Black-Box Separations for One-More (Static) CDH and Its Generalization[⋆]

Jiang Zhang[1], Zhenfeng Zhang[⋆⋆1], Yu Chen[2], Yanfei Guo[1], and Zongyang Zhang[3]

[1] Trusted Computing and Information Assurance Laboratory,
State Key Laboratory of Computer Science,
Institute of Software, Chinese Academy of Sciences, P.R. China
[2] State Key Laboratory of Information Security,
Institute of Information Engineering, Chinese Academy of Sciences, P.R. China
[3] National Institute of Advanced Industrial Science and Technology (AIST), Japan
jiangzhang09@gmail.com, zfzhang@tca.iscas.ac.cn,
cycosmic@gmail.com, guoyanfei@tca.iscas.ac.cn,
zongyang.zhang@gmail.com

Abstract. As one-more problems are widely used in both proving and analyzing the security of various cryptographic schemes, it is of fundamental importance to investigate the hardness of the one-more problems themselves. Bresson *et al.* (CT-RSA '08) first showed that it is difficult to rely the hardness of some one-more problems on the hardness of their "regular" ones. Pass (STOC '11) then gave a stronger black-box separation showing that the hardness of some one-more problems cannot be based on standard assumptions using black-box reductions. However, since previous works only deal with one-more problems whose solution can be efficiently checked, the relation between the hardness of the one-more (static) CDH problem over non-bilinear groups and other hard problems is still unclear. In this work, we give the first impossibility results showing that black-box reductions cannot be used to base the hardness of the one-more (static) CDH problem (over groups where the DDH problem is still hard) on any standard hardness assumption. Furthermore, we also extend the impossibility results to a class of generalized "one-more" problems, which not only subsume/strengthen many existing separations for traditional one-more problems, but also give new separations for many other interesting "one-more" problems.

1 Introduction

The first one-more problem, n-RSA, was introduced by Bellare *et al.* [4] for proving the security of the Chaum's RSA-based blind signature scheme [17]. Formally, the n-RSA problem asks an algorithm to invert the RSA-function at $n+1$ random points

[⋆] Jiang Zhang, Zhenfeng Zhang and Yanfei Guo are sponsored by the National Basic Research Program of China under Grant No. 2013CB338003, and the National Natural Science Foundation of China (NSFC) under Grant No. 61170278, 91118006. Yu Chen is sponsored by NSFC under Grant No. 61303257, and the Strategic Priority Research Program of CAS under Grant No. XDA06010701. Zongyang Zhang is an International Research Fellow of JSPS and his work is in part supported by NSFC under grant No. 61303201.

[⋆⋆] Corresponding author.

with at most n calls to an RSA-inversion oracle. In particular, it is the regular RSA problem when $n = 0$. Similar to the n-RSA problem, Bellare *et al.* [4] also suggested that a class of one-more inversion problems can be formulated for any family of *one-way* functions, which basically asks an algorithm to invert a one-way function at some random points with a bounded number of queries (*i.e.*, less than the number of given points) to an inversion oracle. The hardness assumption on this class of problems aims to capture the intuition that an algorithm cannot gain advantage from the inversion oracle other than making "trivial" use of it. Instantiated with the discrete logarithm (DL) function, the one-more DL problem, n-DL, was given in [5]. Later, Boldyreva [8] constructed a secure blind signature scheme based on the hardness of the one-more static CDH problem (or chosen-target CDH problem [8]). Roughly, the one-more static CDH problem (n-sDH for short) defined in [8] is to solve $n + 1$ static Diffie-Hellman (sDH) instances [12] with at most n queries to an sDH solution oracle.[4]

The one-more inversion problems not only make it possible to find security proofs for many classical cryptographic constructions [7,6,2,3,13,19], but are also used to illustrate the impossibilities of proving the security of some other cryptographic schemes such as [36,27,42,25], even though the original intention of introducing them is to "prove security". Due to plenty of fruitful results, many cryptographic researchers also put effort into studying the hardness of one-more inversion problems, "*to see how they relate to other problems and to what extent we can believe in them as assumptions*" [5]. In CRYPTO '08, Garg *et al.* [27] raised it as a major open question "*to understand relationship between the DL problem and the n-DL problem*". Earlier in the same year, Bresson *et al.* [10] and Brown [11] presented the first evidence that one-more inversion problems seem to be weaker than their "regular" ones. Specifically, they showed that the hardness of some $(n + 1)$-P problem (*e.g.,* $(n + 1)$-RSA) cannot be based on the hardness of "its own" n-P problem (*e.g.,* n-RSA) using some "restricted" black-box reductions. Later, Pass [37] showed that black-box reductions cannot be used to base the hardness of a special kind of one-more inversion problems (what was called one-more problems based on homomorphic certified permutations [37]) on any standard assumption. However, all the above impossibility results explicitly require the underlying problem P to be efficiently verifiable, and thus cannot apply to the n-sDH problem over groups where the DDH problem is hard.[5]

1.1 Our Results

In this paper, we present the first impossibility results showing that black-box reductions cannot be used to base the hardness of the n-sDH problem (over groups where the DDH problem is hard) on any standard hardness assumption $\mathcal{C}$. In particular, the assumption $\mathcal{C}$ itself can be n'-sDH problem for smaller n'. Technically, we construct a meta-reduction (*i.e.,* "reduction against the reduction" [9,24,28,21]) which directly

[4] The notation "n-CDH" is used in [10] instead of "n-sDH". We use "n-sDH" because the n-CDH problem can be defined directly based on the CDH problem (instead of the sDH problem), and our separation results apply to such n-CDH problems as well.

[5] We note that the computational complexity between the n-sDH problem and the DL problem over specific groups has also been studied in the literature [31,34,29].

breaks the assumption $\mathcal{C}$ by interacting with any black-box reduction from $\mathcal{C}$ to the n-sDH problem. Due to the nice feature of meta-reductions [28,37], our results also apply to black-box reductions that may make non-black-box use of the assumption $\mathcal{C}$. Then, we extend our proof techniques to obtain separation results for a class of more generalized "one-more" problems, which not only subsume/strengthen many existing separations for the traditional one-more problems (*e.g.*, n-RSA, n-DL, and the unforgeability of blind signatures), but also give new separations for many other interesting "one-more" problems.

Throughout the paper, the security of a cryptographic problem P is defined via a game between a challenger $\mathcal{C}(\mathrm{P})$ and an adversary $\mathcal{A}$. In particular, the challenger $\mathcal{C}(\mathrm{P})$ provides the adversary $\mathcal{A}$ with a stateful (and possibly unbounded) oracle Orcl. The actual behavior of the oracle Orcl is determined by the description of P. We say that a problem P is *non-interactive* if its oracle $\mathsf{Orcl} = \perp$. Sometimes, we will slightly abuse the notation, and use $\mathcal{C}$ to denote both the problem and its associated challenger. A hard cryptographic problem $\mathcal{C}$ is said to be t-*round*, if the number of the messages exchanged between the Orcl and the adversary $\mathcal{A}$ is at most t (which might be a priori bounded polynomial in the security parameter).

Impossibility for One-More Static CDH Problems. By a nice observation that Cash *et al.*'s *trapdoor test* (for the twin Diffie-Hellman problems [16]) allows some form of verification for the CDH problem, we give the first black-box separations for the n-sDH problem over general groups by carefully injecting the trapdoor test technique into our meta-reduction. The difficultly of this approach lies in the fact that the trapdoor test does not really allow us to publicly and efficiently check the validity of any single CDH tuple (since it can only check whether or not two carefully prepared tuples are both CDH tuples by using some private coins. Especially, if one of the two tuples is not a CDH tuple, it cannot determine which one is not). We overcome this difficulty by designing an unbounded adversary $\mathcal{A}$ with "delay verifications" and a meta-reduction $\mathcal{M}$ with "dynamic decisions" on whether or not to use, and how to use the trapdoor test in simulating $\mathcal{A}$ to the reduction $\mathcal{R}$. Formally, we have the following theorem.

Theorem 1. *There is no black-box reduction $\mathcal{R}$ for basing the hardness of the n-sDH problem on any $t(k)$-round hard problem $\mathcal{C}$ (or else $\mathcal{C}$ could be solved efficiently), where k is the security parameter and $n = 2 \cdot \omega(k + t + 1)$.*

Since we consider very general black-box reductions, the requirement on $n = 2 \cdot \omega(k+t+1)$ seems a bit loose. However, if one would like to consider a class of restricted black-box reductions—single-instance reductions [26,25], a tighter separation result for $n \geq 2(t + 1)$ can be achieved.

Black-Box Separations for Generalized "One-More" Problems. A natural extension of the traditional one-more problem is defined by "relaxing" the requirement on the oracle. Formally, we consider a class of generalized "one-more" problems, where each problem is associated with two non-interactive subproblems P_1 and P_2. Here, we do not require $\mathrm{P}_1 = \mathrm{P}_2$. For any integer $n \geq 0$, we denote n-$(\mathrm{P}_1, \mathrm{P}_2)$ as the problem which asks an algorithm to solve $n + 1$ random P_1 instances with at most n calls to a

P_2 oracle (*e.g.*, $\mathrm{P}_1 =$ CDH, $\mathrm{P}_2 =$ DL). Obviously, the traditional one-more problem is a special case of our generalization with $\mathrm{P}_1 = \mathrm{P}_2$. In particular, we briefly denote it as n-P_1 if $\mathrm{P}_1 = \mathrm{P}_2$, which coincides with traditional notations (*e.g.*, n-DL). Now, we consider a class of n-$(\mathrm{P}_1, \mathrm{P}_2)$ problems that there exists an efficient reduction T from P_1 to P_2 (*e.g.*, from CDH to DL) with the following two properties:

– T solves one P_1 instance by using at most γ (non-adaptive) queries to a P_2 oracle, where γ is a constant;
– T always correctly solves its input P_1 instance after obtaining γ correct responses from the P_2 oracle, and outputs "$\perp$" if one of the γ responses is incorrect with overwhelming probability (we remark that this condition implicity require that T can somehow verify the correctness of all the γ responses as a whole, but it is not required to determine which one of the responses is incorrect, *e.g.,* we have $\gamma = 2$ for the n-sDH problem).

Then, similar separation results also hold for such class of n-$(\mathrm{P}_1, \mathrm{P}_2)$ problems if, in addition, P_1 has unique solution [23,37] and P_2 is randomly self-reducible [1]. Note that here we still do not explicitly require P_2 to be efficiently verifiable as for the sDH problem. Formally, we have the following theorem.

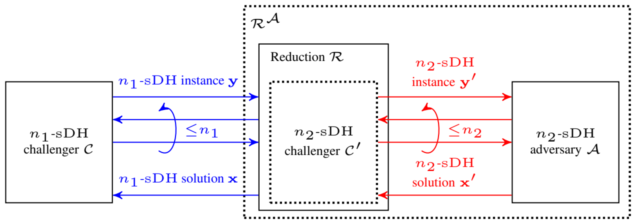
Theorem 2. *If there exists an efficient reduction* T *from* P_1 *to* P_2 *with the above two properties,* P_1 *has unique solution and* P_2 *is randomly self-reducible, then there is no black-box reduction* $\mathcal{R}$ *for basing the hardness of the* n-$(\mathrm{P}_1, \mathrm{P}_2)$ *problem on any* $t(k)$*-round hard problem* $\mathcal{C}$ *(or else* $\mathcal{C}$ *could be solved efficiently), where* k *is the security parameter and* $n = \gamma \cdot \omega(k + t + 1)$.

Like the discussion after Theorem 1, if only single-instance reductions are considered, we can get a tighter separation result for $n \geq \gamma \cdot (t+1)$. Note that for the traditional n-DL, n-RSA and n-sDH over gap Diffie-Hellman groups [35] where $\mathrm{P}_1 = \mathrm{P}_2$, there is a natural reduction T with $\gamma = 1$. The above theorem indeed subsumes/strengthens existing separations for those problems in the literature [10,37]. Since our generalized "one-more" problem also captures the "one-more unforgeability" of blind signatures and many other interesting "one-more" problems (*e.g.*, n-(CDH, DL)), our results actually give a broad separation for some of those problems. For instance, one can directly define the one-more CDH problem, n-CDH, based on the CDH problem instead of the sDH problem, and our impossibility results apply to the n-CDH problem.

1.2 The Idea Behind Our Impossibility Results

To better illustrate our techniques, we start from a simple *vanilla* reduction $\mathcal{R}$ (depicted in Fig.1) from the traditional one-more problem n_1-sDH to n_2-sDH (*i.e.*, $\mathrm{P}_1 = \mathrm{P}_2 =$ sDH) for integers $n_2 > n_1 \geq 0$, which only runs a single instance of the n_2-sDH adversary $\mathcal{A}$ without rewinding [10,26]. Concretely, upon receiving the challenge n_1-sDH instance $\mathbf{y}$ from $\mathcal{C}$, the reduction $\mathcal{R}$ invokes a *single instance* of $\mathcal{A}$ by simulating an n_2-sDH challenger $\mathcal{C}'$, and tries to find the solution $\mathbf{x}$ of $\mathbf{y}$ by interacting with $\mathcal{A}$.

Intuitively, if the adversary $\mathcal{A}$ can somehow see the n_1-sDH instance input $\mathbf{y}$ of $\mathcal{R}$, it can directly solve them by using its own n_2 sDH queries to $\mathcal{R}$. This intuition is actually



$\mathcal{R}^{\mathcal{A}}$
Reduction $\mathcal{R}$
n_1-sDH challenger $\mathcal{C}$
n_1-sDH instance $\mathbf{y}$
$\leq n_1$
n_1-sDH solution $\mathbf{x}$
n_2-sDH challenger $\mathcal{C}'$
n_2-sDH instance $\mathbf{y}'$
$\leq n_2$
n_2-sDH solution $\mathbf{x}'$
n_2-sDH adversary $\mathcal{A}$

Fig. 1. A single-instance reduction $\mathcal{R}$ from n_1-sDH to n_2-sDH, where $n_2 > n_1 \geq 0$.

the basic idea of [10], which constructed a meta-reduction $\mathcal{M}$ that runs $\mathcal{R}$ with its own n_1-sDH instance, and simulates an n_2-sDH adversary $\mathcal{A}$ to $\mathcal{R}$. However, this approach has two technical barriers. First, the sDH queries that $\mathcal{A}$ is allowed to make might not be in the same group or have the same public parameters as the input $\mathbf{y}$ of $\mathcal{R}$. Second, $\mathcal{R}$ can cheat $\mathcal{M}$ by returning random group elements if DDH is hard in the considered group. To get around these two barriers, the authors [10] put on additional restrictions on $\mathcal{R}$ (*e.g.*, algebraic or parameter-invariant [10]), and considered the n-sDH problem over gap Diffie-Hellman groups [35] where the DDH problem is easy.

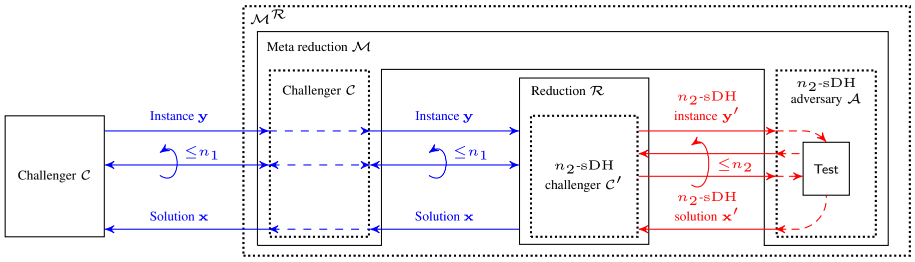


Fig. 2. Our meta-reduction $\mathcal{M}$ against $\mathcal{R}$ from n_1-round hard problem $\mathcal{C}$ to n_2-sDH problem, where $n_2 > n_1 \geq 0$.

We remove the restrictions on $\mathcal{R}$ by using rewinding technique, which allows our meta-reduction $\mathcal{M}$ (depicted in Fig.2) to directly solve $\mathbf{y}'$ (*i.e.*, to make $n_2 + 1$ sDH queries to $\mathcal{R}$ for all the instances in $\mathbf{y}'$), and outputs whatever $\mathcal{R}$ returns as the solution to its own challenge instance $\mathbf{y}$. In this case, $\mathcal{M}$ actually does not care about what $\mathbf{y}$ is. Without loss of generality, we simply denote $\mathbf{y}$ as an instance of any n_1-round hard problem $\mathcal{C}$. The requirement $n_2 \geq n_1 + 1$ is still needed to ensure that there is at least one query that $\mathcal{R}$ answers without having interactions with its own challenger $\mathcal{C}$. In other words, we have to guarantee that there is at least one chance that $\mathcal{M}$ can safely rewind $\mathcal{R}$ without affecting the external interactions between $\mathcal{R}$ and $\mathcal{C}$.

To deal with the n-sDH problem over general groups where DDH is hard, we use a good observation on the remarkable **trapdoor test** algorithm (denoted by Test hereafter) introduced by Cash *et al.* [16] for twin Diffie-Hellman problems. Informally, given an element $y \in \mathbb{G}$, the algorithm outputs another uniformly distributed element

$z \in \mathbb{G}$ together with some private coins r. Then, for any elements $h, f_1, f_2 \in \mathbb{G}$, the Test algorithm can use r to determine whether or not both (y, h, f_1) and (z, h, f_2) are CDH tuples with overwhelming probability. Briefly, the Test algorithm cannot *publicly* check the validity of any single CDH tuple, but it can determine whether or not two carefully prepared tuples are both CDH tuples (by using the private coins r). Especially, if one of them is not a CDH tuple, the algorithm cannot determine which one is not. This "inability" of the Test algorithm poses an obstacle when we try to use it in our meta-reduction $\mathcal{M}$ to prevent the reduction $\mathcal{R}$ from cheating, since $\mathcal{R}$ might also notice this. To overcome this obstacle (*i.e.,* to hide the use of the Test algorithm from $\mathcal{R}$), we first present an unbound adversary $\mathcal{A}$ (against the n_2-sDH problem) with "delay verifications" such that it delays the verification of the odd-numbered response to the point immediately after obtaining the next even-numbered response, and checks the validity of the responses "two by two". Then, we construct a meta-reduction $\mathcal{M}$ that carefully tracks all the "private coins"used by the Test algorithm, and (statistically) hides the two sDH queries needed by the Test algorithm into its own sDH queries to the reduction $\mathcal{R}$. This requires the meta-reduction $\mathcal{M}$ to make "dynamic decisions" on whether or not to use, and how to use the Test algorithm in preparing each sDH query to $\mathcal{R}$.

To finally establish the separation results for general black-box reductions (*i.e.*, without any additional restrictions on $\mathcal{R}$), we have to deal with two technical issues. First, $\mathcal{R}$ might rewind the (unbounded) adversary $\mathcal{A}$ to obtain extra advantage. This is circumvented by designing a "magical" adversary $\mathcal{A}$ such that it performs "deterministically" [37,25]. Second, $\mathcal{R}$ might invoke many instances of $\mathcal{A}$, a naive rewinding of $\mathcal{R}$ will result in an exponential running-time due to "nested rewindings" [22,20]. We deal with this problem by making use of recursive rewinding techniques [41,37], which allow our meta-reduction $\mathcal{M}$ to cleverly find a "safe rewinding chance" and cancel a rewinding when it has to do too much work [38,20,15].

In all, we finally separate the n-sDH problem from any other (priori bounded) polynomial round hard problems. The impossibility results for generalized "one-more" problems can be analogously obtained if there is a reduction T for the underlying problem which can play a similar role as the Test algorithm for the n-sDH problem.

1.3 Related Work, Comparison and Discussion

Relation to Bresson et al. [10]. In CT-RSA '08, Bresson *et al.* [10] studied the relations between the traditional one-more inversion problems and their "regular" ones, and showed that the hardness of the traditional n-P problem cannot be based on the $(n-1)$-P problem using some "restricted" (*e.g.*, algebraic or parameter-invariant [10]) black-box reductions. Concretely, they showed that a class of restricted black-box reductions cannot be used to base the hardness of n-DL, n-RSA, and n-sDH over gap Diffie-Hellman groups [35], on their corresponding one-more problems with less oracle queries, *e.g.,* $(n-1)$-DL, $(n-1)$-RSA, $(n-1)$-sDH over gap Diffie-Hellman groups. As discussed in Section 1.2, the restrictions on $\mathcal{R}$ in their results seem unavoidable since their meta-reductions heavily rely on the "direct" connections between the challenge n_1-P instance input of the reduction $\mathcal{R}$ and the n_2-P instance output by $\mathcal{R}$, where $n_2 \geq n_1 + 1$. This is also the reason why separations of the one-more problems from other hard problems cannot be derived. For comparison, our meta-reduction makes use of the "rewinding" technique and the "inner" connections between the instances in the n-P problem and its associated oracle queries, which allows us to separate the n-P problem from any other (priori bounded) polynomial round hard problems. In particular, we rule out the existence of general black-box reductions (*i.e.,* without imposing any other restrictions on $\mathcal{R}$) for sufficiently large n.

Relation to Pass [37]. In STOC '11, Pass [37] presented a broad separation result showing that the security of constant-round, public-coin, (generalized) computational special-sound arguments for unique witness relations cannot be based on any standard assumption. Pass's results apply to many well-known cryptographic problems such as the traditional one-more inversion problems and the security of the two-move *unique* blind signatures (*i.e.*, each message has a unique signature for a fixed verification key). In particular, Pass showed that the hardness of n-DL and n-RSA cannot be based on any t-round standard assumption using black-box reductions if $n = \omega(k + 2t + 1)$, where k is the security parameter.[6] The use of recursive rewinding in this work is inspired by Pass [37], which makes our meta-reduction have an analogous structure to that in [37] and a similar requirement on $n = \gamma \cdot \omega(k + t + 1)$.

Since Pass's proof [37] crucially relies on the fact that the underlying problem is publicly and efficiently verifiable, their results cannot apply to the n-sDH problem over general groups. Especially, since the Test algorithm [16] does not really allow us to publicly and efficiently check the validity of any single CDH tuple (as discussed in Section 1.2), one cannot trivially use Pass's separation results and the Test algorithm to obtain our impossibility results in a "black-box fashion". Actually, our results are achieved by carefully combining many known techniques in the literature (*e.g.,* [38,20,15,37,25]) and new techniques such as *"delay verifications"* and *"dynamic decisions"* in our meta-reduction. We also extend our proof techniques to a class of generalized "one-more" problems, which allows us to obtain separation results for many interesting "one-more" problems such as the security of a class of two-move blind signatures.

Other Related Work. Fischlin and Schröder [26] showed that a class of "restricted" black-box reductions cannot be used to prove the security of three-move blind signatures based on any hard *non-interactive* problem. Katz *et al.* [33] showed that there is no black-box construction of blind signatures from one-way permutations. Both results overlap with ours in the context of two-move blind signatures, and we strengthen the separation result (in this context) by proving that the security of a class of two-move blind signatures (including non-black-box constructions) cannot be based on any polynomial round hard problem using black-box reductions.

2 Preliminaries

Let $|x|$ denote the length of a string x, and $|S|$ denote the size of a set S. Denote $x\|y$ as the bit concatenation of two strings $x, y \in \{0, 1\}^*$. We use the notation $\leftarrow$ to indicate the output of some algorithm, and the notation $\leftarrow_r$ to denote randomly choosing elements

[6] The factor "2" before t is because a knowledge extractor, whose behavior might dependent on the distribution of its its input transcripts, is used [37]. Please refer to [37] for details.

from some distribution (or the uniform distribution over some finite set). For example, if $\mathcal{A}$ is a probabilistic algorithm, $z \leftarrow \mathcal{A}(x, y, \ldots; r)$ means that the output of algorithm $\mathcal{A}$ with inputs $x, y, \ldots$, and randomness r is z. When r is unspecified, we mean running $\mathcal{A}$ with uniformly random coins. We say that $\mathcal{A}$ is a PPT algorithm if it runs in probabilistic polynomial-time.

The natural security parameter throughout the paper is k, and all other quantities are implicit functions of k. We use standard notation O and ω to classify the growth of functions. We say that a function $f(k)$ is negligible if for any constant $c > 0$, there exists an N such that $f(k) < 1/k^c$ for all $k > N$.

2.1 Cryptographic Problems

In this subsection, we recall several definitions of cryptographic problems.

Definition 1 (Cryptographic Problem). *A cryptographic problem* $\mathrm{P} = (\mathsf{PGen}, \mathsf{IGen}, \mathsf{Orcl}, \mathsf{Vrfy})$ *consists of four algorithms:*

- *The parameter generator* PGen *takes as input the security parameter* 1^k*, outputs a public parameter* $param$*, which specifies the instance space* $\mathcal{Y}$ *and the solution space* $\mathcal{X}$*, in brief,* $param \leftarrow \mathsf{PGen}(1^k)$.
- *The instance generator* IGen *takes as input the public parameter* $param$*, outputs an instance* $y \in \mathcal{Y}$*, i.e.,* $y \leftarrow \mathsf{IGen}(param)$.
- *The stateful oracle algorithm* $\mathsf{Orcl}(param, \cdot)$ *takes as input a query* $q \in \{0,1\}^*$*, returns a response* r *for* q *or a special symbol* $\perp$ *if* q *is an invalid query.*
- *The deterministic verification algorithm* Vrfy *takes as inputs the public parameter, an instance* $y \in \mathcal{Y}$ *and a candidate solution* $x \in \mathcal{X}$*, returns* 1 *if and only if* x *is a correct solution of* y*, else returns* 0.

Throughout this paper, we implicitly assume it is easy to check whether an element y (resp., x) is in $\mathcal{Y}$ (resp., $\mathcal{X}$). We say that a cryptographic problem $\mathrm{P} = (\mathsf{PGen}, \mathsf{IGen}, \mathsf{Orcl}, \mathsf{Vrfy})$ is efficiently verifiable if Vrfy is running in polynomial-time. When $\mathsf{Orcl} = \perp$, we say that P is a *non-interactive* problem, and denote $\mathrm{P} = (\mathsf{PGen}, \mathsf{IGen}, \mathsf{Vrfy})$ in brief. Usually, the two algorithms PGen and IGen are also required to be PPT algorithms, but we do not explicitly need the requirements in this paper.

Definition 2 (Hard Cryptographic Problem). *Let* k *be the security parameter. A cryptographic problem* $\mathrm{P} = (\mathsf{PGen}, \mathsf{IGen}, \mathsf{Orcl}, \mathsf{Vrfy})$ *is said to be hard with respect to a threshold function* $\mu(k)$*, if for all PPT algorithm* $\mathcal{A}$*, the advantage of* $\mathcal{A}$ *in the security game with the challenger* $\mathcal{C}$ *(who provides inputs to* $\mathcal{A}$ *and answers* $\mathcal{A}$*'s oracle queries) is negligible in* k*:*

$$\begin{aligned}\mathrm{Adv}_{\mathrm{P},\mathcal{A}}(1^k) = \Pr[param \leftarrow \mathsf{PGen}(1^k), y \leftarrow \mathsf{IGen}(param);\\ x \leftarrow \mathcal{A}^{\mathsf{Orcl}(param,\cdot)}(y) : \mathsf{Vrfy}(param, y, x) = 1] - \mu(k).\end{aligned}$$

Usually, $\mu(k) = 0$ is used for computational problems (*e.g.*, DL, CDH, n-DL), and $\mu(k) = 1/2$ is used for decisional problems (*e.g.*, DDH, DBDH).

As in [37], we also put on restrictions on the number of interactions between $\mathcal{A}$ and the oracle in the game, and a hard cryptographic problem is said to be t-*round* if the number of the messages exchanged (via oracle queries) between $\mathcal{C}$ and $\mathcal{A}$ is at most t.

2.2 Black-Box Reductions

A black-box reduction $\mathcal{R}$ from a cryptographic problem P_1 to another cryptographic problem P_2 is a PPT oracle algorithm such that $\mathcal{R}^{\mathcal{A}}$ solves P_1 whenever $\mathcal{A}$ solves P_2 with non-negligible probability. In addition to normally communicating with $\mathcal{A}$, $\mathcal{R}$ also has many powers such as restarting or "rewinding" $\mathcal{A}$. Black-box reductions often take advantage of these features. For example, $\mathcal{R}$ could make use of "rewind" to get out of a "bad condition" by first rewinding $\mathcal{A}$ to a previous state and then trying some different choices [14,20].

3 Black-Box Separations of One-More Static CDH Problems

In this section, we present the first separation results for one-more static CDH problems over general groups where the DDH problem may still be hard.

Let k be a security parameter, $\mathbb{G}$ be a group of prime order $q \geq 2^{2k}$, and g be a generator of $\mathbb{G}$. For any two group elements $A = g^a, B = g^b$, we denote $\mathrm{CDH}(A, B) = g^{ab} = A^b = B^a$. Recall that the n-sDH problem is asking an algorithm to solve $n+1$ static Diffie-Hellman (sDH) instances [12] with at most n queries to an oracle that solves sDH problems. Formally, given parameters $param = (k, \mathbb{G}, q, g, h)$ and $n+1$ group elements $\mathbf{y} = (y_1, \ldots, y_{n+1})$, the algorithm is asked to output $\mathbf{x} = (x_1, \ldots, x_{n+1})$ such that $x_i = \mathrm{CDH}(y_i, h)$ for all $i = \{1, \ldots, n+1\}$, with at most n queries to an oracle $\mathrm{sDH}(\cdot, h)$.

Now, we recall the trapdoor test algorithm (with compatible notations) in the following lemma, please refer to [16] for details.

Lemma 1 (Trapdoor Test [16]). *Let $\mathbb{G}$ be a cyclic group of prime order q, generated by $g \in \mathbb{G}$. Let $y \in \mathbb{G}$ be an element of $\mathbb{G}$, and $r, r' \in \mathbb{Z}_q$ are uniformly distributed over $\mathbb{Z}_q$. Define $z = g^{r'}/y^r$. Then, for any elements $h, f_1, f_2 \in \mathbb{G}$, we have: 1) z is uniformly distributed over $\mathbb{G}$; 2) y and z are independent, then the probability that the truth value of*

$$f_1^r f_2 = h^{r'} \tag{1}$$

does not agree with the truth value of

$$f_1 = \mathrm{CDH}(y, h) \wedge f_2 = \mathrm{CDH}(z, h) \tag{2}$$

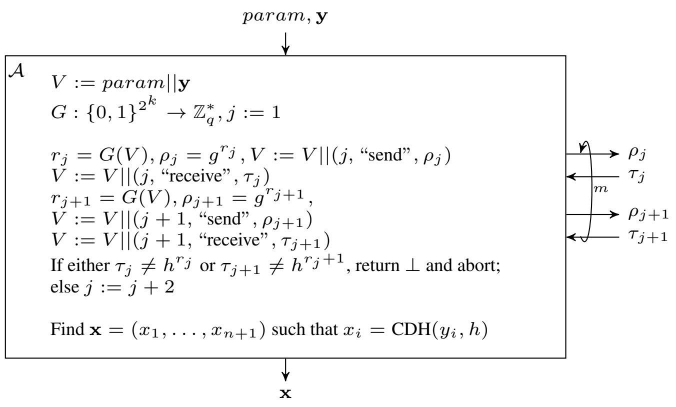
is at most $1/q$; moreover, if (2) holds, then (1) certainly holds.

Note that the probability in the above lemma is over the random choices of r and r', and is independent of the choices of h, f_1 and f_2. This fact is very important for our separation results. For simplicity, we denote the PPT algorithm in the above lemma as Test, and assume that it works in two phases. In the initial phase, it takes the parameter $param = (k, \mathbb{G}, q, g, h)$, a group element $y \in \mathbb{G}$, and randomness $r, r' \in \mathbb{Z}_q$ as inputs, returns a group element $z = g^{r'}/y^r \in \mathbb{G}$, *i.e.*, $z \leftarrow \mathsf{Test}(\mathbf{init}, param, y; r, r')$. In the finish phase, it takes another two elements (f_1, f_2) as inputs, returns a bit $e \in \{0, 1\}$ that indicates whether the condition $f_1^r f_2 = h^{r'}$ holds, in brief, $e \leftarrow \mathsf{Test}(\mathbf{finish}, param, y, z, f_1, f_2; r, r')$. Besides, we say that the algorithm

Test ***fails*** if it returns 1, but at least one of the two tuples (y, h, f_1) and (z, h, f_2) is not a CDH tuple. By Lemma 1, the probability that the Test algorithm *fails* is at most $1/q$ (which is negligible in the security parameter k), where the probability is over random choices of $r, r' \leftarrow_r \mathbb{Z}_q$.

3.1 An Unbounded Adversary

In this subsection, we present an unbounded adversary $\mathcal{A}$ (depicted in Fig. 3) that solves the n-sDH problem for $n \geq 2$. Informally, the adversary $\mathcal{A}$ only makes random queries to its oracle, and delays the verification of the odd-numbered response to the point immediately after obtaining the next even-numbered response from its oracle (*i.e.*, it verifies the responses from its oracle two by two). Besides, $\mathcal{A}$ always makes even number of queries to its oracle, and omits the last query if n is odd. As in [37,25], a random function G is used by $\mathcal{A}$ to generate its inner random coins with its own view as input.



```
V := param||y
G : {0,1}^{2^k} → Z_q^*, j := 1

r_j = G(V), ρ_j = g^{r_j}, V := V||(j, "send", ρ_j)
V := V||(j, "receive", τ_j)
r_{j+1} = G(V), ρ_{j+1} = g^{r_{j+1}},
V := V||(j + 1, "send", ρ_{j+1})
V := V||(j + 1, "receive", τ_{j+1})
If either τ_j ≠ h^{r_j} or τ_{j+1} ≠ h^{r_{j+1}}, return ⊥ and abort;
else j := j + 2

Find x = (x_1, ..., x_{n+1}) such that x_i = CDH(y_i, h)
```

Fig. 3. The adversary $\mathcal{A}$ uses a random function G to generate all its internal randomness.

Description of $\mathcal{A}$. Given the public parameter $param = (k, \mathbb{G}, q, g, h)$ with security parameter k, and an n-sDH instance $\mathbf{y} = (y_1, \ldots, y_{n+1})$, $\mathcal{A}$ is asked to compute the solution of $\mathbf{y}$, with at most n queries to its sDH$(\cdot, h)$ oracle. Let $V = param\|\mathbf{y}$, *i.e.*, the view of $\mathcal{A}$. Then, the adversary $\mathcal{A}$ randomly chooses a function G from all functions $\{0,1\}^{2^k} \rightarrow \mathbb{Z}_q^*$, and let $m = \lfloor \frac{n}{2} \rfloor$.

For $j = \{1, 3, \ldots, 2m-1\}$, $\mathcal{A}$ computes $r_j \leftarrow G(V)$, and $\rho_j = g^{r_j}$. Then, it updates the view $V := V\|(j, \text{"send"}, \rho_j)$ and sends an external sDH query with ρ_j. After obtaining the solution τ_j of ρ_j, $\mathcal{A}$ updates the view $V := V\|(j, \text{"receive"}, \tau_j)$. Then, it makes another sDH query ρ_{j+1} in the same way as ρ_j by using randomness $r_{j+1} \leftarrow G(V)$, and obtains τ_{j+1} from its sDH oracle. If $\tau_j \neq h^{r_j}$ or $\tau_{j+1} \neq h^{r_{j+1}}$, $\mathcal{A}$ returns $\perp$ and aborts; else let $j := j + 2$.

After completing $2m$ sDH queries without abort, $\mathcal{A}$ computes $x_i = \mathrm{CDH}(y_i, h)$ for all $i \in \{1, \ldots, n+1\}$ by brute-force search (which is not necessarily done in polynomial time), and outputs $\mathbf{x} = (x_1, \ldots, x_{n+1})$ as the solution to its own challenge $\mathbf{y} = (y_1, \ldots, y_{n+1})$.

The use of the random function G brings us two benefits. First, the distribution of each sDH query of $\mathcal{A}$ is uniformly random over $\mathbb{G}$, which allows our meta-reduction $\mathcal{M}$ to (statistically) hide its real "intention". Second, it is hard for the reduction $\mathcal{R}$ to obtain (significant) advantage by rewinding $\mathcal{A}$, since $\mathcal{A}$ always generates the same sDH query at the same view, and a random sDH query at a freshly different view.

Besides, the way that the adversary $\mathcal{A}$ verifies the responses from its oracle two by two is very crucial for our separation results, which allows our meta-reduction $\mathcal{M}$ to "delay" the verification of the odd-numbered sDH response from the reduction $\mathcal{R}$ (actually, it cannot efficiently do the verification if DDH is hard), and to embed two sDH queries needed by the Test algorithm into two consecutive sDH queries to $\mathcal{R}$.

3.2 The Meta-Reduction for One-More Static CDH Problems

Let $\mathcal{R}$ be a black-box reduction from some hard problem $\mathcal{C}$ to the n-sDH problem, namely, $\mathcal{R}^{\mathcal{A}}$ can solve the hard problem $\mathcal{C}$ with non-negligible probability. Basically, the reduction $\mathcal{R}$ is given access to the deterministic "next-messages" function of the adversary $\mathcal{A}$, *i.e.,* the function, given a partial view $(x, m_1, \ldots, m_j)$ of $\mathcal{A}$, computes the next message output by $\mathcal{A}$, where x is the input (which includes the randomness that $\mathcal{R}$ chooses for $\mathcal{A}$), and $(m_1, \ldots, m_j)$ are the transcripts of the interactions between $\mathcal{R}$ and $\mathcal{A}(x)$. As in [37], we use the following (standard) assumptions about $\mathcal{R}$ to simplify our presentation:

- $\mathcal{R}$ never feeds the same partial view twice to its oracle $\mathcal{A}$;
- When $\mathcal{R}$ feeds a partial view q to its oracle $\mathcal{A}$, the transcripts contained in q are generated in previous interactions between $\mathcal{R}$ and $\mathcal{A}$.

Both of the two assumptions are without loss of generality, since the oracle $\mathcal{A}$ is a "deterministic function" and we can easily modify $\mathcal{R}$ to satisfy the two conditions. Besides, in order to better illustrate how our meta-reduction works, and how the Test algorithm is injected, we denote instance $\mathcal{A}_i$ as a copy of $\mathcal{A}$ on a specified input x (*i.e.,* $\mathcal{A}(x)$). In particular, a unique positive integer i is used for each different input x (recall that x includes the randomness that $\mathcal{R}$ chooses for $\mathcal{A}$).

In the concurrent zero-knowledge protocols [41,40,38], the term "**slot**" usually denotes the point where a rewinding is possible. We adapt this notion to our case, which is slightly different from that in [37]. Intuitively, a **slot** in our context always opens with an odd-numbered query and ends with the response of the immediately followed even-numbered query. Formally, let V_{R} be the view of $\mathcal{R}$, which includes all the messages sent and received by $\mathcal{R}$ in the interactions with both the adversary $\mathcal{A}$ and the challenger $\mathcal{C}$. A partial view of V_{R} is a prefix of V_{R}. For some integer $j \geq 1$, we say that a **slot** s_i of $\mathcal{A}_i$ opens at a partial view $V_o^{s_i}$ if $\mathcal{A}_i$ sends the $(2j-1)$-th query q_1 to $\mathcal{R}$ immediately after $V_o^{s_i}$ (note that q_1 must be a "fresh" query of $\mathcal{A}_i$ by our simplification assumption), and we say that a **slot** s_i of $\mathcal{A}_i$ closes at a partial view $V_c^{s_i}$ if $\mathcal{A}_i$ receives a response r_2 to the $2j$-th query q_2 from $\mathcal{R}$ at the end of $V_c^{s_i}$. In particular, we denote the partial view $V_o^{s_i}$ as the **opening** of s_i, and the partial view $V_c^{s_i}$ as the **closing** of s_i. Since a **slot** may open without being closed (*i.e.*, $\mathcal{R}$ may never respond to some query q_i sent by $\mathcal{A}_i$), we uniquely identify a **slot** s_i by its **opening**. We also denote a particular closed **slot** s_i as a pair of its **opening** and **closing** for convenience, *i.e.*, $s_i = (V_o^{s_i}, V_c^{s_i})$.

Definition 3 (Good Slot). *Let $M = M(k)$ be the maximum number of the messages sent and received by the reduction $\mathcal{R}$ on input the security parameter k. For any positive integer d, we say that a closed **slot** $s_i = (V_o^{s_i}, V_c^{s_i})$ (of $\mathcal{A}_i$) is d-**good** if the following two conditions hold:*

- *Between the time that s_i opens and the time that s_i closes, $\mathcal{R}$ does not interact with the challenger $\mathcal{C}$;*
- *Between the time that s_i opens and the time that s_i closes, the number of **slots** that open is at most $\frac{M}{n^d}$.*

Informally, the definition of a ***good*** **slot** brings us three benefits. First, since the reduction $\mathcal{R}$ does not interact with the challenger $\mathcal{C}$ during the **slot**, rewinding the reduction $\mathcal{R}$ to the opening of a ***good*** **slot** will not affect the interactions between $\mathcal{R}$ and $\mathcal{C}$. Second, each **slot** always contains two consecutive (fresh) queries, which allows our meta-reduction $\mathcal{M}$ to embed two sDH queries in a **slot**. Third, each **slot** always opens with an odd-numbered query, $\mathcal{M}$ can delay the verification of the response to the first query, and simultaneously check the validity of the two responses from $\mathcal{R}$ (by using the Test algorithm).

We make use of recursive rewinding techniques in [41,38,20,15] to rule out general black-box reductions (*i.e.*, without any additional restrictions on the reductions), which was recently introduced by Pass [37] to give impossibility results for a class of witness-hiding protocols. Basically, we provide the meta-reduction $\mathcal{M}$ with many rewinding chances (*i.e.*, **slots**), and let $\mathcal{M}$ be always "on the lookout" for ***good*** **slots** to rewind $\mathcal{R}$ such that the rewinding will not "blow up" the running time of $\mathcal{M}$ too much (*e.g.*, running in an exponential time). Formally, we have the following theorem.

Theorem 3 (Black-Box Separations for One-More Static CDH Problems). *For any integers $t(k)$ and $n = 2 \cdot \omega(k + t + 1)$, there is no black-box reduction $\mathcal{R}$ for basing the hardness of the n-sDH problem on any t-round hard problem $\mathcal{C}$ (or else $\mathcal{C}$ could be solved efficiently), where k is the security parameter.*[7]

Proof. We now proceed to give the description of our meta-reduction $\mathcal{M}$, which recursively calls a procedure SOLVE to rewind the reduction $\mathcal{R}$. In the simulation of the adversary $\mathcal{A}$ to $\mathcal{R}$, the meta-reduction $\mathcal{M}$ has to maintain three technical tables $\mathcal{L}_1$, $\mathcal{L}_2$ and $\mathcal{L}_3$. Informally, the first table $\mathcal{L}_1$ is used to record all the n-sDH instances (that $\mathcal{M}$ has to solve) and the corresponding solutions (that have been found by $\mathcal{M}$). The

[7] Basically, the constant '2' can be safely absorbed by the asymptotic function '$\omega(\cdot)$'. We leave it here mainly because it is introduced by our "delay verification" proof technique, which is different from previous results, *e.g.*, [37]. We also hope this can give a clear relation between the results in Theorem 3 and its generalized results in Theorem 4.

other two tables are used to successfully inject the Test algorithm into the simulation, where table $\mathcal{L}_2$ records the randomness used by the Test algorithm (for each target sDH instance) and table $\mathcal{L}_3$ records the randomness used to re-randomize each sDH query made by the Test algorithm. For functionality, table $\mathcal{L}_1$ is used in a "call by reference" method, namely, the changes of $\mathcal{L}_1$ at the $(d+1)$-th recursive level will be reflected at the d-th recursive level. But this is not required for table $\mathcal{L}_2$ and table $\mathcal{L}_3$. Formally,

– Table $\mathcal{L}_1$ consists of four tuples $(i, param_i \| \mathbf{y}_i, b_i, \mathbf{x}_i)$ and indicates that 1) $\mathcal{R}$ invokes the i-th instance of $\mathcal{A}$ with parameter $param_i$ and $n+1$ sDH instances $\mathbf{y}_i = (y_{i,1}, \cdots, y_{i,n+1})$ as inputs; 2) $\mathcal{M}$ has found the first b_i solutions of $\mathbf{y}_i$, and stored them in $\mathbf{x}_i$ (*i.e.*, $|\mathbf{x}_i| = b_i$). Thus, $\mathcal{L}_1 = \perp$ when $\mathcal{R}$ is invoked (by $\mathcal{M}$), and $(b_i = 0, \mathbf{x}_i = \perp)$ when a new instance of $\mathcal{A}$ is invoked (by $\mathcal{R}$).
– Table $\mathcal{L}_2$ consists of six tuples $(i, t_i, y_{i,t_i}, r_{i,t_i}, r'_{i,t_i}, z_{i,t_i})$ and indicates that the i-th instance of $\mathcal{A}$ prepares an "aided" element z_{i,t_i} with randomness r_{i,t_i} and r'_{i,t_i}, and aims to find the solution of the t_i-th sDH instance y_{i,t_i}, *i.e.*, $z_{i,t_i} \leftarrow \mathsf{Test}(\mathbf{init}, param_i, y_{i,t_i}; r_{i,t_i}, r'_{i,t_i})$.
– Table $\mathcal{L}_3$ consists of five tuples $(i, t_i, \delta, u_i, \rho_i)$ that indicates the actual query made by $\mathcal{A}_i$, where $\delta \in \{0,1\}$ and $u_i \in \mathbb{Z}_q^*$. If $\delta = 0$, it means that $\mathcal{A}_i$ sends an odd-numbered sDH query with $\rho_i = y_{i,t_i}^{u_i}$. Else if $\delta = 1$, it means that $\mathcal{A}_i$ sends an even-numbered sDH query with $\rho_i = z_{i,t_i}^{u_i}$.

Description of $\mathcal{M}$. Given a security parameter k, a description of $\mathcal{C}$ instance, let $V_{\mathrm{R}} = k \| \mathcal{C}$, $\mathcal{M}$ runs $\mathcal{R}$ with $\mathcal{C}$, and executes $\mathsf{SOLVE}(1^k, 0, V_{\mathrm{R}}, \perp, \perp, \perp)$ to simulate the unbounded adversary $\mathcal{A}$. Whenever $\mathcal{R}$ outputs the solution of $\mathcal{C}$, $\mathcal{M}$ outputs it and halts. Let $c = \lceil \log_k M \rceil$, for each level $0 \le d \le c$, procedure SOLVE works as follows:

$\mathsf{SOLVE}(1^k, d, V, \mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3)$:
On input a security parameter k, the current recursive level d, the partial view V of $\mathcal{R}$ and three tables $\mathcal{L}_1$, $\mathcal{L}_2$ and $\mathcal{L}_3$, let $v = V$ and repeat the following steps:

1. If $d = 0$ and $\mathcal{R}$ makes external interactions with $\mathcal{C}$, simply relay the messages between $\mathcal{R}$ and $\mathcal{C}$, and update the view v.
2. If $d > 0$ and $\mathcal{R}$ attempts to make external interactions with $\mathcal{C}$ or the number of **slots** opened after V in v exceeds $\frac{M}{k^d}$, cancel the current recursive level and return. (Note that this case happens if and only if the probability that V becomes the **opening** of a d-***good*** slot is non-negligible. Thus, the algorithm can simply cancel the current recursive rewinding at level d and return to the $(d-1)$-th level whenever the slot starting from V cannot be d-***good*** anymore.)
3. If $\mathcal{R}$ feeds $\mathcal{A}$ with a partial view which only contains the input message, denote $(param, \mathbf{y})$ as the corresponding n-sDH instance.[8] Choose an unused smallest positive integer i for this instance $\mathcal{A}_i$ of $\mathcal{A}$, and add $(i, param \| \mathbf{y}, 0, \perp)$ to $\mathcal{L}_1$. Finally, update the view v.
4. If $\mathcal{R}$ feeds $\mathcal{A}_i$ with a partial view which contains a response τ_i to a previous sDH instance query ρ_i from $\mathcal{A}_i$, update the view v and proceed as follows:
   – If ρ_i is the $(2j-1)$-th query of $\mathcal{A}_i$ for some $j \ge 1$, continue;

[8] Recall that the input message contains the n-sDH instance and the randomness that $\mathcal{R}$ chooses for $\mathcal{A}$. Besides, $\mathcal{R}$ never fed the same input to $\mathcal{A}$ before by our simplification assumptions.

- If ρ_i is the $2j$-th query of $\mathcal{A}_i$ for some $j \geq 1$, let (ρ'_i, τ'_i) and (ρ_i, τ_i) be the last two consecutive pairs of query and response of $\mathcal{A}_i$, retrieve $(i, t_i, 0, u'_i, \rho'_i)$ and $(i, t_i, 1, u_i, \rho_i)$ from $\mathcal{L}_3$, and $(i, t_i, y_{i,t_i}, r_{i,t_i}, r'_{i,t_i}, z_{i,t_i})$ from $\mathcal{L}_2$, such that $\rho'_i = y_{i,t_i}^{u'_i}$ and $\rho_i = z_{i,t_i}^{u_i}$, and compute $f_1 = (\tau'_i)^{1/u'_i}, f_2 = (\tau_i)^{1/u_i}$. Then, if $\mathsf{Test}(\textbf{finish}, param_i, y_{i,t_i}, z_{i,t_i}, f_1, f_2; r_{i,t_i}, r'_{i,t_i}) = 0$, abort the simulation of $\mathcal{A}_i$. Otherwise, retrieve $(i, param_i \| \mathbf{y}_i, b_i, \mathbf{x}_i)$ from $\mathcal{L}_1$. If $t_i = b_i + 1$, update the tuple $(i, param_i \| \mathbf{y}_i, b_i, \mathbf{x}_i)$ in $\mathcal{L}_1$ by letting $b_i = t_i$ and $\mathbf{x}_i = \mathbf{x}_i \| f_1$ (or else, we must have $t_i \leq b_i$, which means the solution of y_{i,t_i} has been found previously). Finally, let $s_i = (V_o^{s_i}, v)$ be the **slot** with **closing** v, and distinguish the following cases:
  - If s_i opened before V (*i.e.*, s_i did not open at the current recursive level), continue;
  - Else if $V_o^{s_i} = V$ (*i.e.*, s_i opened at V) and $d > 0$, end the current recursive level and return.[9]
  - Else if $V_o^{s_i} \neq V$ (*i.e.*, s_i opened after V) and s_i is a $(d+1)$-***good*** **slot**, repeat the procedure $\mathsf{SOLVE}(1^k, d{+}1, V_o^{s_i}, \mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3)$ until $b_i = n{+}1$.
  - Otherwise, continue;

5. If $\mathcal{R}$ is expecting a message from $\mathcal{A}_i$, retrieve $(i, b_i, param_i \| \mathbf{y}_i,\ \mathbf{x}_i)$ from $\mathcal{L}_1$ (recall that $param_i = (k, \mathbb{G}, p, g, h)$), and proceed as follows:
   - If $\mathcal{A}_i$ has completed $2m = 2 \cdot \lfloor \frac{n}{2} \rfloor$ sDH queries (*i.e.*, $\mathcal{A}_i$ has to send the solution of $\mathbf{y}_i$ to $\mathcal{R}$), send $\mathbf{x}_i$ to $\mathcal{R}$ if $b_i = n + 1$ (*i.e.*, the solution $\mathbf{x}_i$ of $\mathbf{y}_i$ has been found), else output "**fail**" and halt.
   - Else if it is the $(2j-1)$-th query for some $j \geq 1$, let $t_i = b_i + 1$ if $b_i < n + 1$, else $t_i = n + 1$. If there is no tuple $(i, t_i, y_{i,t_i}, *, *, *)$ in $\mathcal{L}_2$, choose $r_{i,t_i}, r'_{i,t_i} \leftarrow_r \mathbb{Z}_q$, compute $z_{i,t_i} \leftarrow \mathsf{Test}(\textbf{init}, param_i, y_{i,t_i}; r_{i,t_i}, r'_{i,t_i})$, and add the tuple $(i, t_i, y_{i,t_i}, r_{i,t_i}, r'_{i,t_i}, z_{i,t_i})$ to $\mathcal{L}_2$. Then, choose $u_i \leftarrow_r \mathbb{Z}_q^*$, send $\rho_i = y_{i,t_i}^{u_i}$ to $\mathcal{R}$, and add the tuple $(i, t_i, 0, u_i, \rho_i)$ to $\mathcal{L}_3$.
   - Else if it is the $2j$-th query for some $j \geq 1$, let ρ_i be the $(2j-1)$-th query of $\mathcal{A}_i$, retrieve $(i, t_i, 0, u_i, \rho_i)$ and $(i, t_i, y_{i,t_i}, *, *, z_{i,t_i})$ from $\mathcal{L}_3$ and $\mathcal{L}_2$, respectively, such that $\rho_i = y_{i,t_i}^{u_i}$. Then, choose $u'_i \leftarrow_r \mathbb{Z}_q^*$, send $\rho'_i = z_{i,t_i}^{u'_i}$ to $\mathcal{R}$, and add the tuple $(i, t_i, 1, u'_i, \rho'_i)$ to $\mathcal{L}_3$.

   Finally, update the view v accordingly.
6. If $\mathcal{R}$ returns the solution of $\mathcal{C}$, output the solution and halt.

Remark 1. By our simplification assumptions, $\mathcal{R}$ never feeds the same partial view twice to $\mathcal{A}$. This allows $\mathcal{M}$ to simply prepare each sDH query (on behalf of $\mathcal{A}$) by using freshly chosen randomness, since our unbounded adversary $\mathcal{A}$ (in Section 3.1) uses a random function G to deterministically generate its "inner" randomness by using the interaction transcripts with $\mathcal{R}$ as input.

To show that our meta-reduction $\mathcal{M}$ can efficiently solve problem $\mathcal{C}$ (with non-negligible probability), we only have to show that 1) $\mathcal{M}$ perfectly simulates the un-

[9] This means that the rewinding at the partial view V (at level d) is successful, and the algorithm returns to the $(d-1)$-th level to check whether it has found all the solutions of $\mathbf{y}_i$ for instance $\mathcal{A}_i$ (*i.e.,* to check whether $b_i = n + 1$).

bounded adversary $\mathcal{A}$ except with negligible probability; 2) $\mathcal{M}$ runs in expected polynomial time. We prove the two claims in the following two lemmas.

Lemma 2. *$\mathcal{M}$ perfectly simulates the unbounded adversary $\mathcal{A}$ except with negligible probability.*

Proof. Since $\mathcal{M}$ always randomizes its sDH queries to $\mathcal{R}$ (on behalf of $\mathcal{A}$) by using uniformly chosen randomness from $\mathbb{Z}_q^*$ (*i.e.,* u_i or u_i' in step 5), the distribution of these queries is essentially the same to that of $\mathcal{A}$ (which always makes random sDH queries to $\mathcal{R}$). We finish the proof of this lemma by proving the following two cases: 1) If $\mathcal{R}$ cheats in the interactions (*i.e.,* by returning a false answer to an sDH query), $\mathcal{M}$ will reject it in the same way as $\mathcal{A}$ except with negligible probability; 2) If $\mathcal{R}$ does not cheat, $\mathcal{M}$ can find the correct solution $\mathbf{x}_i$ of the n-sDH input $\mathbf{y}_i$ of instance $\mathcal{A}_i$ from table $\mathcal{L}_1$ (*i.e.,* it will not output "**fail**" in step 5).

For the first case, since both the meta-reduction $\mathcal{M}$ and the unbounded adversary $\mathcal{A}$ do not immediately check the validity of any odd-numbered response, the simulation of $\mathcal{A}$ after receiving an odd-numbered response from $\mathcal{R}$ (*i.e.,* the first case in step 4) is essentially the same as that of the real adversary $\mathcal{A}$. After receiving an even-numbered query, the unbounded adversary $\mathcal{A}$ will check the validity of the previous two consecutive responses, and will always return $\perp$ and abort if one of the previous two responses is invalid. As for the meta-reduction $\mathcal{M}$, it always embeds the first query of the Test algorithm in an odd-numbered query, and the second query of the Test algorithm in the immediately followed even-numbered query, and then checks the two consecutive responses by using the Test algorithm. Obviously, if the Test algorithm does not *fail*, $\mathcal{M}$ can perfectly detect whether $\mathcal{R}$ cheats or not, which is essentially the same as $\mathcal{A}$. Now, we show that the probability that the Test algorithm *fails* at least one time is negligible. Recall that the total number of the messages sent and received by $\mathcal{R}$ is bounded by $M(k)$, for any partial view v of $\mathcal{R}$, there are at most $M(k)$ messages in v. Thus, $\mathcal{M}$ has to run the Test algorithm at most $(n+1)M(k)$ times (since $\mathcal{R}$ can invoke at most $M(k)$ instances of $\mathcal{A}$, and for each instance $\mathcal{A}_i$, $\mathcal{M}$ has to run the Test algorithm at most $(n+1)$ times). Since $\mathcal{M}$ always independently and randomly chooses the randomness for each time running of the Test algorithm (in step 5), the probability that the Test algorithm will *fail* at least one time is at most $\frac{(n+1)M(k)}{q}$ by Lemma 1, which is negligible in k.

For the second case, let v be the partial view of $\mathcal{R}$, at which $\mathcal{R}$ expects the simulated instance $\mathcal{A}_i$ to provide the solution of its input n-sDH $\mathbf{y}_i$. Since the unbounded adversary $\mathcal{A}$ always sequentially makes $2m = 2 \cdot \lfloor \frac{n}{2} \rfloor$ sDH queries, there must exist at least m **slots** of $\mathcal{A}_i$ in the partial view v. Recall that the total number of recursive levels is bounded by $c = \lceil \log_k M \rceil$ (which is a constant if $\mathcal{R}$ runs in polynomial time), there must exist some recursive level d such that there are at least $\frac{m}{c+1}$ **slots** of $\mathcal{A}_i$ in the partial view v (by the pigeon hole principle). Since $n = 2 \cdot \omega(k+t+1)$, we have $\frac{m}{c+1} \geq k+t+1$ for sufficiently large k. Hereafter, we always assume that there is at least $k+t+1$ **slots** of $\mathcal{A}_i$ at level d. By the definition of SOLVE, the total number of **slots** opened at level d is at most $\frac{M}{k^d}$ (this obviously holds at level $d=0$). Thus, there are at least $t+1$ **slots** of $\mathcal{A}_i$ that contain at most $\frac{M}{k^{d+1}}$ **slots** (or else there are at least $k+1$ **slots** of $\mathcal{A}_i$ that contain more than $\frac{M}{k^{d+1}}$ **slots**, which makes the total number of **slots** opened at level d exceeds $\frac{M}{k^d}$, and the recursive rewinding at level d will be canceled). Since $\mathcal{C}$ is t-round, there is at least one **slot** of $\mathcal{A}_i$ (in those $t+1$ **slots**) during which $\mathcal{R}$ has no interactions with the challenger $\mathcal{C}$. Obviously, such a **slot** is $(d+1)$-***good***, thus will be rewound. In all, we have proven that for each complete instance $\mathcal{A}_i$, there must exist at least one ***good*** **slot** that would be rewound. Since $\mathcal{M}$ will end the recursive calls to SOLVE at the opening of a $(d+1)$-***good*** **slot** in step 4 until $b_i = n+1$, $\mathcal{M}$ can always find the solution $\mathbf{x}_i$ of the input n-sDH instance $\mathbf{y}_i$ of $\mathcal{A}_i$ from table $\mathcal{L}_1$ before it has to send the solution of $\mathbf{y}_i$ to $\mathcal{R}$ (*i.e.*, $\mathcal{M}$ will not output "**fail**" in step 5). Since $\mathcal{R}$ does not cheat in this case, $\mathbf{x}_i$ must be the correct (unique) solution of $\mathbf{y}_i$. This completes the proof of Lemma 2.

Lemma 3. *$\mathcal{M}$ runs in expected polynomial time.*

Proof. We estimate the maximum running time of $\mathcal{M}$ that never outputs "**fail**" and halts in the simulation. Since the total number of the messages sent and received by $\mathcal{R}$ is bounded by $M(k)$, there are at most $M(k)$ ***good*** **slots** that might be rewound at each recursive level. Let v be a partial view (at level d) immediately after which a **slot** s opens. Then, let δ be the probability that s becomes $(d+1)$-***good***, where the probability is over all the randomness used in the interactions between $\mathcal{M}$ and $\mathcal{R}$ after v. Now, assume that we arrived at a partial view v' such that $s = (v, v')$ is $(d+1)$-***good***, then if $\mathcal{M}$ rewinds $\mathcal{R}$ to v (i.e., rewinding $\mathcal{R}$ to the opening of s at recursive level $d+1$), the probability that the slot becomes $(d+1)$-***good*** is essentially close to δ. This is because the behavior of the simulated adversary (by $\mathcal{M}$) after rewinding $\mathcal{R}$ to v (i.e., at level $d+1$) is almost identically distributed to that starting from v at level d. Since δ is non-negligible (or else it is unlikely to arrive at the partial view v'), $\mathcal{M}$ can expect to obtain a $(d+1)$-***good*** **slot** with probability negligibly close to 1 by rewinding $\mathcal{R}$ polynomial times at v. Let $p(k)$ be such a polynomial. Then, $\mathcal{M}$ can expect to find the solution of some $y_{i,j}$ for $\mathcal{A}_i$ with probability negligibly close to 1 by rewinding $\mathcal{R}$ at most $p(k)$ times. Thus, for each of those ***good*** **slots**, the expected number of rewindings is bounded by $(n+1)p(k)$ (recall that $\mathcal{M}$ has to find the solutions of $(n+1)$ sDH instances for each $\mathcal{A}_i$), and the total number of rewindings at each recursive level is expected at most $(n+1)p(k)M(k)$. By an induction computation, we have the expected number of the messages sent and received by $\mathcal{M}$ is bounded by $((n+1)p(k)M(k))^{c+1}$, which is a polynomial in k if $M(k)$ is a polynomial in k. This completes the proof of Lemma 3.

4 More Black-Box Separations for One-More Problems

Let $\mathrm{P}_1 = (\mathsf{PGen}, \mathsf{IGen}_1, \mathsf{Vrfy}_1)$ and $\mathrm{P}_2 = (\mathsf{PGen}, \mathsf{IGen}_2, \mathsf{Vrfy}_2)$ be two non-interactive cryptographic problems with the same parameter generator PGen. Now, we give the definition of the generalized "one-more" problems.

Definition 4 (Generalized "One-More" Problems). *For any integer $n \geq 0$, the generalized "one-more" problem n-$(\mathrm{P}_1, \mathrm{P}_2) = (\mathsf{PGen}, \mathsf{IGen}, \mathsf{Orcl}, \mathsf{Vrfy})$ associated with two subproblems P_1 and P_2 is defined as follows:*

– *The parameter generator $\mathsf{PGen}(1^k)$ algorithm returns the public parameter $param$ for P_1 and P_2.*

- *The instance generator* $\mathsf{IGen}(param)$ *independently runs the* $\mathsf{IGen}_1(param)$ *algorithm* $n+1$ *times to generate* $(n+1)$ *random* P_1 *instances* $\{y_i\}_{i\in\{1,\dots,n+1\}}$*, and returns an instance* $\mathbf{y} = (y_1, \dots, y_{n+1})$*.*
- *The stateful oracle algorithm* Orcl *takes as inputs a public parameter* $param$*, and a* P_2 *instance* y*, returns the solution* x *of* y *or a special symbol* $\perp$ *if* y *is an invalid query. (If* P_2 *is not a unique solution problem, the oracle can return one of the candidate solutions in a predefined rule, e.g., the first one in lexicographic order.)*
- *The verification algorithm* Vrfy *takes the public parameter* $param$*, an instance* $\mathbf{y} = (y_1, \dots, y_{n+1})$*, and a candidate solution* $\mathbf{x} = (x_1, \dots, x_{n+1})$ *as inputs, returns 1 if and only if* $\mathsf{Vrfy}_1(param, y_i, x_i) = 1$ *for all* $i \in \{1, \dots, n+1\}$*.*

In particular, if $\mathrm{P}_1 = \mathrm{P}_2$*, we briefly denote* n-$(\mathrm{P}_1, \mathrm{P}_2)$ *as* n-P_1*. Besides, the* n-$(\mathrm{P}_1, \mathrm{P}_2)$ *problem is said to be hard if for any PPT adversary* $\mathcal{A}$*, the advantage of* $\mathcal{A}$ *in solving all the* $(n+1)$ *random* P_1 *instances in* $\mathbf{y}$ *with at most* n P_2 *queries to the oracle* Orcl *is negligible.*

This class of problems not only subsumes the traditional one-more problems (where $\mathrm{P}_1 = \mathrm{P}_2$, *e.g.*, n-RSA, n-DL), the unforgeability of two-move blind signatures (where it is likely $\mathrm{P}_1 \neq \mathrm{P}_2$) and so forth, but also encompasses many other interesting problems that have not been (well) studied in the literature. For example, to solve $n+1$ BDH instances by using n DL queries, *i.e.*, n-$(\mathrm{BDH}, \mathrm{DL})$ in our notation [18].

Our separation results in Theorem 3 can be generalized to the n-$(\mathrm{P}_1, \mathrm{P}_2)$ problem if there is a PPT reduction T which can be used to solve one P_1 instance by only using γ (non-adaptive) queries to a P_2 oracle, where γ can be any constant (*e.g.,* $\gamma = 2$ for the n-sDH problem). In particular, we assume that reduction T works in two phases: Given the public parameter $param$, a P_1 instance y, and a randomness r, it enters into the $\mathbf{query}$ phase and outputs a vector of P_2 instances $\mathbf{z} = (z_1, \dots, z_\gamma)$. In brief, $\mathbf{z} \leftarrow \mathsf{T}(\mathbf{query}, param, y; r)$. After being fed back with the solution vector $\mathbf{f} = (f_1, \dots, f_\gamma)$ of $\mathbf{z}$ where f_i is a candidate solution of z_i, T enters into the $\mathbf{finish}$ phase, and returns the solution x of y or a special symbol $\perp$. In brief, $x/\perp \leftarrow \mathsf{T}(\mathbf{finish}, param, y, \mathbf{z}, \mathbf{f}; r)$. Informally, we say that a reduction T is a "promise reduction" if it always tries to return a correct answer, otherwise returns $\perp$ to indicate that "some of its inputs are invalid".

Definition 5 (Promise Reduction). *Let* Ω_t *be the randomness space of* T*, we say that a reduction* T *from* P_1 *to* P_2 *is a promise reduction if it satisfies the following two properties:*

Efficient computability: *There is a PPT algorithm that computes* T*.*

Correctness-preserving: *Fixing the parameter* $(param, y)$*, for any* $\mathbf{z} \leftarrow \mathsf{T}(\mathbf{query}, param, y; r)$*, any candidate solutions* $\mathbf{f}$ *of* $\mathbf{z}$*, and* $x \leftarrow \mathsf{T}(\mathbf{finish}, param, y, \mathbf{z}, \mathbf{f}; r)$*, we have*

- *If* $\mathsf{Vrfy}_2(param, z_i, f_i) = 1$ *holds for all* $i \in \{1, \dots, \gamma\}$*, we have that* $x \neq \perp$ *and* $\mathsf{Vrfy}_1(param, y, x) = 1$ *hold;*
- *If there exists* $i \in \{1, \dots, \gamma\}$ *such that* $\mathsf{Vrfy}_2(param, z_i, f_i) = 0$*, then we have* $x = \perp$ *with overwhelming probability;*

where the probabilities are over the random choice of $r \leftarrow_r \Omega_t$*.*

Remark 2. The first requirement on the correctness-preserving property of T can be relaxed to "hold with non-negligible probability" if P_1 is efficiently verifiable (*i.e.*, Vrfy_1 is a PPT algorithm). Since we can repeat the reduction T polynomial times (at a cost of slightly increasing the running time of $\mathcal{M}$) to get a correct solution with probability negligibly close to 1. The strong requirement is used here for simplicity.

Theorem 4 (Black-Box Separation for Generalized "One-More" Problems). *For integer $n > 0$, let n-$(\mathrm{P}_1, \mathrm{P}_2)$ be defined as in Definition 4. If P_1 has unique solution, P_2 is randomly self-reducible and there is a promise reduction T from P_1 to P_2 with at most γ queries. Then, there is no black-box reduction $\mathcal{R}$ for basing the hardness of the n-$(\mathrm{P}_1, \mathrm{P}_2)$ problem on any $t(k)$-round hard problem $\mathcal{C}$ (or else $\mathcal{C}$ could be solved efficiently), where k is the security parameter and $n = \gamma \cdot \omega(k + t + 1)$.*

The proof is very similar to the proof of Theorem 3, we defer it to the full version.

Remark 3. The requirement on $n = \gamma \cdot \omega(k+t+1)$ is needed to successfully apply "recursive rewinding" [41,38,20,15] to rule out general black-box reductions. However, if one would like to consider restricted black-box reductions—single-instance reductions [26,25], a tighter separation result for $n \geq \gamma \cdot (t + 1)$ can be achieved.

Since our generalized "one-more" problems abstract many interesting problems, Theorem 4 actually gives a very broad impossibility result for a large class of problems. For the three traditional one-more inversion problems we have the following corollary.

Corollary 1. *There is no black-box reduction $\mathcal{R}$ for basing the hardness of n-DL, n-RSA, or n-sDH over gap Diffie-Hellman groups on any $t(k)$-round hard problem $\mathcal{C}$ (or else $\mathcal{C}$ could be solved efficiently), where k is the security parameter and $n = \omega(k + t + 1)$.*

Actually, our separation results naturally apply to the n-CDH problem, which can be directly defined based on CDH problems as in Definition 4 (*i.e.,* $\mathrm{P}_1 = \mathrm{P}_2 =$ CDH).

Corollary 2. *There is no black-box reduction $\mathcal{R}$ for basing the hardness of the n-CDH problem over general groups on any $t(k)$-round hard problem $\mathcal{C}$ (or else $\mathcal{C}$ could be solved efficiently), where k is the security parameter and $n = 2 \cdot \omega(k + t + 1)$.*

Since the one-more unforgeability of blind signatures can be treated as a standard generalized "one-more" problems, our separation results actually apply to a class of two-move blind signatures [17,8,30] that are statistically blinding [32,39] and allow statistical signature-derivation check [26]. We defer the details to the full version.

Corollary 3. *If a two-move (unique) blind signature* BS *is statistically blinding and allows statistical signature-derivation check, then there is no black-box reduction $\mathcal{R}$ for basing the one-more unforgeability of* BS *on any polynomially round hard problem $\mathcal{C}$ (or else $\mathcal{C}$ could be solved efficiently).*

Besides, our results also apply to many other interesting problems that may not have been (well) studied in the literature. For example, if P_2 is the DL problem, P_1 can be any other DL-based problems such as CDH, DDH and BDH.

Finally, we clarify that our impossibility result only rules out black-box reductions from other (standard) hard problems to this class of problems, it does not mean the problems in the class are easy to solve, or there are no non-black-box reductions basing the hardness of these problems on other (standard) hard problems. In fact, some of them might be very useful in proving or analyzing cryptographic constructions.

Acknowledgments. We thank Yi Deng, Phong Q. Nguyen, and the anonymous reviewers of CRYPTO 2014 and ASIACRYPT 2014 for their helpful comments and suggestions on earlier versions of our paper.

References

1. M. Abadi, J. Feigenbaum, and J. Kilian. On hiding information from an oracle. *Journal of Computer and System Sciences*, 39(1):21–50, 1989.
2. M. Bellare, C. Namprempre, and G. Neven. Security proofs for identity-based identification and signature schemes. In *EUROCRYPT*, pages 268–286, 2004.
3. M. Bellare, C. Namprempre, and G. Neven. Security proofs for identity-based identification and signature schemes. *Journal of Cryptology*, 22(1):1–61, 2009.
4. M. Bellare, C. Namprempre, D. Pointcheval, and M. Semanko. The power of RSA inversion oracles and the security of Chaum's RSA-based blind signature scheme. In *Financial Cryptography*, pages 309–328, 2001.
5. M. Bellare, C. Namprempre, D. Pointcheval, and M. Semanko. The one-more-RSA-inversion problems and the security of Chaum's blind signature scheme. *Journal of Cryptology*, 16(3):185–215, 2003.
6. M. Bellare and G. Neven. Transitive signatures: new schemes and proofs. *IEEE Transactions on Information Theory*, 51(6):2133–2151, 2005.
7. M. Bellare and A. Palacio. GQ and Schnorr identification schemes: Proofs of security against impersonation under active and concurrent attacks. In *CRYPTO*, pages 162–177, 2002.
8. A. Boldyreva. Threshold signatures, multisignatures and blind signatures based on the Gap-Diffie-Hellman-group signature scheme. In *PKC*, pages 31–46, 2003.
9. D. Boneh and R. Venkatesan. Breaking RSA may not be equivalent to factoring. In *EUROCRYPT*, pages 59–71, 1998.
10. E. Bresson, J. Monnerat, and D. Vergnaud. Separation results on the "one-more" computational problems. In *CT-RSA*, pages 71–87, 2008.
11. D. R. L. Brown. Irreducibility to the one-more evaluation problems: More may be less. Cryptology ePrint Archive, Report 2007/435, 2007.
12. D. R. L. Brown and R. P. Gallant. The static Diffie-Hellman problem. Cryptology ePrint Archive, Report 2004/306, 2004.
13. S. Canard, A. Gouget, and J. Traoré. Improvement of efficiency in (unconditional) anonymous transferable e-cash. In *Financial Cryptography*, pages 202–214, 2008.
14. R. Canetti, R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Adaptive security for threshold cryptosystems. In *CRYPTO*, pages 98–115, 1999.
15. R. Canetti, H. Lin, and R. Pass. Adaptive hardness and composable security in the plain model from standard assumptions. In *FOCS*, pages 541–550, 2010.
16. D. Cash, E. Kiltz, and V. Shoup. The twin Diffie-Hellman problem and applications. In *EUROCRYPT*, pages 127–145, 2008.
17. D. Chaum. Blind signatures for untraceable payments. In *CRYPTO*, pages 199–203, 1982.
18. Y. Chen, Q. Huang, and Z. Zhang. Sakai-ohgishi-kasahara non-interactive identity-based key exchange scheme, revisited. In *ACISP*, pages 274–289. 2014.

19. E. D. Cristofaro and G. Tsudik. Practical private set intersection protocols with linear complexity. In *Financial Cryptography*, pages 143–159, 2010.
20. Y. Deng, V. Goyal, and A. Sahai. Resolving the simultaneous resettability conjecture and a new non-black-box simulation strategy. In *FOCS*, pages 251–260, 2009.
21. Y. Dodis, I. Haitner, and A. Tentes. On the instantiability of hash-and-sign RSA signatures. In *TCC*, pages 112–132. 2012.
22. C. Dwork, M. Naor, and A. Sahai. Concurrent zero-knowledge. *Journal of the ACM*, 51(6):851–898, 2004.
23. D. Fiore and D. Schröder. Uniqueness is a different story: Impossibility of verifiable random functions from trapdoor permutations. In *TCC*, pages 636–653, 2012.
24. M. Fischlin. Black-box reductions and separations in cryptography. In *AFRICACRYPT*, pages 413–422. 2012.
25. M. Fischlin and N. Fleischhacker. Limitations of the meta-reduction technique: The case of Schnorr signatures. In *EUROCRYPT*, pages 444–460, 2013.
26. M. Fischlin and D. Schröder. On the impossibility of three-move blind signature schemes. In *EUROCRYPT*, pages 197–215, 2010.
27. S. Garg, R. Bhaskar, and S. V. Lokam. Improved bounds on security reductions for discrete log based signatures. In *CRYPTO*, pages 93–107, 2008.
28. C. Gentry and D. Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In *STOC*, pages 99–108, 2011.
29. R. Granger. On the static Diffie-Hellman problem on elliptic curves over extension fields. In *ASIACRYPT*, pages 283–302. 2010.
30. J. Herranz and F. Laguillaumie. Blind ring signatures secure under the chosen-target-CDH assumption. In *International Conference on Information Security – ISC*, pages 117–130, 2006.
31. A. Joux, R. Lercier, D. Naccache, and E. Thom. Oracle-assisted static Diffie-Hellman is easier than discrete logarithms. In *Cryptography and Coding*, pages 351–367. 2009.
32. A. Juels, M. Luby, and R. Ostrovsky. Security of blind digital signatures. In *CRYPTO*, pages 150–164. 1997.
33. J. Katz, D. Schröder, and A. Yerukhimovich. Impossibility of blind signatures from one-way permutations. In *TCC*, pages 615–629, 2011.
34. N. Koblitz and A. Menezes. Another look at non-standard discrete log and Diffie-Hellman problems. Cryptology ePrint Archive, Report 2007/442, 2007.
35. T. Okamoto and D. Pointcheval. The Gap-problems: A new class of problems for the security of cryptographic schemes. In *PKC*, pages 104–118, 2001.
36. P. Paillier and D. Vergnaud. Discrete-log-based signatures may not be equivalent to discrete log. In *ASIACRYPT*, pages 1–20, 2005.
37. R. Pass. Limits of provable security from standard assumptions. In *STOC*, pages 109–118, 2011.
38. R. Pass and M. Venkitasubramaniam. On constant-round concurrent zero-knowledge. In *TCC*, pages 553–570, 2008.
39. D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, 2000.
40. M. Prabhakaran, A. Rosen, and A. Sahai. Concurrent zero knowledge with logarithmic round-complexity. In *FOCS*, pages 366–375, 2002.
41. R. Richardson and J. Kilian. On the concurrent composition of zero-knowledge proofs. In *EUROCRYPT*, pages 415–431, 1999.
42. Y. Seurin. On the exact security of Schnorr-type signatures in the random oracle model. In *EUROCRYPT*, pages 554–571, 2012.