}_1$ exists then clearly $\mathcal{C}_2$ also exists.

Now, suppose there exists an adversary $\mathcal{A}$ which $\forall \mathsf{MI}_n(P_m)-$breaks f. We construct an adversary $S^{\mathcal{A}}$ which P_m-breaks f as follows. Given a triple $\mathbf{v} = ([\mathbf{x}], [\mathbf{y}], [\mathbf{U}])$ from R_P, it runs n independent copies of $\mathcal{C}_1$ on input $\mathbf{v}$ and gets back outputs $\mathbf{v}_1, \ldots, \mathbf{v}_n$. Next, it runs n independent copies of $\mathcal{C}_2$ where the i-th copy of $\mathcal{C}_2$ gets as input $\mathbf{v}_i$. Then, collect the outputs $\mathbf{w}_1, \ldots, \mathbf{w}_n$ and pass them to $\mathcal{A}$. Eventually, when $\mathcal{A}$ returns a bit b', output b'. Note that this reduction is tight by the property (iii) of $\mathcal{C}_1$ and by standard hybrid argument. Therefore, what we have left is to construct an algorithm $\mathcal{C}_1$.

$\mathcal{C}_1(1^\lambda, G, p, g, [\mathbf{x}], [\mathbf{y}], [\mathbf{U}])$	
1 :	$\mathbf{R} \leftarrow_\$ \mathbb{Z}_p^{m\times m}$
2 :	$\tilde{\mathbf{r}} \leftarrow_\$ \mathbb{Z}_p^m$
3 :	$[\mathbf{x}'] = [\mathbf{R}\mathbf{x} + \tilde{\mathbf{r}}]$
4 :	$[\mathbf{U}'] = [\mathbf{R}\mathbf{U} + \tilde{\mathbf{r}}\mathbf{y}^t]$
5 :	**return** $([\mathbf{x}'], [\mathbf{y}], [\mathbf{U}'])$

Fig. 6. PPT algorithm for $\mathcal{C}_1$.

Consider the following algorithm for $\mathcal{C}_1$ in Fig. 3. Note that we are able to compute $[\mathbf{x}']$ and $[\mathbf{U}']$ in lines 3 and 4 even though we do not know values for $\mathbf{x}', \mathbf{U}'$. Clearly, property (i) is satisfied. Also, $\mathbf{x}'$ is uniformly random because of the randomness of $\tilde{\mathbf{r}}$. The most challenging part is to show (iii).

First, suppose that $b = 0$. So we have $\mathbf{U} = \mathbf{x}\mathbf{y}^t$. Hence, $\mathbf{U}' = \mathbf{R}\mathbf{U} + \tilde{\mathbf{r}}\mathbf{y}^t = \mathbf{R}\mathbf{x}\mathbf{y}^t + \tilde{\mathbf{r}}\mathbf{y}^t = (\mathbf{R}\mathbf{x} + \tilde{\mathbf{r}})\mathbf{y}^t = \mathbf{x}'\mathbf{y}^t$. Now, consider the case $b = 1$. Then, $\mathbf{U}$ is a uniformly random matrix. We want to show that $\mathbf{U}'$ is also a uniformly random matrix with overwhelming probability. Denote $\mathbf{x} = (x_1, \ldots, x_m)$ and assume that $x_m \neq 0$ (this occurs with an overwhelming probability). We slightly change the algorithm for $\mathcal{C}_1$: we first choose random $\mathbf{x}' = (x'_1, \ldots, x'_m), \tilde{\mathbf{r}} = (\tilde{r}_1, \ldots, \tilde{r}_m)$ from $\mathbb{Z}_p^m$ and $\tilde{\mathbf{R}} = (\tilde{R}_{i,j}) \in \mathbb{Z}_p^{m\times(m-1)}$. Next, we set $\mathbf{r} = (r_1, \ldots, r_m)$, where

$r_i = {x_m}^{-1}(x'_i - \tilde{r}_i - \sum_{j=1}^{m-1} x_j \tilde{R}_{i,j})$, and $\mathbf{R} = \left(\tilde{\mathbf{R}} \,\middle|\, \mathbf{r}\right)$. Note that this change does not affect the input/output behaviour of $\mathcal{C}_1$. Moreover, we can rewrite $\mathbf{R}$ as:

$$\mathbf{R} = \begin{pmatrix} \tilde{r}_1 & \tilde{R}_{1,1} & \tilde{R}_{1,2} & \cdots & \tilde{R}_{1,m-1} & x'_1 \\ \tilde{r}_2 & \tilde{R}_{2,1} & \tilde{R}_{2,2} & \cdots & \tilde{R}_{2,m-1} & x'_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \\ \tilde{r}_m & \tilde{R}_{m,1} & \tilde{R}_{m,2} & \cdots & \tilde{R}_{m,m-1} & x'_m \end{pmatrix} \begin{pmatrix} 0 & 0 & \cdots & 0 & -x_m^{-1} \\ 1 & 0 & \cdots & 0 & -x_1 x_m^{-1} \\ 0 & 1 & \cdots & 0 & -x_2 x_m^{-1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & -x_{m-1} x_m^{-1} \\ 0 & 0 & \cdots & 0 & x_m^{-1} \end{pmatrix} \tag{10}$$

Denote $\bar{\mathbf{R}}$ and $\mathbf{M}$ as the left-hand side and the right-hand side matrices respectively. Also, define $\mathbf{A} = \mathbf{M}\mathbf{U}$ and $\hat{\mathbf{R}}$ such that $\bar{\mathbf{R}} = \left(\hat{\mathbf{R}} \,\middle|\, \mathbf{x}'\right)$. Note that the first column of $\mathbf{M}$ is the (additive) inverse of the last column of $\mathbf{M}$. Consequently, we get the same property in $\mathbf{A}$. Moreover, $\mathtt{WLR}(\mathbf{M})$ is clearly invertible and thus $\mathbf{U}$ being uniformly random matrix implies that $\mathtt{WLR}(\mathbf{A}) = \mathtt{WLR}(\mathbf{M})\mathbf{U}$ is also uniformly random. For simplicity, let us denote $\mathbf{a}$ to be the last row of $\mathbf{A}$, $\mathbf{A}_1 = \mathtt{WLR}(\mathbf{A})$ and $\mathbf{A}_2$ be the matrix $\mathbf{A}_1$ without the first row (which is the inverse of the last row of $\mathbf{A}$). Then, by the observations above we can expand $\mathbf{U}'$ as follows:

$$\begin{aligned} \mathbf{U}' &= \mathbf{R}\mathbf{U} + \tilde{\mathbf{r}}\mathbf{y}^t \\ &= \left(\left(\hat{\mathbf{R}} \,\middle|\, \mathbf{0}\right) + \left(\mathbf{0} \,\middle|\, \mathbf{x}'\right)\right)\mathbf{A} + \tilde{\mathbf{r}}\mathbf{y}^t \\ &= \hat{\mathbf{R}}\mathbf{A}_1 + \mathbf{x}'\mathbf{a}^t + \tilde{\mathbf{r}}\mathbf{y}^t \\ &= \left(\left(\tilde{\mathbf{r}} \,\middle|\, \mathbf{0}\right) + \left(\mathbf{0} \,\middle|\, \tilde{\mathbf{R}}\right)\right)\mathbf{A}_1 + \mathbf{x}'\mathbf{a}^t + \tilde{\mathbf{r}}\mathbf{y}^t \\ &= \tilde{\mathbf{r}}(\mathbf{y}^t - \mathbf{a}^t) + \tilde{\mathbf{R}}\mathbf{A}_2 + \mathbf{x}'\mathbf{a}^t. \end{aligned} \tag{11}$$

This is equivalent to ${\mathbf{U}'}^t = \left(\mathbf{y} - \mathbf{a} \,\middle|\, \mathbf{A}_2^t\right)\left(\tilde{\mathbf{r}} \,\middle|\, \tilde{\mathbf{R}}\right)^t + \mathbf{a}{\mathbf{x}'}^t$. Note that $\left(\mathbf{y} - \mathbf{a} \,\middle|\, \mathbf{A}_2^t\right)$ is with high probability an invertible matrix because $\mathbf{A}_1$ is uniformly random. Moreover, we chose $\left(\tilde{\mathbf{r}} \,\middle|\, \tilde{\mathbf{R}}\right)$ uniformly at random and therefore $\mathbf{U}'$ is also uniformly random (with probability $1 - \mathsf{negl}(\lambda)$). □

Similarly as in [41], we obtain $\forall\mathsf{MI}_n(\mathsf{LTDF}) \overset{1}{\hookrightarrow} \forall\mathsf{MI}_n(P_m)$. Hence, $\forall\mathsf{MI}_n(\mathsf{LTDF}) \overset{1}{\hookrightarrow}$ LTDF/DDH follows from the Theorem 5 applied to the construction by Peikert et al. along with the random self-reducibility of DDH. Thus, combining (9) with Lemma 2 we obtain the following.

Theorem 6. *LTDF is* $(\lambda, \forall)$*-tightly extensible w.r.t. DDH and PKE is* $(2\lambda, \forall)$*-tightly extensible w.r.t. DDH.*

All in all, we have provided a new way of constructing multi-user IND-CPA encryption schemes out of a DDH group using tightly extensible lossy trapdoor functions. We leave it as an open question whether it is possible to obtain tightly extensible LTDFs from different standard assumptions, such as lattices.

References

1. Masayuki Abe, Bernardo David, Markulf Kohlweiss, Ryo Nishimaki, and Miyako Ohkubo. Tagged one-time signatures: Tight security and optimal tag size. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 312–331. Springer, Heidelberg, February / March 2013.
2. Masayuki Abe, Dennis Hofheinz, Ryo Nishimaki, Miyako Ohkubo, and Jiaxin Pan. Compact structure-preserving signatures with almost tight security. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part II*, volume 10402 of *LNCS*, pages 548–580. Springer, Heidelberg, August 2017.
3. Nuttapong Attrapadung, Goichiro Hanaoka, and Shota Yamada. A framework for identity-based encryption with almost tight security. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part I*, volume 9452 of *LNCS*, pages 521–549. Springer, Heidelberg, November / December 2015.
4. Benedikt Auerbach, David Cash, Manuel Fersch, and Eike Kiltz. Memory-tight reductions. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 101–132. Springer, Heidelberg, August 2017.
5. Paul Baecher, Christina Brzuska, and Marc Fischlin. Notions of black-box reductions, revisited. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part I*, volume 8269 of *LNCS*, pages 296–315. Springer, Heidelberg, December 2013.
6. Mihir Bellare, Alexandra Boldyreva, and Silvio Micali. Public-key encryption in a multi-user setting: Security proofs and improvements. In Bart Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 259–274. Springer, Heidelberg, May 2000.
7. Olivier Blazy, Eike Kiltz, and Jiaxin Pan. (Hierarchical) identity-based encryption from affine message authentication. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 408–425. Springer, Heidelberg, August 2014.
8. Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudo random bits. In *23rd FOCS*, pages 112–117. IEEE Computer Society Press, November 1982.
9. Xavier Boyen, Qixiang Mei, and Brent Waters. Direct chosen ciphertext security from identity-based techniques. In Vijayalakshmi Atluri, Catherine Meadows, and Ari Juels, editors, *ACM CCS 05*, pages 320–329. ACM Press, November 2005.
10. Jie Chen, Junqing Gong, and Jian Weng. Tightly secure IBE under constant-size master public key. In Serge Fehr, editor, *PKC 2017, Part I*, volume 10174 of *LNCS*, pages 207–231. Springer, Heidelberg, March 2017.
11. Jie Chen and Hoeteck Wee. Fully, (almost) tightly secure IBE and dual system groups. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 435–460. Springer, Heidelberg, August 2013.
12. Ivan Damgård. Collision free hash functions and public key signature schemes. In David Chaum and Wyn L. Price, editors, *EUROCRYPT'87*, volume 304 of *LNCS*, pages 203–216. Springer, Heidelberg, April 1988.
13. Ivan Damgård. On the randomness of legendre and jacobi sequences. In Shafi Goldwasser, editor, *CRYPTO'88*, volume 403 of *LNCS*, pages 163–172. Springer, Heidelberg, August 1990.
14. Yevgeniy Dodis, Roberto Oliveira, and Krzysztof Pietrzak. On the generic insecurity of the full domain hash. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 449–466. Springer, Heidelberg, August 2005.

15. Yevgeniy Dodis, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 523–540. Springer, Heidelberg, May 2004.
16. Marc Fischlin, Anja Lehmann, Thomas Ristenpart, Thomas Shrimpton, Martijn Stam, and Stefano Tessaro. Random oracles with(out) programmability. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 303–320. Springer, Heidelberg, December 2010.
17. Romain Gay, Dennis Hofheinz, Eike Kiltz, and Hoeteck Wee. Tightly CCA-secure encryption without pairings. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part I*, volume 9665 of *LNCS*, pages 1–27. Springer, Heidelberg, May 2016.
18. Romain Gay, Dennis Hofheinz, and Lisa Kohl. Kurosawa-desmedt meets tight security. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part III*, volume 10403 of *LNCS*, pages 133–160. Springer, Heidelberg, August 2017.
19. Romain Gay, Dennis Hofheinz, Lisa Kohl, and Jiaxin Pan. More efficient (almost) tightly secure structure-preserving signatures. LNCS, pages 230–258. Springer, Heidelberg, 2018.
20. Oded Goldreich. *Foundations of Cryptography: Basic Applications*, volume 2. Cambridge University Press, Cambridge, UK, 2004.
21. Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In *21st ACM STOC*, pages 25–32. ACM Press, May 1989.
22. Junqing Gong, Jie Chen, Xiaolei Dong, Zhenfu Cao, and Shaohua Tang. Extended nested dual system groups, revisited. In Chen-Mou Cheng, Kai-Min Chung, Giuseppe Persiano, and Bo-Yin Yang, editors, *PKC 2016, Part I*, volume 9614 of *LNCS*, pages 133–163. Springer, Heidelberg, March 2016.
23. Iftach Haitner, Thomas Holenstein, Omer Reingold, Salil P. Vadhan, and Hoeteck Wee. Universal one-way hash functions via inaccessible entropy. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 616–637. Springer, Heidelberg, May 2010.
24. Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999.
25. Dennis Hofheinz. Algebraic partitioning: Fully compact and (almost) tightly secure cryptography. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A, Part I*, volume 9562 of *LNCS*, pages 251–281. Springer, Heidelberg, January 2016.
26. Dennis Hofheinz. Adaptive partitioning. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part III*, volume 10212 of *LNCS*, pages 489–518. Springer, Heidelberg, May 2017.
27. Dennis Hofheinz and Tibor Jager. Tightly secure signatures and public-key encryption. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 590–607. Springer, Heidelberg, August 2012.
28. Dennis Hofheinz, Jessica Koch, and Christoph Striecks. Identity-based encryption with (almost) tight security in the multi-instance, multi-ciphertext setting. In Jonathan Katz, editor, *PKC 2015*, volume 9020 of *LNCS*, pages 799–822. Springer, Heidelberg, March / April 2015.
29. Thomas Holenstein. Pseudorandom generators from one-way functions: A simple construction for any hardness. In Shai Halevi and Tal Rabin, editors, *TCC 2006*, volume 3876 of *LNCS*, pages 443–461. Springer, Heidelberg, March 2006.
30. Russell Impagliazzo and Steven Rudich. Limits on the provable consequences of one-way permutations. In *21st ACM STOC*, pages 44–61. ACM Press, May 1989.

31. Russell Impagliazzo and Steven Rudich. Limits on the provable consequences of one-way permutations. In Shafi Goldwasser, editor, *CRYPTO'88*, volume 403 of *LNCS*, pages 8–26. Springer, Heidelberg, August 1990.
32. Jonathan Katz and Chiu-Yuen Koo. On constructing universal one-way hash functions from arbitrary one-way functions. Cryptology ePrint Archive, Report 2005/328, 2005. `http://eprint.iacr.org/2005/328`.
33. Leslie Lamport. Constructing digital signatures from a one-way function. Technical Report SRI-CSL-98, SRI International Computer Science Laboratory, October 1979.
34. Benoît Libert, Marc Joye, Moti Yung, and Thomas Peters. Concise multi-challenge CCA-secure encryption and signatures with almost tight security. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part II*, volume 8874 of *LNCS*, pages 1–21. Springer, Heidelberg, December 2014.
35. Benoît Libert, Thomas Peters, Marc Joye, and Moti Yung. Compactly hiding linear spans - tightly secure constant-size simulation-sound QA-NIZK proofs and applications. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part I*, volume 9452 of *LNCS*, pages 681–707. Springer, Heidelberg, November / December 2015.
36. Christian Mauduit and András Sárközy. On finite pseudorandom binary sequences i: Measure of pseudorandomness, the legendre symbol. *Acta Arithmetica*, 82(4):365–377, 1997.
37. Ralph C. Merkle. A digital signature based on a conventional encryption function. In Carl Pomerance, editor, *CRYPTO'87*, volume 293 of *LNCS*, pages 369–378. Springer, Heidelberg, August 1988.
38. Moni Naor. On cryptographic assumptions and challenges (invited talk). In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 96–109. Springer, Heidelberg, August 2003.
39. Moni Naor and Omer Reingold. Synthesizers and their application to the parallel construction of psuedo-random functions. In *36th Annual Symposium on Foundations of Computer Science, Milwaukee, Wisconsin, 23-25 October 1995*, pages 170–181, 1995.
40. Omkant Pandey, Rafael Pass, and Vinod Vaikuntanathan. Adaptive one-way functions and applications. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 57–74. Springer, Heidelberg, August 2008.
41. Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. Cryptology ePrint Archive, Report 2007/279, 2007. `http://eprint.iacr.org/2007/279`.
42. Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 187–196. ACM Press, May 2008.
43. Omer Reingold, Luca Trevisan, and Salil P. Vadhan. Notions of reducibility between cryptographic primitives. In Moni Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 1–20. Springer, Heidelberg, February 2004.
44. John Rompel. One-way functions are necessary and sufficient for secure signatures. In *22nd ACM STOC*, pages 387–394. ACM Press, May 1990.
45. Daniel R. Simon. Finding collisions on a one-way street: Can secure hash functions be based on general assumptions? In Kaisa Nyberg, editor, *EUROCRYPT'98*, volume 1403 of *LNCS*, pages 334–345. Springer, Heidelberg, May / June 1998.
46. Wim van Dam, Sean Hallgren, and Lawrence Ip. Quantum algorithms for some hidden shift problems. In *14th SODA*, pages 489–498. ACM-SIAM, January 2003.