Separation Results on the "One-More" Computational Problems*

Emmanuel Bresson[1], Jean Monnerat**[2], and Damien Vergnaud[3]

[1] DCSSI Crypto Lab, Paris, France
[2] Department of Computer Science & Engineering, University of California San Diego, USA
[3] École Normale Supérieure – C.N.R.S. – I.N.R.I.A.
45 rue d'Ulm, 75230 Paris CEDEX 05, France

Abstract. In 2001, Bellare, Namprempre, Pointcheval and Semanko introduced the notion of *"one-more" computational problems*. Since their introduction, these problems have found numerous applications in cryptography. For instance, Bellare *et al.* showed how they lead to a proof of security for Chaum's RSA-based blind signature scheme in the random oracle model.
In this paper, we provide separation results for the computational hierarchy of a large class of algebraic "one-more" computational problems (*e.g.* the one-more discrete logarithm problem, the one-more RSA problem and the one-more static Computational Diffie-Hellman problem in a bilinear setting). We also give some cryptographic implications of these results and, in particular, we prove that it is very unlikely, that one will ever be able to prove the unforgeability of Chaum's RSA-based blind signature scheme under the sole RSA assumption.

Keywords. "One-more" problems, Black-box reductions, Random self-reducible problems, Algebraic algorithms.

1 Introduction

Background. In cryptography, a one-way function f is a function that can be computed by some algorithm in polynomial time (with respect to the input size) but such that no probabilistic polynomial-time algorithm can compute a preimage of $f(x)$ with a non-negligible probability, when x is chosen uniformly at random in the domain of f. At the very beginning of the century, it has been observed that there seems little hope of proving the security of many cryptographic constructions based only on the "standard" one-wayness assumption of the used primitive. The security of some schemes seems to rely on different, and probably stronger, properties of the underlying one-way function. Cryptographers have therefore suggested that one should formulate explicit new computational problems to prove the security of these protocols. For instance, Okamoto and Pointcheval [14] introduced in 2001 a novel class of computational problems, the *gap problems*, which find a nice and rich practical instantiation with the Diffie-Hellman problems. They used the gap Diffie-Hellman problem for solving a more than 10-year old open security problem: the unforgeability of Chaum-van Antwerpen undeniable signature scheme [11].

In 2001, Bellare, Namprempre, Pointcheval and Semanko [2] introduced the notion of *one-more one-way function*. A function is one-more one-way if it can be computed by some algorithm in polynomial time (in the input size) but for which there exists no probabilistic polynomial-time algorithm $\mathcal{A}$ with non-negligible probability to win the following game:

- $\mathcal{A}$ gets the description of f as input and has access to two oracles;
- an *inversion* oracle that given y in f's codomain returns x in f's domain such that $f(x) = y$;
- a *challenge* oracle that, each time it is invoked (it takes no inputs), returns a random challenge point from f's codomain;

* This paper has appeared in *Topics in cryptology - CT-RSA 2008*, Lect. Notes Comput. Sci., T. Malkin ed., Springer, Lect. Notes Comput. Sci. vol. 4964, 2008, p. 71-87.
** Supported by a fellowship of the Swiss National Science Foundation, PBEL2–116915

- $\mathcal{A}$ wins the game if it succeeds in inverting all n points output by the challenge oracle using strictly less than n queries to the inversion oracle.

Bellare *et al.* showed how these problems lead to a proof of security for Chaum's RSA-based blind signature scheme [10] in the random oracle model.

The approach consisting in introducing new computational problems to study the security of cryptosystems is not completely satisfactory since the proof of security often relies on an extremely strong assumption which is hard to validate. Nevertheless, it is better to have such a security argument than nothing since as mentioned in [2]: "*These problems can then be studied, to see how they relate to other problems and to what extent we can believe in them as assumptions.*" The purpose of this paper is to study the hierarchy of the computational difficulty of the "one-more" problems of Bellare *et al.* and its cryptographic implications. In particular, we prove that it is very unlikely, that one will ever be able to prove the unforgeability of Chaum's RSA-based blind signature scheme under the sole RSA assumption.

Related work. Since the one-more-inversion problems were introduced in [2], they have found numerous other applications in cryptography.

- Bellare and Palacio [4] proved in 2002 that Guillou-Quisquater and Schnorr identification schemes [12,17] are secure against impersonation under active (and concurrent) attack under the assumption that the *one-more* RSA *problem* and the *one-more discrete logarithm problem* are intractable (respectively).
- Bellare and Neven [3] proved the security of an RSA based transitive signature scheme suggested by Micali and Rivest in 2002 [13] under the assumption of the hardness of the one-more RSA problem.
- Bellare and Sandhu had used the same problem to prove the security of some two-party RSA-based signature protocols [5].
- In [6], Boldyreva proposed a new blind signature scheme – based on Boneh-Lynn-Shacham signature [7] – which is very similar to the RSA blind signature protocol. She introduced a new computational problem: the *one-more static Computational Diffie-Hellman problem* (see also [9]) and proved the security (in the random oracle model) of her scheme assuming the intractability of this problem.
- Paillier and Vergnaud [15] provided evidence that the security of Schnorr signatures [17] cannot be equivalent to the discrete log problem in the standard model. They proposed a method of converting a reduction of the unforgeability of this signature scheme to the discrete logarithm problem into an algorithm solving the one-more discrete log problem. Their technique applies whenever the reduction belongs to a certain "natural" class of reductions that they refer to as *algebraic reductions.*

Contributions of the paper. Following the approach from [15], we give arguments showing that, for any integer $n > 1$, solving the one-more problem with access to the inversion oracle up to n times cannot be reduced to the resolution of this problem with access to this oracle limited to $n + 1$ queries. Our results apply to the class of black-box reductions and are extended in the case of the one-more discrete logarithm problems to a class of algebraic black-box reductions.

These separation results apply to many computational problems used in the cryptographic literature, like the one-more RSA problem and the one-more static Diffie-Hellman problem in a bilinear setting. Due to the equivalence of the unforgeability of Chaum and Boldyreva blind signatures [10,6] and the intractability of the one-more RSA problem and the one-more static Diffie-Hellman problem in a bilinear setting, our results imply that it is very unlikely, that one will ever be able to prove the unforgeability of these schemes under the sole assumption of the one-wayness of their respective underlying primitive.

We stress that our work sheds more light on the computational complexity of these problems but does not explicitly lead to actual way to solve them. Finally, we mention that Brown [8] *independently* found similar separation results[4].

2 Preliminaries

NOTATIONS. Taking an element x uniformly at random from a set X will be denoted $x \leftarrow_U X$. Assigning a value a to a variable x is denoted by $x \leftarrow a$. Algorithms are modeled by probabilistic Turing machines and are usually considered polynomial-time. The term "efficient" will refer to polynomial-time. We write $\mathcal{A}(\star; \varpi)$ the output of algorithm $\mathcal{A}$ when running on input $\star$ and using random ϖ. With $\mathcal{A}(\star)$, we mean the random variable resulting from $\mathcal{A}(\star; \varpi)$ by choosing ϖ uniformly at random. For any algorithm $\mathcal{A}$, $T(\mathcal{A})$ denotes its running time. An algorithm $\mathcal{A}$ with a black-box oracle access to an algorithm $\mathcal{B}$ is denoted $\mathcal{A}^{\mathcal{B}}$.

BLACK-BOX REDUCTIONS. An algorithm $\mathcal{R}$ is said to be a black-box reduction from a problem P_2 to a problem P_1 if for any algorithm $\mathcal{A}$ solving P_1, algorithm $\mathcal{R}^{\mathcal{A}}$ solves P_2 thanks to a black-box access to $\mathcal{A}$. Below, we provide more details about our black-box model. Namely, we describe what we mean by a "black-box access" and give a characterization of the classes of algorithms $\mathcal{A}$ we will consider. In other words, we specify which algorithms are transformed by $\mathcal{R}$ and how $\mathcal{R}$ can interact with them.

BLACK-BOX ACCESS. A black-box access essentially means that $\mathcal{R}$ is allowed to use $\mathcal{A}$ as a subroutine without taking advantage of its internal structure (code). $\mathcal{R}$ can only provide the inputs to $\mathcal{A}$ and observe the resulting outputs. If $\mathcal{A}$ has access to an oracle, the corresponding queries must be answered by $\mathcal{R}$. In other words, the reduction should simulate $\mathcal{A}$'s environment through its input-output interface.

When $\mathcal{A}$ is probabilistic, a new black-box access by $\mathcal{R}$ results in a new execution of $\mathcal{A}$ with fresh random coins. In this paper, we do not consider the random tape of $\mathcal{A}$ to be seen by $\mathcal{R}$. This is in accordance with the work by Barak [1] saying that the knowledge of such randomness can hardly help a black-box reduction.

As usually in the literature, we allow $\mathcal{R}$ to rewind $\mathcal{A}$ with a previously used random tape. Our approach is formalized by restricting the reduction $\mathcal{R}$ to sequentially execute some of the following operations when interacting with $\mathcal{A}$:

- **Launch**. Any previously launched execution of $\mathcal{A}$ is aborted. $\mathcal{R}$ launches a new execution of $\mathcal{A}$ with a fresh random tape ϖ on an input of its choice.
- **Rewind**. Any previously launched execution of $\mathcal{A}$ is aborted. $\mathcal{R}$ restarts $\mathcal{A}$ with a previously used random tape and an input of its choice.
- **Stop.** $\mathcal{R}$ definitely stops the interaction with $\mathcal{A}$.

We assume that all executions with fresh random tapes are uniquely identified so that $\mathcal{R}$ can make some "**Rewind**" without explicitly knowing these random tapes. Note that a call of any above procedure is counted as a single time unit in the complexity of $\mathcal{R}$.

For some results, we will need to consider a weaker model obtained by relaxing the "**Rewind**" queries made by $\mathcal{R}$. Instead, we only tolerate a kind of weak rewinding of $\mathcal{A}$ with the same random tape and its corresponding input. So, in this weaker model, we replace the "**Rewind**" queries by the following one:

[4] His paper appeared on the IACR eprint, while our paper was already under submission. His work is based on the very same core idea but does not explicitly handle the case where reductions make use of rewinding techniques.

– **Relaunch**. Any previously launched execution of $\mathcal{A}$ is aborted. $\mathcal{R}$ restarts $\mathcal{A}$ with a previously used random tape *and* the corresponding input.

Hence, rewinding techniques which restart $\mathcal{A}$ on the same random tape and a *different* input are not allowed in this model. As a consequence, reductions involving *"forking-Lemma"*-like [16] techniques are not considered. We however point out that a "**Relaunch**" query can be useful to $\mathcal{R}$ when $\mathcal{A}$ has access to some oracles. Namely, $\mathcal{R}$ may differently simulate the oracle outputs from an execution to another one in order to gain some information to solve its challlenge.

Classes of Algorithms. For any τ and ε non-negative functions defined on $\mathbf{N}$ and a computational problem P with associated security parameter $k \in \mathbf{N}$, an algorithm $\mathcal{A}$ is said to be an (ε, τ)-P solver if it succeeds in solving P (fed with k) with probability at least $\varepsilon(k)$ and at most time complexity $\tau(k)$ for any $k \in \mathbf{N}$, where the probability is over random tapes of all involved algorithms. We denote by $\mathcal{CL}(P, \varepsilon, \tau)$ the class of such probabilistic algorithms. We say that $\mathcal{R}$ is an $(\varepsilon_1, \tau_1, \varepsilon_2, \tau_r)$-reduction from P_2 to P_1 if it transforms any algorithm in $\mathcal{CL}(P_1, \varepsilon_1, \tau_1)$ into a P_2-solver with success probability greater or equal to ε_2 and the running time of $\mathcal{R}$ is less or equal to τ_r. Usually, black-box reductions transform any adversary with a given success probability without any consideration of the time complexity of this one. In this case, we have $\tau_1(k) = +\infty$ for any $k \in \mathbf{N}$ and use the term of $(\varepsilon_1, \varepsilon_2, \tau_r)$-reduction from P_2 to P_1 reduction. We call such reductions "classical" while those transforming only bounded adversaries are called "sophisticated" reductions. As far as we know, we are not aware of the existence in the literature of "sophisticated" reductions which are not implicitly classical, i.e., which do not succeed in transforming adversaries with a greater complexity than the given τ_1.

Black-Box Separations. A black-box reduction $\mathcal{R}$ from P_2 to P_1 can just be seen as an oracle Turing machine solving the problem P_2. Thus it can be transformed through a so-called *meta-reduction* to solve another problem (say, P_3). When the latter is assumed to be hard, an efficient meta-reduction rules out the existence of $\mathcal{R}$, hence proving a separation between problems P_1 and P_2. In other words, it proves that P_2 is strictly harder than P_1, conditioned by the hardness of P_3.

More formally, the construction is as follows. We start from the reduction $\mathcal{R}$ from P_2 to P_1. Our goal is to specify an algorithm $\mathcal{M}$ that solves the problem P_3, having a black-box access to $\mathcal{R}$. The algorithm $\mathcal{M}$ needs to simulate the environment of $\mathcal{R}$, *i.e.*, all its oracles, and in particular, the correct behavior of the P_1-solver. In the present work, such *correct behavior* (viewed from $\mathcal{R}$) is formalized by assuming that the P_1-solver belongs to a certain class $\mathcal{CL}(P_1, \varepsilon, \tau)$ for some given ε and τ. However, in the classical case, the reduction is (formally) able to transform any P_1-solver[5] whatever its running time is ($\tau = +\infty$).

In this work, we will focus on "classical" reductions but also show that our results hold for "sophisticated" reductions in the weaker model, where "**Rewind**" queries are replaced by the "**Relaunch**" ones. In this case, we confine the P_1 to a given finite τ, thus to a smaller class $\mathcal{CL}(P_1, \varepsilon, \tau)$. Hence, by restricting the class of solvers that the reduction $\mathcal{R}$ is able to deal with, we enlarge the class of such reductions. In fact, the smaller τ is, the bigger the number of possible reductions do exist. Thus, excluding the existence of such reductions using a meta-reduction leads to a separation result which is at least as strong as the case where $\tau = +\infty$.

We may wonder whether this is *strictly* stronger to do so. Usually the reduction should not care about the complexity of the P_1-solver which is treated as a black-box and for which an access is counted as a single unit anyway. However, when the P_1-problem is specified with some oracle access to a solver (as for the "one-more" problems), one must be more careful. The main reason is that there may be a correlation between the "external behavior" of a P_1-solver

[5] Of course, for a cryptographic result to make sense in practice, the solver is restricted to polynomial-time.

and its complexity. What we call the "external behavior" of an algorithm corresponds to the distribution of both the intermediate outputs (queries) sent to the oracles and the final output. As a result, a reduction may be able to only transform the P_1-solvers with a specific "external behavior", guaranteed by the bound τ. For instance, a one-more discrete logarithm solver which would never query the discrete logarithm oracle must be a discrete logarithm solver. Assuming that no discrete logarithm solver exists with a time less than τ, a sophisticated reduction knows that it does not need to transform such an n-DL solver.

When devising the meta-reduction $\mathcal{M}$, this difficulty is induced in the simulation of the P_1-solver. To be more explicit, the meta-reduction needs to simulate a P_1-solver with the correct "external behavior", since this one may need to comply to a specific form when τ is finite. In our results, the class $\mathcal{CL}(P_1, \varepsilon, \tau)$ will be chosen so that $\mathcal{M}$ will simulate the correct behavior of a solver of this class.

3 Random Self-Reducible Problems

3.1 Definitions

Let $\mathsf{P} = (\mathsf{PG}, \mathsf{IG})$ be a generic computational problem, where PG is a parameter generator and IG is an instance generator[6]. Given a security parameter $k \in \mathbf{N}$, the probabilistic polynomial-time algorithm $\mathsf{PG}(1^k)$ generates some parameters param. These parameters notably characterize an instance set $\mathbf{I}$ and a solution set $\mathbf{S}$. The instance generator IG takes param as input and outputs an instance $\mathtt{ins} \in \mathbf{I}$.

We assume that there exists an efficient verification algorithm $\mathsf{V}(\mathsf{param}, \mathtt{ins}, \mathtt{sol})$ that, for any $(\mathtt{ins}, \mathtt{sol}) \in \mathbf{I} \times \mathbf{S}$, outputs 1 if $\mathtt{sol}$ is a solution to $\mathtt{ins}$ (with respect to param) and 0 otherwise. For an algorithm $\mathcal{A}$, we consider the following experiment:

Experiment $\mathbf{Exp}^{\mathsf{P}}_{\mathcal{A}}(k)$.
  $\mathsf{param} \leftarrow \mathsf{PG}(1^k)$
  $\mathtt{ins} \leftarrow \mathsf{IG}(\mathsf{param})$
  $\mathtt{sol} \leftarrow \mathcal{A}(\mathsf{param}, \mathtt{ins})$
  Output $\mathsf{V}(\mathsf{param}, \mathtt{ins}, \mathtt{sol})$

The success probability of $\mathcal{A}$ is $\mathsf{Succ}^{\mathsf{P}}_{\mathcal{A}}(k) = \Pr[\mathbf{Exp}^{\mathsf{P}}_{\mathcal{A}}(k) = 1]$, where the probability is taken over the random coins of all algorithms PG, IG and $\mathcal{A}$.

We introduce the "one-more" variants of the problem P. Let n be a non-negative integer. We denote by $\mathcal{O}_{\mathsf{P}}$ an oracle which perfectly solves the problem P, that is, on any input $\mathtt{ins} \in \mathbf{I}$, the oracle outputs some $\mathtt{sol}$ such that $\mathsf{V}(\mathsf{param}, \mathtt{ins}, \mathtt{sol}) = 1$. The one-more n-P problem is defined by the following experiment for an algorithm $\mathcal{A}$.

Experiment $\mathbf{Exp}^{n\text{-}\mathsf{P}}_{\mathcal{A}}(k)$.
  $\mathsf{param} \leftarrow \mathsf{PG}(1^k)$
  For $i = 0, \ldots, n$, generate $\mathtt{ins}_i \leftarrow \mathsf{IG}(\mathsf{param}; \varpi_i)$ with fresh random tapes $\varpi_0, \ldots, \varpi_n$.
  $(\mathtt{sol}_0, \ldots, \mathtt{sol}_n) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{P}}}(\mathtt{ins}_0, \ldots, \mathtt{ins}_n)$
  If $\mathsf{V}(\mathsf{param}, \mathtt{ins}_i, \mathtt{sol}_i) = 1$ for $i = 0, \ldots, n$ and $\mathcal{A}$ made at most n queries to $\mathcal{O}_{\mathsf{P}}$ output 1 else output 0

The success probability of $\mathcal{A}$ is $\mathsf{Succ}^{n\text{-}\mathsf{P}}_{\mathcal{A}}(k) = \Pr[\mathbf{Exp}^{n\text{-}\mathsf{P}}_{\mathcal{A}}(k) = 1]$, where the probability is taken over the random coins of all involved algorithms. For any functions $\varepsilon, \tau : \mathbf{N} \to \mathbf{R}$, an algorithm $\mathcal{A} \in \mathcal{CL}(n\text{-}\mathsf{P}, \varepsilon, \tau)$ is called an (ε, τ)-n-P solver.

[6] We separate PG and IG for exposition convenience.

Proposition 1 (Reduction from $(n+1)$-P to n-P). *Let n, m be two integers such that $n < m$. Then the m-P problem cannot be harder than the n-P problem.*

We omit the proof since it is elementary. We now give a definition for a computational problem P to be random self-reducible.

Definition 1 (Random self-reducibility). *A problem P defined as above is said to be random self-reducible if there exists an efficient blinding algorithm B and an efficient un-blinding algorithm UB such that for any $k \in \mathbf{N}$, any string param generated by PG, and any element $\mathtt{ins} \in \mathbf{I}$:*

1. $\mathsf{B}(\mathsf{param}, \mathtt{ins}; \varpi)$ *is a uniformly distributed element $\mathtt{ins}_{\mathrm{bl}}$ in $\mathbf{I}$, w.r.t. the random choice of ϖ;*
2. *for any random tape ϖ, any blinded instance $\mathtt{ins}_{\mathrm{bl}}$ generated by B from instance $\mathtt{ins}$ using random tape ϖ, the algorithm UB satisfies*

$$\mathsf{V}(\mathsf{param}, \mathtt{ins}_{\mathrm{bl}}, \mathtt{sol}_{\mathrm{bl}}) = 1 \implies \mathsf{V}(\mathsf{param}, \mathtt{ins}, \mathsf{UB}(\mathsf{param}, \mathtt{sol}_{\mathrm{bl}}; \varpi)) = 1.$$

In what follows, we denote by Ω the set of random tapes ϖ used by B (and UB) and the time complexity of algorithms B, UB, V by $\tau_{\mathrm{BL}}, \tau_{\mathrm{UB}}, \tau_{\mathrm{VER}}$ respectively. We remark that our definition is similar to that of Okamoto-Pointcheval [14] except that we do not require that UB outputs a uniform element in $\mathbf{S}$. Our definition is also like that given by Tompa-Woll [18] with the relaxation of a similar condition. Note that both the discrete logarithm and the RSA inversion problems satisfy this definition.

3.2 Black-Box Separation

Definition 2 (Parameter-invariant reductions). *Let n and n' be some non-negative integers. A black-box reduction $\mathcal{R}$ from n-P to n'-P is said to be* parameter-invariant *if $\mathcal{R}$ only feeds the n'-P-solver with challenges containing the same string param that was in the challenge given to $\mathcal{R}$.*

We first assume that the reduction executes the $(n+1)$-P-solver at most one time and never rewinds.

Lemma 1 (Separation, basic case). *Let n be a non-negative integer and $\varepsilon, \varepsilon', \tau_r$ be some positive functions defined on $\mathbf{N}$. We set $\tau_{\mathrm{TOT}} := \tau_{\mathrm{BL}} + \tau_{\mathrm{UB}} + \tau_{\mathrm{VER}}$. There exists a meta-reduction $\mathcal{M}$ such that, for any parameter-invariant black-box $(\varepsilon, \varepsilon', \tau_r)$-reduction $\mathcal{R}$ from n-P to $(n+1)$-P which makes at most only one "**Launch**" (and no "**Rewind**") query to the $(n+1)$-P-solver, $\mathcal{M}^{\mathcal{R}}$ is an $(\varepsilon', \tau_r + (n+1) \cdot \tau_{\mathrm{TOT}})$-$n$-$\mathsf{P}$ solver.*

Proof. First, remark that for any $\epsilon > 0$ there exists a $(n+1)$-P-solver that succeeds with probability ϵ. This is because P is verifiable so the exhaustive search is possible, and that we do not consider the execution time at that point. Moreover, we can assume that this solver always makes $n+1$ uniformly distributed queries to $\mathcal{O}_{\mathsf{P}}$. We denote this (naive) algorithm by $\mathcal{A}_1$. It receives as input an instance of $(n+1)$-P, picks $n+1$ uniform random P-instances $\mathtt{ins}_1^*, \ldots, \mathtt{ins}_{n+1}^* \in_U \mathbf{I}$, and submits them sequentially to $\mathcal{O}_{\mathsf{P}}$. For each query, $\mathcal{A}_1$ checks the answer of $\mathcal{O}_{\mathsf{P}}$ using the verification algorithm V and if one answers is wrong, $\mathcal{A}_1$ outputs $\perp$ and aborts. Otherwise, it does an exhaustive search and outputs the correct answer of the $(n+1)$-P challenge with probability ϵ.

One may wonder why it is necessary to check the validity of the $\mathcal{O}_{\mathsf{P}}$ oracle, since we have assumed it is a perfect oracle. Indeed, in the real $\mathbf{Exp}_{\mathcal{A}}^{(n+1)\text{-}\mathsf{P}}(k)$ experiment, such a verification is always successful ($\mathcal{A}_1$ never outputs $\perp$). However these checks will be crucial in the simulated experiment in which $\mathcal{O}_{\mathsf{P}}$ is simulated by $\mathcal{R}$, which can possibly cheats. We also emphasize that the queries $\mathtt{ins}_i^*$ are always the same if the random tape of $\mathcal{A}_1$ and param are the same.

Description of the Meta-Reduction. We now explain how to build a meta-reduction $\mathcal{M}$ that solves the n-P problem. First, $\mathcal{M}$ is given access to a $\mathcal{O}_{\mathsf{P}}$-oracle (with at most n queries allowed) and receives an n-P challenge $(\mathsf{param}, \mathtt{ins}_0, \dots, \mathtt{ins}_n)$. Then it will use $\mathcal{R}$ to solve it, by simulating adversary $\mathcal{A}_1$ (which is assumed to make uniform queries). To this goal, $\mathcal{M}$ simply feeds $\mathcal{R}$ with the given challenge. Remind that $\mathcal{R}$ has oracle access to $\mathcal{O}_{\mathsf{P}}$ and $\mathcal{A}_1$ (the so-called "**Launch**" query), so $\mathcal{M}$ must be able to answer both types of queries. The $\mathcal{O}_{\mathsf{P}}$ queries are answered in a trivial way; namely, $\mathcal{M}$ simply forwards the queries to its oracle $\mathcal{O}_{\mathsf{P}}$. Below, we describe how $\mathcal{M}$ processes the (unique) "**Launch**" query made by $\mathcal{R}$, by simulating the execution of $\mathcal{A}_1$.

Launch$(\mathsf{param}, \mathtt{ins}'_0, \dots, \mathtt{ins}'_{n+1})$
for $i = 0, 1, \dots, n$ **do**
  $\varpi_i \leftarrow_U \Omega$
  $\mathtt{bl\text{-}ins}_i \leftarrow \mathsf{B}(\mathsf{param}, \mathtt{ins}_i; \varpi_i)$
  Submit $\mathtt{bl\text{-}ins}_i$ to $\mathcal{O}_{\mathsf{P}}$ (simulated by $\mathcal{R}$) and receive $\mathtt{bl\text{-}sol}_i$
  $\mathtt{sol}_i \leftarrow \mathsf{UB}(\mathsf{param}, \mathtt{bl\text{-}sol}_i; \varpi_i)$
  if $0 \leftarrow \mathsf{V}(\mathsf{param}, \mathtt{ins}_i, \mathtt{sol}_i)$ **then**
    Return $\perp$
Abort the interaction with $\mathcal{R}$

If all the $\mathcal{O}_{\mathsf{P}}$-queries are correctly answered by $\mathcal{R}$, the meta-reduction aborts the interaction with $\mathcal{R}$ after the $(n+1)$-th $\mathcal{O}_{\mathsf{P}}$-query, and simply returns $(\mathtt{sol}_0, \dots, \mathtt{sol}_n)$ as the solution of the n-P problem given as input. On the other hand, if the simulation of the oracle appears to be wrong, the "**Launch**" query is answered with $\perp$ — this correctly simulates the behavior of $\mathcal{A}_1$ described at the beginning of the proof. In that case, the reduction eventually outputs a tuple that $\mathcal{M}$ just relays as its own answer. If $\mathcal{R}$ does not ask any "**Launch**" query or decides to prematurely "**Stop**" the interaction with $\mathcal{A}_1$, then $\mathcal{M}$ also outputs the same answer as $\mathcal{R}$.

In simulating $\mathcal{A}_1$, $\mathcal{M}$ needs to run the algorithms B, UB, and V once per query to $\mathcal{O}_{\mathsf{P}}$. Since $\mathcal{M}$ launches $\mathcal{R}$ one time, we get $T(\mathcal{M}^{\mathcal{R}}) \le \tau_r + (n+1) \cdot \tau_{\text{TOT}}$.

Probability Analysis. Let $\mathsf{Succ}_{\mathcal{R}}$ and $\mathsf{Succ}_{\mathcal{M}}$ be the events that "$\mathcal{R}$ succeeds" and "$\mathcal{M}$ succeeds" respectively. Let us denote Good the event, where $\mathcal{R}$ correctly answers to all $\mathcal{O}_{\mathsf{P}}$ queries made by $\mathcal{A}_1$. In particular, event $\neg\mathsf{Good}$ includes the executions in which $\mathcal{R}$ does not make any "**Launch**" query or prematurely stops the interaction with $\mathcal{A}_1$. From the above description, we can easily see that, if Good occurs, $\mathcal{M}$ always recovers the correct solutions $\mathtt{sol}_i$ for $i = 0, \dots, n$. On the other hand, if $\neg\mathsf{Good}$ occurs, $\mathcal{M}$ outputs whatever $\mathcal{R}$ outputs, and thus we have $\Pr[\mathsf{Succ}_{\mathcal{M}} | \neg\mathsf{Good}] = \Pr[\mathsf{Succ}_{\mathcal{R}} | \neg\mathsf{Good}]$. Then we obtain

$$\begin{aligned}
\Pr[\mathsf{Succ}_{\mathcal{M}}] &= \Pr[\mathsf{Succ}_{\mathcal{M}} | \mathsf{Good}] \cdot \Pr[\mathsf{Good}] + \Pr[\mathsf{Succ}_{\mathcal{M}} | \neg\mathsf{Good}] \cdot \Pr[\neg\mathsf{Good}] \\
&= \Pr[\mathsf{Good}] + \Pr[\mathsf{Succ}_{\mathcal{R}} | \neg\mathsf{Good}] \cdot \Pr[\neg\mathsf{Good}] \\
&\ge \Pr[\mathsf{Succ}_{\mathcal{R}} \wedge \mathsf{Good}] + \Pr[\mathsf{Succ}_{\mathcal{R}} \wedge \neg\mathsf{Good}] = \Pr[\mathsf{Succ}_{\mathcal{R}}] \ge \varepsilon',
\end{aligned}$$

which concludes the proof. □

Theorem 1 (Separation, general case). *Let $n, \varepsilon, \varepsilon', \tau_r, \tau_{\text{TOT}}$ be as in Lemma 1. There exists a meta-reduction $\mathcal{M}$ such that, for any parameter-invariant black-box $(\varepsilon, \varepsilon', \tau_r)$-reduction $\mathcal{R}$ from n-P to $(n+1)$-P which makes at most ℓ queries "**Launch**" or "**Rewind**" to the $(n+1)$-P-solver, $\mathcal{M}^{\mathcal{R}}$ is an $(\varepsilon', \tau_r + (n+1)\ell \cdot \tau_{\text{TOT}})$-$n$-$\mathsf{P}$-solver.*

Proof. The proof works like in Lemma 1 except that $\mathcal{M}$ needs to deal with possibly many runs and/or rewindings of the $(n+1)$-P-solver $\mathcal{A}_1$. We summarize how we deal with the queries of type "**Launch**" and "**Rewind**".

Launch. For each such query, $\mathcal{M}$ needs to simulate a new execution of $\mathcal{A}_1$ with fresh random coins. The simulation works like for Lemma 1. The main difference is that, when a wrong simulation of $\mathcal{O}_\mathsf{P}$ leads to an abortion of $\mathcal{A}_1$, the reduction $\mathcal{R}$ is allowed to continue with a new query "**Launch**" or "**Rewind**". The same holds if $\mathcal{R}$ aborts the current execution by itself by just making a new query. This does not affect the simulation of $\mathcal{A}_1$ by $\mathcal{M}$, which simply waits for a correct simulation of $\mathcal{O}_\mathsf{P}$ by $\mathcal{R}$ on the $n+1$ queries[7].

Rewind. The simulation of such an execution is done similarly as for "**Launch**" except that the randomness is not fresh. This implicitly means that $\mathcal{M}$ keeps track of the history of previous $\mathcal{A}_1$'s executions in a non-ambiguous way. Recall that the $\mathcal{O}_\mathsf{P}$ queries made by $\mathcal{A}_1$ does not depend on the received n-P instance, but only its random tape and param, which is unchanged by definition. Thus, if $\mathcal{R}$ rewinds several times, $\mathcal{M}$ can simulate correctly: the same queries to the $\mathcal{O}_\mathsf{P}$ oracle are repeated from a previous execution, and a wrong answer is still detected, leading to an abortion. In that case, $\mathcal{R}$ continues with another query "**Launch**" or "**Rewind**", or makes its final "**Stop**". As above, $\mathcal{M}$ stops the execution if it receives all answers of the queries sent to $\mathcal{O}_\mathsf{P}$ and wins the game[8]. If this does not occur, $\mathcal{M}$ simply answers the $\mathcal{R}$ output to its challenger.

A probability analysis as in Lemma 1 shows that $\mathcal{M}$ succeeds with probability at least ε'. The running time of $\mathcal{M}^{\mathcal{R}}$ is easily upper-bounded by $\tau_r + (n+1)\ell \cdot \tau_{\text{TOT}}$. □

Remark 1. We observe that the reduction $\mathcal{R}$ solves the n-P problem using an $(n+1)$-P solver as a subroutine, while our meta-reduction $\mathcal{M}$ solves the same problem without such subroutine (but with access to $\mathcal{R}$). Thus, if the n-P problem is hard, the existence of an efficient $\mathcal{M}$ shows that there cannot exist an efficient black-box algorithm $\mathcal{R}$ from the n-P problem to the $(n+1)$-P problem. On the other hand, if the n-P problem is easy, we know that the $(n+1)$-P problem is easy as well, and so basing the security of cryptographic schemes on these problems (m-P for $m \geq n$) would be hopeless.

Remark 2. Note that if the problem P is not efficiently verifiable, the above result does not apply anymore. For instance, this does not work with the one-more (static) Computational Diffie-Hellman, except in a group with an easy decisional Diffie-Hellman problem (DDH). Namely, if $\mathcal{R}$ simulates the oracle by answering random elements, the adversary $\mathcal{A}_1$ cannot easily detect that the simulation is not correct unless this one can solve DDH. However, in the context of pairing cryptography the bilinear pairing allows to efficiently solve DDH. In this case, our results apply to the one-more CDH.

3.3 The Case of Sophisticated Reductions

Here, we investigate our separation results in the context of "sophisticated" reductions, i.e., those which are supposed to only transform a class of algorithms with a bounded time complexity. In what follows, we are able to exhibit such a separation under the assumption that the reduction does not rewind the adversary with the same random and a different input.

[7] Note that we can optimize by simply waiting until $\mathcal{M}$ could get $n+1$ correct answers of fresh queries made to $\mathcal{O}_\mathsf{P}$, even if these ones were not made during one single execution of $\mathcal{A}_1$. For this, $\mathcal{M}$ should not re-use a solved $\mathtt{ins}_i$ by sending a new blinded $\mathtt{ins}_i$ value to $\mathcal{O}_\mathsf{P}$ in a subsequent fresh (new random tape) execution of $\mathcal{A}_1$.

[8] Though the queries are the same, $\mathcal{R}$ may simulate $\mathcal{O}_\mathsf{P}$ in a different way. In particular, it may succeed in simulating $\mathcal{O}_\mathsf{P}$, even if it failed to do so in a previous execution.

Theorem 2 (Sophisticated reductions). *Let $n, \varepsilon, \varepsilon', \tau_r, \tau_{\text{TOT}}$ be as in Lemma 1. Consider $\tau_0 : \mathbf{N} \to \mathbf{R}$ an arbitrary time-complexity upper bound of some existing $(n+1)$-P solver succeeding with probability greater or equal to ε. In other words, we assume*

$$\mathcal{CL}((n+1)\text{-}\mathsf{P}, \varepsilon, \tau_0) \neq \emptyset.$$

*Let τ such that $\tau(k) \geq \tau_0(k) + (n+1) \cdot \tau_{\text{TOT}}(k)$ for any $k \in \mathbf{N}$. There exists a meta-reduction $\mathcal{M}$ such that, for any "sophisticated" parameter-invariant black-box $(\varepsilon, \tau, \varepsilon', \tau_r)$-reduction $\mathcal{R}$ from n-P to $(n+1)$-P making at most ℓ queries "**Launch**" or "**Relaunch**" to the $(n+1)$-P-solver, $\mathcal{M}^{\mathcal{R}}$ is an $(\varepsilon', \tau_r + (n+1)\ell \cdot \tau_{\text{TOT}})$-$n$-$\mathsf{P}$-solver.*

Proof. The proof is similar to that of Theorem 1 except that the existence of a certain $(n+1)$-P-solver $\mathcal{A}_1$ belonging to $\mathcal{CL}((n+1)\text{-}\mathsf{P}, \varepsilon, \tau)$ needs to be shown ($\mathcal{A}_1$ is no more a naive algorithm). It is constructed as follows.

By definition of τ_0, there exists an algorithm $\mathcal{A}_0$ belonging to the class $\mathcal{CL}((n+1)\text{-}\mathsf{P}, \varepsilon, \tau_0)$. The algorithm $\mathcal{A}_1$ receives as input an instance of the $(n+1)$-P-problem, starts $\mathcal{A}_0$ with this input, processes the $\mathcal{A}_0$'s queries as explained hereafter, and finally outputs whatever $\mathcal{A}_0$ outputs. For each $\mathcal{O}_{\mathsf{P}}$ query $\texttt{ins}$, $\mathcal{A}_1$ picks a uniformly distributed random tape ϖ and computes $\texttt{ins}_{\text{bl}} \leftarrow \mathsf{B}(\mathsf{param}, \texttt{ins}; \varpi)$. It then queries $\texttt{ins}_{\text{bl}}$ to $\mathcal{O}_{\mathsf{P}}$ and gets the answer $\texttt{sol}_{\text{bl}}$. It checks whether $\mathsf{V}(\mathsf{param}, \texttt{ins}_{\text{bl}}, \texttt{sol}_{\text{bl}}) \to 1$: if it is the case, it forwards $\texttt{sol} \leftarrow \mathsf{UB}(\mathsf{param}, \texttt{sol}_{\text{bl}}; \varpi)$ as the answer to $\mathcal{A}_0$, otherwise it terminates and outputs $\perp$. If $\mathcal{A}_0$ asks less than $n+1$ queries to $\mathcal{O}_{\mathsf{P}}$, $\mathcal{A}_1$ asks as many uniformly distributed random queries as necessary.

This algorithm $\mathcal{A}_1$ has the same behavior as in Theorem 1 (it always makes $(n+1)$ uniform queries), except that for a given randomness the queries to $\mathcal{O}_{\mathsf{P}}$ may depend on the input. The rest of the proof works like for Theorem 1. In particular, $\mathcal{M}$ simulates $\mathcal{A}_1$ in the very same way and handles the "**Relaunch**" queries made by $\mathcal{R}$ as the "**Rewind**" ones in the proof of Theorem 1. □

Remark 3. The difficulty of extending the above proof to "**Rewind**" queries comes from our inability to correctly simulate $\mathcal{A}_1$ after a rewinding of $\mathcal{R}$ with a *different* input. Since $\mathcal{A}_1$ must be restarted with the same random tape, we cannot produce uniform $\mathcal{O}_{\mathsf{P}}$-queries anymore: the blinding on a different input with the same randomness would produce different blinded queries, while in Theorem 1 the queries should not change in case of a "**Rewind**".

4 One-More Discrete Logarithm Problems

4.1 Definitions

Let $k \in \mathbf{N}$ be a security parameter and Gen be an efficient algorithm taking k (or 1^k) as input and which outputs the description of a cyclic group $\mathbf{G}$ of prime order (written multiplicatively), a generator g of $\mathbf{G}$, and the k-bit prime group order $q = \#\mathbf{G}$. We assume that elementary group operations in $\mathbf{G}$ can be done efficiently, i.e., $g_1 g_2$ and g_1^{-1} can be efficiently computed for any $g_1, g_2 \in \mathbf{G}$, and we denote by τ_{EXP} the time required for computing an exponentiation g^x, where $x \in [1, q]$. We also consider a perfect discrete logarithm oracle DL_g, i.e., an oracle which on any queried element always answers its discrete logarithm with respect to g. For a non-negative integer n, the n-DL problem (one-more discrete logarithm) consists in extracting the discrete logarithms of $n+1$ elements of $\mathbf{G}$ with respect to g using at most n queries to the oracle DL_g. More formally, for an algorithm $\mathcal{A}$, we consider the following experiment [2]:

Experiment $\mathbf{Exp}^{n\text{-}\mathsf{DL}}_{\mathsf{Gen},\mathcal{A}}(k)$.

$(\mathbf{G}, q, g) \leftarrow \mathsf{Gen}(1^k)$

$(t_0, t_1, \ldots, t_n) \leftarrow_U \mathbf{Z}_q^{n+1}$; $y_i \leftarrow g^{t_i}$ for $i = 0, \ldots, n$

$(t'_0, \ldots, t'_n) \leftarrow \mathcal{A}^{\mathsf{DL}_g}(\mathbf{G}, g, q, y_0, \ldots, y_n)$

Return 1 if the following conditions hold else return 0

- $t'_i \equiv t_i \pmod q$ for all $i = 0, \ldots, n$
- DL_g has been queried at most n times

We define the success probability of $\mathcal{A}$ in the above experiment as

$$\mathsf{Succ}^{n\text{-}\mathsf{DL}}_{\mathsf{Gen},\mathcal{A}}(k) = \Pr[\mathbf{Exp}^{n\text{-}\mathsf{DL}}_{\mathsf{Gen},\mathcal{A}}(k) = 1],$$

where the probability is taken over the t_i's and the random tapes of Gen and $\mathcal{A}$.

For any functions $\varepsilon, \tau : \mathbf{N} \to \mathbf{R}$, we denote by $\mathcal{DL}(n, \varepsilon, \tau)$ the set $\mathcal{CL}(n\text{-}\mathsf{DL}, \varepsilon, \tau)$. An algorithm $\mathcal{A}$ of this class is said to be an (ε, τ)-n-DL solver.

4.2 Algebraic Separations

First, we note that Theorems 1 and 2 apply to the discrete logarithm problems if we assume that the reduction is base-invariant, i.e., it always feeds the $(n+1)$-DL solver with the same group $\mathbf{G}$ and generator g given in the n-DL experiment. In what follows, we show that we can extend these separation results to some non-invariant base (but same group) reductions under the assumption that these reductions are algebraic. We restrict to classical black-box reductions with rewinding and follow the spirit of Theorem 1. A treatment of "sophisticated" reductions with a rewinding relaxation (as in Theorem 2) can be done in the very same manner so that we omit such a presentation.

Algebraic Algorithms. We use the concept of algebraic algorithms introduced by Paillier and Vergnaud [15]. Roughly, an algorithm $\mathcal{R}$ is *algebraic* with respect to a cyclic group $\mathbf{G}$ (of order q) if any element of $\mathbf{G}$ output by the algorithm at any step can be described as an explicitly known "multiplicative combination" of its $\mathbf{G}$ inputs. More precisely, there should exist an algorithm $\mathsf{Extract}$ which, given the random tape ϖ of $\mathcal{R}$, its inputs $(s, g_1, \ldots, g_\ell) \in \{0,1\}^* \times \mathbf{G}^\ell$, and its code $\mathsf{co}(\mathcal{R})$, enables to retrieve, for any $y \in \mathbf{G}$ output by $\mathcal{R}$, the coefficients $a_1, \ldots, a_\ell \in \mathbf{Z}_q$ such that

$$y = g_1^{a_1} \cdots g_\ell^{a_\ell}.$$

Moreover, it is required that the procedure $\mathsf{Extract}$ runs in polynomial time with respect to $|\mathsf{co}(\mathcal{R})|$ (the size of the code of $\mathcal{R}$) and $\tau = T(\mathcal{R})$. We denote the time complexity of one $\mathsf{Extract}$ execution by τ_{EXT}. Though an algebraic algorithm may not be treated as a black-box, we will use the notation $\mathcal{M}^{\mathcal{R}}$ to express the algorithm obtained by an algorithm $\mathcal{M}$ which uses $\mathcal{R}$ as a subroutine and possibly makes calls to $\mathsf{Extract}$.

Definition 3 (Group-invariant reductions). *Let n and n' be two non-negative integers. A reduction $\mathcal{R}$ from n-DL to n'-DL is said to be* group-invariant *if $\mathcal{R}$ exclusively feeds the n'-DL solver with challenges containing the same group $\mathbf{G}$ which was given by Gen in the n-DL experiment.*

Theorem 3 (Separation for algebraic reductions). *Let $n, \varepsilon, \varepsilon', \tau_r$ be as in Lemma 1. There exists a meta-reduction $\mathcal{M}$ (non black-box) such that, for any algebraic group-invariant black-box $(\varepsilon, \varepsilon', \tau_r)$-reduction $\mathcal{R}$ from n-DL to $(n+1)$-DL which makes at most ℓ "**Launch**" or "**Rewind**" queries to the underlying $(n+1)$-DL-solver, $\mathcal{M}^{\mathcal{R}}$ is an $(\varepsilon', \tau_r + 2(n+1)\ell \cdot \tau_{\text{EXP}} + \ell \cdot \tau_{\text{EXT}})$-$n$-$\mathsf{DL}$ solver.*

Proof. This proof is similar to that of Theorem 1 except that $\mathcal{M}$ needs to simulate the $(n+1)$-DL-solver $\mathcal{A}_1$ in a different way.

DESCRIPTION OF $\mathcal{M}$. At the beginning, $\mathcal{M}$ is challenged with $(\mathbf{G}, g_1, q, y_0, \dots, y_n)$ and forwards this challenge to $\mathcal{R}$. Then $\mathcal{M}$ has to deal with the queries made by $\mathcal{R}$: "**Launch**", "**Rewind**" and queries to the DL_{g_1} oracle. That latter is simulated in a straightforward way by the meta-reduction, since its own oracle is relative to base g_1 as well and the number of queries asked by $\mathcal{R}$ is less than n. For the "**Launch**" and "**Rewind**" queries, we have to show how $\mathcal{M}$ simulates the $(n+1)$-DL-solver $\mathcal{A}_1$. We assume that at least one execution of $\mathcal{A}_1$ will terminate correctly: $\mathcal{A}_1$ asks $n+1$ discrete logarithm queries and receives correct answers. We denote this event by Good.

The subtlety is as follows. On a "**Rewind**"-query, $\mathcal{R}$ can specify another base (and the DL-queries made by $\mathcal{A}_1$ will be answered by $\mathcal{R}$ relatively to this base). However, $\mathcal{A}_1$ being started with new inputs but unchanged random tape must ask the same DL-queries (they only depend on the random tape). We now show that this is not a problem, as long as one execution goes correctly (event Good). For convenience of notation, we denote g_1 as y_{-1} and for any $i \in [-1, n]$ we note $\alpha_i = \log_{g_2} y_i$. For completeness we explicitly specify the random tape ϖ of the reduction $\mathcal{R}$ (keep in mind that this randomness is provided by $\mathcal{M}$ which has *non black-box* access to $\mathcal{R}$).

A "**Launch**"-query is processed as follows:

Launch$(\mathbf{G}, g_2, q, z_0, \dots, z_{n+1})$
$(b_{-1}, b_0, \dots, b_n) \leftarrow \mathsf{Extract}(g_2, \varpi, \mathsf{co}(\mathcal{R}))$ `//` `we have:` $g_2 = \prod_{j=-1}^{n} y_j^{b_j}$
`// up to a permutation, we can assume` $b_{-1} \neq 0$
for $i = 0$ to n **do**
  $r_i \leftarrow_U \mathbf{Z}_q$ `//` $\mathcal{A}_1$ `asks at most` $n+1$ `queries`
  Submit $g_2^{r_i} y_i$ to DL_{g_2} (simulated by $\mathcal{R}$) and receive answer θ_i
  $\alpha_i \leftarrow \theta_i - r_i$ `// clearly` $\alpha_i = \log_{g_2} y_i = x_i$
$\alpha_{-1} \leftarrow b_{-1}^{-1}(1 - \sum_{j=0}^{n} b_j \alpha_j)$ `// we have` $\alpha_{-1} = \log_{g_2} g_1 = x_{-1} \neq 0$
for $i = 0$ to n **do**
  $c_i \leftarrow \alpha_i / \alpha_{-1} \bmod q$ `//` $c_i = \log_{g_1} y_i$
Abort the interaction with $\mathcal{R}$

From above, it is clear that if a "**Launch**"-query goes successfully ($n+1$ DL-queries that are answered correctly), $\mathcal{M}$ is able to recover all $c_i = \log_{g_1} y_i$ for $i \in [0, n]$.

On the other hand, if the first $\mathcal{A}_1$'s execution that goes successfully[9] is a "**Rewind**"-query, then $\mathcal{M}$ does as follows. Let us denote by g_2' the (new) generator provided by $\mathcal{R}$ as an input of this query, $\mathcal{M}$ still constructs the DL-queries as $g_2^{r_i} y_i$ (and not ${g_2'}^{r_i} y_i$). However the answers are relative to base g_2' and must be exploited differently. We first note that we have $n+3$ equations: one for the $\mathsf{Extract}(g_2, \dots)$-query made in the underlying "**Launch**"-query, one for the $\mathsf{Extract}(g_2', \cdots)$ in the successful "**Rewind**", and $n+1$ equations $\delta\theta_i = r_i + x_i$, for $i = 0, \dots, n$ with $\delta = \log_{g_2} g_2'$. The obtained matrix relative to the linear system with $n+3$ unknowns

[9] We recall that a "successful" execution is defined by the fact that $\mathcal{A}_1$ receives correct answers, not that $\mathcal{A}_1$ terminates. In fact it never terminates, since $\mathcal{M}$ aborts $\mathcal{A}_1$ as soon as it has enough information to conclude.

$(x_{-1}, x_0, \ldots, x_n, \delta)$ is:

$$\begin{pmatrix} b_{-1} & b_0 & b_1 & \cdots & b_n & 0 \\ b'_{-1} & b'_0 & b'_1 & \cdots & b'_n & -1 \\ & 1 & & & & -\theta_0 \\ & & 1 & & & -\theta_1 \\ & & & \ddots & & \vdots \\ & & & & 1 & -\theta_n \end{pmatrix} \quad \begin{array}{r} \texttt{// from } g_2 = \prod_{-1}^n y_i^{b_i} \\ \texttt{// from } g'_2 = \prod_{-1}^n y_i^{b'_i} \\ \\ \\ \texttt{// from } {g'_2}^{\theta_i} = g_2^{r_i} y_i = g_2^{r_i + x_i} \\ \\ \end{array}$$

Up to the sign, the determinant of this matrix is easily seen to be

$$\pm\Delta = b_{-1}\Big(-1 \cdot 1 + \sum_0^n b'_i \theta_i \Big) - b'_{-1}\Big(\sum_0^n b_i \theta_i \Big)$$

From the two "$\mathsf{Extract}$" equations it is easy to see that for any $i \in [-1, n]$ we have $b'_i = \delta b_i$ with overwhelming probability (if it was not the case, linearly combining the two equations would lead to some x_i by expliciting $(b'_i - \delta b_i)x_i = \cdots$). It follows immediately that $\Delta = \pm b_{-1} \neq 0$.

As a conclusion, as soon as event Good occurs, the meta-reduction can solve the system and obtain all the $\log_{g_2} y_i$ as well as $\log_{g_2} g_1$, and thus can solve its challenge.

Otherwise (if $\mathcal{R}$ always halts $\mathcal{A}_1$ before it asks $n+1$ DL-queries or if $\mathcal{R}$ always answers incorrectly), then $\mathcal{M}$ outputs whatever $\mathcal{R}$ outputs. Thus:

$$\Pr[\mathsf{Succ}_\mathcal{M}] = \underbrace{\Pr[\mathsf{Succ}_\mathcal{M} | \mathsf{Good}]}_{\text{we show } =1} \Pr[\mathsf{Good}] + \underbrace{\Pr[\mathsf{Succ}_\mathcal{M} | \neg\mathsf{Good}]}_{=\Pr[\mathsf{Succ}_\mathcal{R} | \neg\mathsf{Good}]} \Pr[\neg\mathsf{Good}] \geq \epsilon'$$

(as in Lemma 1). The running time is easily checked. □

Definition 4 (Regular reductions). *Let $\mathcal{R}$ be a black-box reduction from a problem P_2 to a problem P_1. We denote by* suc *the success event for $\mathcal{R}$ in solving P_2 and* Exec *the event that $\mathcal{R}$ launches at least one complete execution of the P_1-solver and correctly simulates its environment. The reduction $\mathcal{R}$ is said to be* regular *if the event* Succ *always implies the event* Exec.

This definition captures the fact that the reduction really exploits the access given to the P_1-solver. This assumption seems quite natural, since the reduction would simply be a P_2-solver otherwise. The following result shows that the separation holds under the hardness of the DL problem (rather than the one-more DL) if we assume $\mathcal{R}$ to be regular. Moreover we get an improvement on the "extra" time of $\mathcal{M}$ w.r.t. $\mathcal{R}$, which drops from $2(n+1)\ell \cdot \tau_{\text{EXP}} + \ell \cdot \tau_{\text{EXT}}$ down to $2\ell \cdot \tau_{\text{EXP}} + (n+\ell) \cdot \tau_{\text{EXT}}$.

Theorem 4 (Separation for regular reductions). *Let $n, \varepsilon, \varepsilon', \tau_r$ be as in Lemma 1. There exists a (non-black-box) meta-reduction $\mathcal{M}$ such that, for any regular algebraic group-invariant black-box $(\varepsilon, \varepsilon', \tau_r)$-reduction $\mathcal{R}$ from n-DL to $(n+1)$-DL which makes at most ℓ queries of type "**Launch**" or "**Rewind**" to the $(n+1)$-DL-solver, $\mathcal{M}^\mathcal{R}$ is an $(\varepsilon', \tau_r + 2\ell \cdot \tau_{\text{EXP}} + (n+\ell) \cdot \tau_{\text{EXT}})$-$\mathsf{DL}$-solver.*

Proof. The proof differs from that of Theorem 3 in the way $\mathcal{M}$ simulates the oracle DL_{g_1} and feeds the reduction.
Description of $\mathcal{M}$. At the beginning, $\mathcal{M}$ is challenged with $(\mathbf{G}, g_1, q, y)$. The meta-reduction picks a tuple $(r_0, \ldots r_n) \leftarrow_U \mathbf{Z}_q^n$ and computes $w_i \leftarrow g_1^{r_i}$ for $i = 0, \ldots, n$. Then, $\mathcal{M}$ feeds $\mathcal{R}$ with $(\mathbf{G}, g_1, q, w_0, \ldots, w_n)$ and has to deal with the queries made by $\mathcal{R}$. The DL_{g_1} oracle is simulated using the algebraicity of $\mathcal{R}$ as follows:

Query $\mathsf{DL}_{g_1}(u)$
$(a, a_0, \ldots, a_n) \leftarrow \mathsf{Extract}(u, \varpi, \mathsf{co}(\mathcal{R}))$ `// we have:` $u = g_1^a \cdot \prod_{i=0}^n w_i^{a_i}$
Return $a + \sum_{i=0}^n a_i r_i \bmod q$

We now describe how $\mathcal{M}$ simulates $\mathcal{A}_1$ (on "**Launch**" queries).

Launch$(\mathbf{G}, g_2, q, z_0, \ldots, z_{n+1})$
$(b, c_0, \ldots, c_n) \leftarrow \mathsf{Extract}(g_2, \varpi, \mathsf{co}(\mathcal{R}))$ `// we have:` $g_2 = g_1^b \cdot \prod_{i=0}^n w_i^{c_i}$
$\alpha \leftarrow b + \sum_{i=0}^n r_i c_i$ `//` $\alpha = \log_{g_1}(g_2)$
$r \leftarrow_U \mathbf{Z}_q$
Submit $g_2^r \cdot y$ to DL_{g_2} (simulated by $\mathcal{R}$) and receive $r + \beta$ `//` $\beta = \log_{g_2}(y)$
$d \leftarrow \alpha \cdot \beta$
Abort the interaction with $\mathcal{R}$

As in previous proof, the interaction is aborted if $\mathcal{R}$ answers incorrectly. By assumption on $\mathcal{R}$, there always exists a successful interaction ($\mathcal{R}$ answering correctly). If this is a "**Launch**"-query, we can easily see from the above that $\mathcal{M}$ successfully outputs $d = \log_{g_1} y$. If this is a "**Rewind**$(g_2', \cdots)$"-query, then we have three unknowns: α, β and δ (the values of $\log_{g_1} g_2$, $\log_{g_2} y$ and $\log_{g_2} g_2'$, respectively) and three equations:

$$\begin{cases} \alpha = b + \sum_i r_i c_i & \texttt{// from } \mathsf{Extract}(g_2, \cdots) \\ \delta \cdot \alpha = b' + \sum_i r_i c_i' & \texttt{// from } \mathsf{Extract}(g_2', \cdots) \\ \delta \cdot \theta = r + \beta & \texttt{// answer } \theta \texttt{ to } \mathsf{DL}_{g_2'}(g_2^r y) \end{cases}$$

This is clearly solvable. Thus after one sucessful execution of $\mathcal{A}_1$, $\mathcal{M}$ is always able to compute its solution $\log_{g_1} y = \alpha \cdot \beta$. The "regular" notion ensures that $\mathcal{M}$ has a success probability greater or equal to ε'. □

Remark 4. Note that Theorem 4 is valid if we only assume that $\mathcal{R}$ correctly answered (at least) a single DL_g query made by $\mathcal{A}_1$ when $\mathcal{R}$ succeeds in solving its n-DL challenge. This condition is a relaxation of the "regular" notion.

5 Some Applications to the Public-Key Cryptography

Here, we derive some cryptographic consequences from the above separation results (mainly Theorem 1). All follow from the following reasoning: if a cryptographic primitive is equivalent to an n-P problem with $n > 1$, then our results show that the security of this primitive cannot rely on the hardness of solving P (aka 0-P) by using classical black-box reductions[10]. Below, we consider cryptographic algorithms which have been proven secure under a "one-more problem" and summarize equivalence results.

5.1 Chaum's Blind Signature

Chaum's RSA-based blind signature [10] originally motivates the introduction of "one-more problems" (see [2]). In this scheme, the public key is (N, e) and the signer's private key is d (with, $ed = 1 \mod \phi(N)$ and the factorization of N is unknown). The signature of a message

[10] We can derive a similar conclusion for sophisticated reductions which do not use "forking-Lemma" like techniques. Since we are not aware of such reductions in the literature, we do not expand on this.

M is $x = \mathsf{RSA}_{N,e}^{-1}(H(M)) = H(M)^d \mod N$, where $H : \{0,1\}^* \to \mathbf{Z}_N$ is a hash function. The blind signature protocol allows a user to get the signature of a message without revealing it to the signer. To do so, the user picks $r \leftarrow_U \mathbf{Z}_N^*$ and sends $\bar{M} = r^e \cdot H(M) \mod N$ to the signer; the signer computes $\bar{x} = \mathsf{RSA}_{N,e}^{-1}(\bar{M}) = \bar{M}^d \mod N$ and returns $\bar{x}$ to the user, who extracts $x = \bar{x} \cdot r^{-1} \mod N$. In their paper, Bellare *et al.* [2] defined the notion of one-more RSA problems[11] and prove there exists a reduction from n-RSA to the one-more unforgeability of the blind signature in the random oracle model. Briefly speaking, the one-more unforgeability means that no efficient algorithm can produce $n+1$ valid message-signature pairs, after at most n interactions with the signer (remind that in such interaction, the signer does not see the actual message, he just extracts e-th modular roots).

The other direction is fairly simple. One key point is that the forger sees the randomness used in the signature; its "signing" oracle is actually an "RSA-inversion" oracle. We will not go into the details here, just give the rough idea. Assume we have an algorithm $\mathcal{A}$ that solves the n-RSA problem. Building a one-more forger against the Chaum's blind signature scheme is easy: just launch algorithm $\mathcal{A}$, answer its queries using the *"signing"* oracle (an e-th root extractor), and use its output to produce a forgery.

Now assume that the scheme can be proven secure under the standard RSA assumption, using a classical black-box reduction $\mathcal{R}$. Then, from a one-more RSA solver $\mathcal{A}$, we can construct a forger as above. And applying $\mathcal{R}$ to this forger would lead to an efficient RSA-inverter: in other words, we would have inverted RSA starting from algorithm $\mathcal{A}$. But this would contradict our Theorem 1 above. Thus, the unforgeability of Chaum blind signatures cannot be (black-box) based on the standard RSA problem.

5.2 Blind BLS Signature

In 2003, Boldyreva [6] proposed variants of the BLS signature [7], whose security holds in GDH groups [14] (groups in which CDH is hard, but deciding if a 4-tuple is a Diffie-Hellman one can be efficiently decided). The blind signature described in [6] was proven secure (in the random oracle model) under the one-more CDH problem. It considers a cyclic group $\mathbf{G}$ of prime order q generated by g and a bilinear pairing $e : \mathbf{G} \times \mathbf{G} \to \mathbf{G}'$ to a group $\mathbf{G}'$ of order q. The secret key is an element $x \leftarrow_U \mathbf{Z}_q$ and public key is $y = g^x$. The BLS signature σ of a message M is given by $H(M)^x$, where $H : \{0,1\}^* \to \mathbf{G}$ is a hash function (modeled by a random oracle). The verification consists in checking whether $e(H(M), y) = e(\sigma, g)$ holds, i.e., we check that $(g, y, H(M), \sigma)$ is a correct DDH-tuple. The blinding signing procedure consists in picking $r \in \mathbf{Z}_q$ and sending $\bar{M} = H(M) \cdot g^r$ to the signer who computes $\bar{\sigma} = \bar{M}^x$. The signature is finally obtained by computing $\sigma = \bar{\sigma} \cdot y^{-r}$.

The security model is the same as for Chaum's blind signature except that the forger has access to an oracle $(\cdot)^x$ which computes the scalar exponentiation in $\mathbf{G}$ on the query with factor x. The one-more CDH problem consists in receiving $n+1$ random elements $h_0, \ldots, h_n \leftarrow_U \mathbf{G}$ and in returning $y_0, \ldots, y_n$ such that $y_i = h_i^x$, while asking at most n queries to the oracle $(\cdot)^x$.

One can show that n-CDH is equivalent to the one-more unforgeability of the blind BLS. Namely, one feeds the n-CDH solver with $h_i := H(m_i)$ for $i = 0, \ldots, n$ with any chosen m_i's and returns the same output as the solver's one. In addition, the oracle $(\cdot)^x$ of the n-CDH-solver is trivially simulated using the same oracle as in the unforgeability game. The other direction was

[11] As noted in [2], these problems can be hard only if factoring does not reduce to RSA inversion.

proved by Boldyreva[12]. As for Chaum's blind signature, one deduces from Theorem 1 that one cannot get a black-box reduction from CDH problem to the unforgeability of blind BLS.

6 Conclusion

We presented rigorous arguments that a "one-more" problem n-P maybe not as hard as the corresponding 0-P when P is self-random reducible and efficiently verifiable. This class of problems include RSA inversion problem, computational Diffie-Hellman problem in the pairing context, and discrete logarithm problem. As main cryptographic consequences, we showed that the security of some blind signatures may hardly rely on standard assumption such as the RSA inversion problem or computational Diffie-Hellman problem. Furthermore, we showed that an equivalence result between the security of a primitive and the hardness of an n-P problem rules out the existence of a black-box reduction from 0-P to the security notion. Finally, our results also show that relying the security of a cryptographic primitive on a "one-more" problem n-P clearly does not give any guarantee that the security of the primitive can be relied on the corresponding 0-P problem, i.e., to a standard computational problem.

Acknowledgments. We would like to thank Mihir Bellare, Daniel Brown, and Daniele Micciancio for very interesting discussions and valuable comments on this work.

References

1. B. Barak, *How to Go Beyond the Black-Box Simulation Barrier.*, Proceedings of the 44th IEEE Symposium on Foundations of Computer Science (FOCS 2003) (M. Naor, ed.), IEEE Computer Society, 2001, pp. 106–115.
2. M. Bellare, C. Namprempre, D. Pointcheval, and M. Semanko, *The One-More-RSA-Inversion Problems and the Security of Chaum's Blind Signature Scheme.*, J. Cryptology **16** (2003), no. 3, 185–215.
3. M. Bellare and G. Neven, *Transitive signatures: new schemes and proofs.*, IEEE Trans. Inf. Theory **51** (2005), no. 6, 2133–2151.
4. M. Bellare and A. Palacio, *GQ and Schnorr Identification Schemes: Proofs of Security against Impersonation under Active and Concurrent Attacks.*, Advances in Cryptology - Crypto 2002 (M. Yung, ed.), Lect. Notes Comput. Sci., vol. 2442, Springer, 2002, pp. 162–177.
5. M. Bellare and R. Sandhu, *The Security of Practical Two-Party RSA Signature Schemes*, Tech. Report 2001/060, IACR eprint, 2001.
6. A. Boldyreva, *Threshold Signatures, Multisignatures and Blind Signatures Based on the Gap-Diffie-Hellman-Group Signature Scheme.*, 6th International Workshop on Practice and Theory in Public Key Cryptography, PKC 2003 (Y. Desmedt, ed.), Lect. Notes Comput. Sci., vol. 2567, Springer, 2003, pp. 31–46.
7. D. Boneh, B. Lynn, and H. Shacham, *Short Signatures from the Weil Pairing.*, J. Cryptology **17** (2004), no. 4, 297–319.
8. D. R. L. Brown, *Irreducibility to the One-More Evaluation Problems: More May Be Less.*, Tech. Report 2007/435, IACR eprint, 2007.
9. D. R. L. Brown and R. P. Gallant, *The Static Diffie-Hellman Problem.*, Tech. Report 2004/306, IACR eprint, 2004.
10. D. Chaum, *Blind Signatures for Untraceable Payments.*, Advances in Cryptology - Crypto'82 (D. Chaum, R. L. Rivest, and A. T. Sherman, eds.), Plenum Press, New York, 1983, pp. 199–203.
11. D. Chaum and H. van Antwerpen, *Undeniable Signatures.*, Advances in Cryptology - Crypto'89 (G. Brassard, ed.), Lect. Notes Comput. Sci., vol. 435, Springer, 1990, pp. 212–216.
12. L. C. Guillou and J.-J. Quisquater, *A Practical Zero-Knowledge Protocol Fitted to Security Microprocessors Minimizing both Transmission and Memory.*, Advances in Cryptology - Eurocrypt'88 (C. G. Günther, ed.), Lect. Notes Comput. Sci., vol. 330, Springer, 1988, pp. 123–128.

[12] To be more precise, she proved the security of this signature under a variant called one-more chosen-target CDH. A one-more chosen-target problem is like the variant presented in this article except that the solver receives m instances (with $m > n + 1$) and solves $n + 1$ instance of his choice with n oracle accesses. This variant is closer to the unforgeability notion, since a forger can make more than n hash evaluations. Bellare *et al.* showed that both variants are equivalent in the case of RSA and the discrete logarithm. We can apply the same technique to show that this also holds for CDH.

13. S. Micali and R. L. Rivest, *Transitive Signature Schemes.*, Topics in Cryptology - CT-RSA 2002 (B. Preneel, ed.), Lect. Notes Comput. Sci., vol. 2271, Springer, 2002, pp. 236–243.
14. T. Okamoto and D. Pointcheval, *The Gap-Problems: a New Class of Problems for the Security of Cryptographic Schemes.*, 4th International Workshop on Practice and Theory in Public Key Cryptography, PKC 2001 (K. Kim, ed.), Lect. Notes Comput. Sci., vol. 1992, Springer, 2001, pp. 104–118.
15. P. Paillier and D. Vergnaud, *Discrete-Log Based Signatures May Not Be Equivalent to Discrete-Log.*, Advances in Cryptology - Asiacrypt 2005 (B. Roy, ed.), Lect. Notes Comput. Sci., vol. 3788, Springer, 2005, pp. 1–20.
16. D. Pointcheval and J. Stern, *Security Arguments for Digital Signatures and Blind Signatures.*, J. Cryptology **13** (2000), no. 3, 361–396.
17. C. P. Schnorr, *Efficient signature generation by smart cards.*, J. Cryptology **4** (1991), no. 3, 161–174.
18. M. Tompa and H. Wol, *Random Self-Reducibility and Zero Knowledge Interactive Proofs of Possession of Information.*, Proceedings of the 28th IEEE Symposium on Foundations of Computer Science (FOCS 1987), IEEE Computer Society, 1987, pp. 472–482.