Journal of Cryptology, Volume 16, Number 3.
Pages 185–215, Springer-Verlag, 2003.

The One-More-RSA-Inversion Problems and the Security of Chaum's Blind Signature Scheme *

M. Bellare[1], C. Namprempre[1], D. Pointcheval[2], and M. Semanko[1]

[1] Dept. of Computer Science & Engineering, University of California, San Diego
9500 Gilman Drive, La Jolla, CA 92093, USA
Email: {mihir,cnamprem,msemanko}@cs.ucsd.edu
URL: http://www-cse.ucsd.edu/users/{mihir,cnamprem,msemanko}
Supported in part by a 1996 Packard Foundation Fellowship in Science and Engineering, NSF grant CCR-0098123, NSF grant ANR-0129617 and an IBM Faculty Partnership Development Award.

[2] Dépt. d'Informatique-CNRS, École Normale Supérieure
45 rue d'Ulm, 75230 Paris, Cedex 05, France.
Email: David.Pointcheval@ens.fr
URL: http://www.di.ens.fr/users/pointche

Abstract. We introduce a new class of computational problems which we call the "one-more-RSA-inversion" problems. Our main result is that two problems in this class, which we call the chosen-target and known-target inversion problems respectively, have polynomially-equivalent computational complexity. We show how this leads to a proof of security for Chaum's RSA-based blind signature scheme in the random oracle model based on the assumed hardness of either of these problems. We define and prove analogous results for "one-more-discrete-logarithm" problems. Since the appearence of the preliminary version of this paper, the new problems we have introduced have found other uses as well.

Keywords: Blind digital signatures, Digital cash, RSA

1 Introduction

We introduce a new class of computational problems which we call the "one-more-RSA-inversion" problems. We then use problems from this class as a basis to prove security of Chaum's RSA-based blind signature scheme [12]. We begin with a high-level description of our approach and its motivation.

Example. Blind signature schemes are the central cryptographic component of digital-cash schemes, used to enable a Withdrawer to obtain a Bank's signature on some token without revealing this token to the bank, thereby creating a valid but anonymous e-coin. Chaum's RSA-based blind signature scheme [12] is simple and practical, and (assuming the underlying hash function is properly chosen) has so far resisted attacks. Yet there seems little hope of proving its security based on the "standard" one-wayness assumption about the RSA function: the security of the scheme seems to rely on different, and perhaps stronger, properties of RSA.

* This is the full version of "The Power of RSA Inversion Oracles and the Security of Chaum's RSA-Based Blind Signature Scheme" [4], presented at Financial Cryptography '01.

1.1 *The Gap between Proofs and Practice*

The above is a common situation in cryptography. It exhibits a gap created by the computational assumptions we prefer to make and the schemes we want to use.

The reliance on unproven computational properties of RSA for security naturally inclines us to be conservative and to stick to standard assumptions, of which the favorite is that RSA is one-way. Designers who have worked with RSA know, however, that it seems to have many additional strengths. These are typically exploited, implicitly rather than explicitly, in their designs. The resulting schemes might well resist attack but are dubbed "heuristic" because no proof of security based on the standard assumption seems likely. This leads designers to seek alternative schemes that can be proven secure under the standard assumptions. If the alternatives have cost comparable to that of the original scheme then they are indeed attractive replacements for the latter. But often they are more expensive. Meanwhile, the use of the original practical scheme is being discouraged even though it might very well be secure.

Making New Assumptions. We suggest that practical RSA-based schemes that have resisted attack (for example, Chaum's RSA-based blind signature scheme) might be manifestations of strengths of the RSA function that have not so far been properly abstracted or formalized. We suggest that one should build on the intuition of designers and formulate explicit computational problems that capture the above-mentioned strengths and suffice to prove the security of the scheme. 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. This process will lead to a better understanding of the security of schemes. It will also highlight computational problems that might then be recognized as being at the core of other schemes, and enlarge the set of assumptions we might be willing to make, leading to benefits in the design or analysis of other schemes.

This Paper. In this paper, we formalize a class of computational problems which we call *one-more-RSA-inversion* problems. They are natural extensions of the RSA-inversion problem underlying the notion of one-wayness to a setting where the adversary has access to a decryption oracle. We study several specific problems in this class and the relations between them, and then show how the assumed hardness of some of these problems leads to a proof of security of Chaum's blind signature scheme in the random oracle model [7].

1.2 *The One-More-RSA-Inversion Problems*

Recall that associated to a modulus N and an encryption exponent e are the RSA function and its RSA-inverse defined by

$$\mathsf{RSA}_{N,e}(x) = x^e \bmod N \text{ and } \mathsf{RSA}^{-1}_{N,e}(y) = y^d \bmod N$$

where $x, y \in \mathbb{Z}_N^*$ and d is the decryption exponent. To *invert* RSA at a point $y \in \mathbb{Z}_N^*$ means to compute $x = \mathsf{RSA}^{-1}_{N,e}(y)$. The commonly made and believed assumption that the RSA function is one-way says that the following problem is (computationally) hard: Given N, e and a random target point $y \in \mathbb{Z}_N^*$, compute $y^d \bmod N$. In this paper we are interested in settings where the legitimate user (and hence the adversary) has access to an oracle $\mathsf{RSA}^{-1}_{N,e}(\cdot)$ for the inverse RSA function. (The adversary can provide a value $y \in \mathbb{Z}_N^*$ to its oracle and get back $x = \mathsf{RSA}^{-1}_{N,e}(y) = y^d \bmod N$, but it is not directly given d.) A security property apparently possessed by RSA is that an adversary can

only make "trivial" use of this oracle. We capture this in the following way. The adversary is given some random *target points* $y_1, \ldots, y_n \in \mathbb{Z}_N^*$, and we say it wins if the number of these points whose RSA-inverse it manages to compute exceeds the number of calls it makes to its oracle. That is, it computes "one more RSA-inverse."

Within this framework we consider two specific problems. In the known-target inversion problem, the adversary, to win, must output the inverses of all target points, using a number of decryption oracle queries that is one fewer than the number of target points. In the chosen-target inversion problem, the adversary does not have to compute the RSA-inverses of all target points but instead can choose to invert any number of them that it likes, and wins as long as the number of decryption oracle queries that it makes is strictly fewer than the number of target points it correctly inverts.

Note that the special case of the known-target inversion problem in which there is just one target point is exactly the problem underlying the notion of one-wayness. In this sense, we consider security against known-target inversion to be a natural extension of one-wayness to a setting where the adversary has access to an RSA-inversion oracle.

The formal definitions of the problems provided in Section 2 are parameterized via the number of target points, and, for the chosen-target inversion problem, also the number of these that the adversary inverts. These parameterized formulations are more convenient for the proof of our main result. An alternative, un-parameterized formulation that seems more convenient for applications is provided and proved equivalent to the original formulation in Section 5.

As noted in Remark 5, these problems can be hard only if factoring does not reduce to RSA inversion. Some evidence that the latter is true is provided by Boneh and Venkatesan [11].

1.3 *Relations Among One-More-RSA-Inversion Problems*

It is easy to see that if the chosen-target inversion problem is hard then so is the known-target inversion problem. However, it is conceivable that the ability to choose the target points might help the adversary considerably. Our main result is that this is not so. Corollary 10 says that the chosen-target inversion problem is hard if and only if the known-target inversion problem is hard. We prove this by showing how, given any polynomial-time adversary B that solves the known-target inversion problem, we can design a polynomial-time adversary A that solves the chosen-target inversion problem with about the same probability. The reduction exploits linear algebraic techniques which in this setting are complicated by the fact that the order $\phi(N)$ of the group over which we must work is not known to the adversary.

1.4 *Security of Chaum's RSA-Based Blind Signature Scheme*

In Chaum's RSA-based blind signature scheme, the signer's public key is N, e, and its secret key is N, d where these quantities are as in the RSA system. The signature of a message M is

$$x = \mathsf{RSA}_{N,e}^{-1}(H(M)) = H(M)^d \bmod N \tag{1}$$

where $H\colon\ \{0,1\}^* \to \mathbb{Z}_N^*$ is a public hash function. A message-tag pair (M, x) is said to be *valid* if x is as in Equation (1). The blind signature protocol enables a user to obtain the signature of a message M without revealing M to the signer, as follows.

The user picks r at random in $\mathbb{Z}_N^*$, computes $\overline{M} = r^e \cdot H(M) \bmod N$, and sends $\overline{M}$ to the signer. The latter computes $\overline{x} = \mathsf{RSA}_{N,e}^{-1}(\overline{M}) = \overline{M}^d \bmod N$ and returns $\overline{x}$ to the user, who extracts the signature $x = \overline{x} \cdot r^{-1} \bmod N$ of M from it. Two properties are desired, *blindness* and *unforgeability*. Blindness means the signer does not learn anything about M from the protocol that it did not know before, and it is easy to show that this is unconditionally true [12]. Unforgeability in this context is captured via the notion of one-more-forgery of Pointcheval and Stern [27, 28]. (The standard notion of [18] does not apply to blind signatures.) The forger can engage in interactions with the signer in which it might not follow the prescribed protocol for the user. (As discussed further in Section 6 there are, in general, a variety of attack models for these interactions [27, 28, 20, 25], but in the case of the RSA blind signature protocol, all are equivalent.) Nothing prevents it from coming up with one valid message-tag pair per protocol execution (to do this, it just has to follow the user protocol) but we want it to be hard to come up with more. We ask that the number of valid message-tag pairs that a forger can produce does not exceed the number of executions of the blind-signature protocol in which it engages with the signer.

It is the unforgeability property that has been the open question about the RSA-based blind signature scheme. Michels, Stadler and Sun [24] show that one can successfully obtain one-more forgery if the hash function is poorly implemented. Here, we will assume that the hash function is a random oracle. (The forger and signer both get an oracle for H.) In that case, the signature scheme is the FDH scheme of [8]. This scheme is proven to meet the standard security notion for digital signatures of [18] in the random oracle model assuming that RSA is one-way [8, 13], but this result will not help us here. To date, no attacks against the one-more-forgery goal are known on the blind FDH-RSA signature scheme. We would like to support this evidence of security with proofs.

When the forger interacts with a signer in Chaum's blind signature protocol detailed above, the former effectively has access to an RSA-inversion oracle: it can provide the signer any $\overline{M} \in \mathbb{Z}_N^*$ and get back $\overline{M}^d \bmod N$. It is the presence of this oracle that makes it unlikely that the one-wayness of RSA alone suffices to guarantee unforgeability. However, the one-more-RSA-decryption problems were defined precisely to capture settings where the adversary has an RSA-inversion oracle, and we will be able to base the security of the signature scheme on hardness assumptions about them.

In Lemma 18, we provide a reduction of the security against one-more-forgery of the FDH-RSA blind signature scheme, in the random oracle model, to the security of the RSA chosen-target inversion problem. Appealing to Corollary 10 we then get a proof of unforgeability for the blind FDH-RSA scheme, in the random oracle model, under the assumption that the RSA known-target inversion problem is hard. These results simplify the security considerations of the blind FDH-RSA scheme by eliminating the hash function and signature issues from the picture, leaving us natural problems about RSA to study.

1.5 *Perspective*

An obvious criticism of our results is that the proof of security of the blind FDH-RSA signature scheme is under a novel and extremely strong assumption which is not only hard to validate but crafted to have the properties necessary to prove the security of the signature scheme. This is true, and we warn that the assumptions should be treated with caution. But we suggest that our approach and results have pragmatic

value. Certainly, one could leave the blind RSA signature scheme unanalyzed until someone proves security based on the one-wayness of RSA, but this is likely to be a long wait. Meanwhile, we would like to use the scheme and the practical thing to do is to understand the basis of its security as best we can. Our results isolate clear and simply stated properties of the RSA function that underlie the security of the blind signature scheme and make the task of the security analyst easier by freeing him or her from consideration of properties of signatures and hash functions. It is better to know exactly what we are assuming, even if this is very strong, than to know nothing at all.

1.6 *One-More-Discrete-Logarithm Problems*

The analogues of the one-more-RSA-inversion problems can be formulated for any family of one-way permutations. Section 7 provides formulations of the one-more-discrete-log problems that underly the discrete exponentiation one-way permutation in appropriate groups, and proves that the known-target inversion and chosen-target inversion problems have polynomially-equivalent computational complexity. This proof is actually easier than the one for RSA because in the discrete log case the order of the group is public information.

Other RSA-Related Assumptions. Other non-standard RSA-related computational problems whose study has been fruitful include: strong-RSA, introduced in [16, 3] and used effectively in [17, 14]; dependent-RSA [26]; and an additive RSA related problem from [21]. For more information about RSA properties and attacks see [10].

1.7 *Relation to Preliminary Version of This Paper*

A preliminary version of this paper appeared as [4]. The version you are reading contains the following extensions or additions: Corollary 10 (and hence Corollary 17) in the preliminary version of this paper had made the assumption that the RSA encryption exponent is a prime number, an assumption removed in this paper; the alternative formulations of the one-more-inversion problems in Section 5 are new; this paper provides the definitions and proofs about one-more-discrete-logarithm problems that were claimed in the preliminary version.

Subsequent Work. Since the one-more-inversion problems were introduced in the preliminary version of this paper [4], they have found numerous other uses:

- Bellare and Palacio [6] prove that the GQ identification scheme [19] is secure against impersonation under active (and concurrent) attack under the assumption that the RSA known-target inversion problem is hard, and that Schnorr's identification scheme [30] is secure against impersonation under active (and concurrent) attack assuming that the known-target one-more-discrete-logarithm problem is hard. In both cases, they are answering quite long-standing open questions.
- Assuming hardness of the RSA known-target inversion problem, Bellare and Neven [5] prove the security of an RSA based transitive signature scheme suggested by Micali and Rivest [23].
- These problems have been used to prove security of some practical two-party RSA-based signature protocols [9].

2 The One-More-RSA-Inversion Problems

Throughout this paper, $k \in \mathbb{N}$ denotes the security parameter. An *RSA key generator* is a (randomized) poly(k)-time algorithm KeyGen that on input k returns a triple (N, e, d) where: the *modulus* N is a product of two, distinct odd primes p, q and satisfies $2^{k-1} \leq N < 2^k$; the *encryption exponent* e and *decryption exponent* d are elements of $\mathbb{Z}^*_{\varphi(N)}$ satisfying $ed \equiv 1 \pmod{N}$ where $\varphi(N) = (p-1)(q-1)$. We say that KeyGen is a *prime-exponent* RSA key generator if e is always a prime number.

In this paper, we do not pin down any particular choice of key generator, but rather formulate computational problems in which the key generator is a parameter. For this purpose, we now fix an RSA key generator KeyGen that will be referred to throughout. (There are numerous different possible RSA key generators, which might differ in the distribution or structure of the primes p, q chosen, or the values of the encryption exponent. For example, one might want to use $e = 3$, in which case p, q would be chosen such that neither $p-1$ nor $q-1$ is a multiple of 3. Another common choice is that both p and q are random primes of approximately equal length, and e is chosen at random from $\mathbb{Z}^*_{\varphi(N)}$.)

In this section we define the parameterized versions of the known-target and chosen-target RSA inversion problems. The alternative formulations are presented in Section 5.

Our definitional paradigm is to associate to any given adversary an *advantage* function which on input the security parameter k returns the probability that an associated *experiment* returns 1. The problem is *hard* if the advantage of any adversary of time-complexity poly(k) is negligible, and is *easy* if it is not hard.

2.1 *Definitions of the Problems*

As an introduction to the notions we introduce, it is useful to recall the standard notion, couching it in a way more suitable for comparison with the new notions.

Definition 1 (Single-Target Inversion Problem: RSA-STI). Let $k \in \mathbb{N}$ be the security parameter. Let A be an adversary. Consider the following experiment:

Experiment $\mathbf{Exp}^{\text{rsa-sti}}_A(k)$
$(N, e, d) \xleftarrow{R} \text{KeyGen}(k)$
$y \xleftarrow{R} \mathbb{Z}^*_N$; $x \leftarrow A(N, e, k, y)$
If $x^e \equiv y \pmod{N}$ then return 1 else return 0

We define the advantage of A via

$$\mathbf{Adv}^{\text{rsa-sti}}_A(k) = \Pr[\,\mathbf{Exp}^{\text{rsa-sti}}_A(k) = 1\,] \,.$$

The RSA-STI problem is said to be *hard* —in more standard terminology, RSA is said to be *one-way*— if the function $\mathbf{Adv}^{\text{rsa-kti}}_{A,m}(\cdot)$ is negligible for any adversary A whose time-complexity is polynomial in the security parameter k. ▮

We now proceed to define the known-target inversion problem. We denote by $(\cdot)^d \bmod N$ the oracle that takes input $y \in \mathbb{Z}^*_N$ and returns its RSA-inverse y^d. An adversary solving the known-target inversion problem is given oracle access to $(\cdot)^d \bmod N$ and is given $m(k)+1$ targets where $m : \mathbb{N} \to \mathbb{N}$. Its task is to compute the RSA-inverses of *all* the targets while submitting at most $m(k)$ queries to the oracle.

Definition 2 (Known-Target Inversion Problem: RSA-KTI$[m]$). Let $k \in \mathbb{N}$ be the security parameter, and let $m : \mathbb{N} \to \mathbb{N}$ be a function of k. Let A be an adversary with access to an RSA-inversion oracle $(\cdot)^d \bmod N$. Consider the following experiment:

Experiment $\mathbf{Exp}^{\text{rsa-kti}}_{A,m}(k)$
$(N, e, d) \xleftarrow{R} \text{KeyGen}(k)$
For $i = 1$ to $m(k) + 1$ do $y_i \xleftarrow{R} \mathbb{Z}_N^*$
$(x_1, \ldots, x_{m(k)+1}) \leftarrow A^{(\cdot)^d \bmod N}(N, e, k, y_1, \ldots, y_{m(k)+1})$
If the following are both true then return 1 else return 0
- $\forall i \in \{1, \ldots, m(k) + 1\} : x_i^e \equiv y_i \pmod N$
- A made at most $m(k)$ oracle queries

We define the advantage of A via

$$\mathbf{Adv}^{\text{rsa-kti}}_{A,m}(k) = \Pr[\,\mathbf{Exp}^{\text{rsa-kti}}_{A,m}(k) = 1\,] .$$

The RSA-KTI$[m]$ problem is said to be *hard* if the function $\mathbf{Adv}^{\text{rsa-kti}}_{A,m}(\cdot)$ is negligible for any adversary A whose time-complexity is polynomial in the security parameter k. The RSA-KTI is said to be *hard* if RSA-KTI$[m]$ is hard for all polynomially-bounded $m(\cdot)$. ∎

Notice that RSA-KTI[0] is the same as RSA-STI. That is, the standard assumption that RSA is one-way is exactly the same as the assumption that RSA-KTI[0] is hard.

We proceed to the chosen-target inversion problem. An adversary solving this problem is given access to an RSA-inversion oracle as above, and $n(k)$ target points, where $n : \mathbb{N} \to \mathbb{N}$. Its task is to compute the RSA-inverses of $m(k) + 1$ of the given targets, where $m : \mathbb{N} \to \mathbb{N}$ and $m(k) < n(k)$, while submitting at most $m(k)$ queries to the oracle. The adversary indicates which $m(k) + 1$ points, out of the $n(k)$ target points, it has chosen to invert, by outputting an injective map π as specified below. We assume that the adversary outputs this map in a canonical format such as a list of values $(\pi(1), \pi(2), \ldots)$.

Definition 3 (Chosen-Target Inversion Problem: RSA-CTI$[n, m]$). Let $k \in \mathbb{N}$ be the security parameter, and let $m, n : \mathbb{N} \to \mathbb{N}$ be functions of k such that $m(\cdot) < n(\cdot)$. Let B be an adversary with access to an RSA-inversion oracle $(\cdot)^d \bmod N$. Consider the following experiment:

Experiment $\mathbf{Exp}^{\text{rsa-cti}}_{B,n,m}(k)$
$(N, e, d) \xleftarrow{R} \text{KeyGen}(k)$
For $i = 1$ to $n(k)$ do $\overline{y}_i \xleftarrow{R} \mathbb{Z}_N^*$
$(\pi, \overline{x}_1, \ldots, \overline{x}_{m(k)+1}) \leftarrow B^{(\cdot)^d \bmod N}(N, e, k, \overline{y}_1, \ldots, \overline{y}_{n(k)})$
If the following are all true then return 1 else return 0
- $\pi\colon \{1, \ldots, m(k) + 1\} \to \{1, \ldots, n(k)\}$ is injective
- $\forall i \in \{1, \ldots, m(k) + 1\} : \overline{x}_i^e \equiv \overline{y}_{\pi(i)} \pmod N$
- A made at most $m(k)$ oracle queries

We define the advantage of B via

$$\mathbf{Adv}^{\text{rsa-cti}}_{B,n,m}(k) = \Pr[\,\mathbf{Exp}^{\text{rsa-cti}}_{B,n,m}(k) = 1\,] .$$

The RSA-CTI$[n, m]$ problem is said to be *hard* if the function $\mathbf{Adv}^{\text{rsa-cti}}_{B,n,m}(\cdot)$ is negligible for any adversary B whose time complexity is polynomial in the security parameter k. The RSA-CTI problem is said to be *hard* if RSA-CTI$[n, m]$ is hard for all polynomially-bounded $n(\cdot)$ and $m(\cdot)$ where $m(\cdot) < n(\cdot)$. ∎

2.2 *Simple Relations among the Problems*

We note a few simple relations before going to the main result.

Remark 4. Let n, m: $\mathbb{N} \to \mathbb{N}$ be polynomially-bounded functions of the security parameter k such that $m(\cdot) < n(\cdot)$. If the RSA-CTI$[n, m]$ problem is hard then so is the RSA-KTI$[m]$ problem. This is justified as follows: given an adversary A for RSA-KTI$[m]$, we let B be the adversary for RSA-CTI$[n, m]$ that runs A on input the first $m(k)+1$ of B's target points and returns the values returned by A, together with the map π defined by $\pi(i) = i$ for $i = 1, \ldots, m(k)+1$. Then B's advantage is the same as A's. ▮

Remark 5. If factoring reduces to RSA inversion then there exists a polynomially-bounded function m: $\mathbb{N} \to \mathbb{N}$ such that RSA-KTI$[m]$ is easy. (So the assumption that either the known-target or chosen-target inversion problems is hard is at least as strong as the assumption that factoring does not reduce to RSA inversion.) Let us briefly justify this. Assume that factoring reduces to RSA inversion. This means there is a polynomial-time algorithm R such that the probability that the following experiment returns 1 is non-negligible:

Experiment
  $(N, e, d) \xleftarrow{R} \mathrm{KeyGen}(k)$
  $p \leftarrow R^{(\cdot)^d \bmod N}(N, e, k)$
  If $p \neq 1$ and $p \neq N$ and p divides N then return 1 else return 0.

Let m: $\mathbb{N} \to \mathbb{N}$ be a polynomially-bounded, polynomial-time computable function such that the number of oracle queries made by R is strictly less than $m(k)$ whenever the security parameter is k in the experiment above. We define adversary A as follows:

Experiment
Adversary $A^{(\cdot)^d \bmod N}(N, e, k, y_1, \ldots, y_{m(k)+1})$
  $p \leftarrow R^{(\cdot)^d \bmod N}(N, e, k)$
  Compute d from p and N/p
  Compute and return $y_1^d, \ldots, y_{m(k)+1}^d \bmod N$

The adversary A runs the algorithm R, answering to its inversion queries with the answers from its own oracle. It uses the fact that possession of the prime factors of N enables computation of the decryption exponent d, and having computed d, it can of course compute the RSA-inversions of as many points as it pleases. ▮

We will now present some technical lemmas, and then proceed to the proof of Theorem 9. The reader might prefer to begin with Section 4 and refer to Section 3 as needed.

3 Technical Lemmas

Below, if $s \geq 1$ is an integer then I_s denotes the s by s identity matrix.

Lemma 6. *Let $s \geq 1$ be an integer, and let*

$$C = \begin{bmatrix} c_{1,1} & \cdots & c_{1,s} \\ \vdots & & \vdots \\ c_{s,1} & \cdots & c_{s,s} \end{bmatrix} \quad \textit{and} \quad D = \begin{bmatrix} d_{1,1} & \cdots & d_{1,s} \\ \vdots & & \vdots \\ d_{s,1} & \cdots & d_{s,s} \end{bmatrix}$$

be integer matrices such that $C \cdot D = \det(C) \cdot I_s$. *Suppose* N, e *is an RSA public key and* N, d *is the corresponding secret key. Suppose* $y_i, \overline{y}_i, v_i \in \mathbb{Z}_N^*$ *for* $i = 1, \ldots, s$ *are related via*

$$\overline{y}_i \;\equiv\; v_i^{-e} \cdot \prod_{j=1}^{s} y_j^{c_{j,i}} \pmod{N} . \tag{2}$$

Let $\overline{x}_i = \overline{y}_i^d \bmod N$ *for* $i = 1, \ldots, s$. *Then, for* $j = 1, \ldots, s$, *we have*

$$(y_j^d)^{\det(C)} \;\equiv\; \prod_{i=1}^{s} (v_i \cdot \overline{x}_i)^{d_{i,j}} \pmod{N} . \;\blacksquare \tag{3}$$

Proof (Lemma 6). Let $\delta_{l,j} = 1$ if $l = j$ and 0 otherwise. Since $C \cdot D = \det(C) \cdot I_s$ we know that

$$\sum_{i=1}^{s} c_{l,i} d_{i,j} \;=\; \det(C) \cdot \delta_{l,j} \tag{4}$$

for all $l, j = 1, \ldots, s$. We now verify Equation (3). Suppose $1 \le j \le s$. In the following, computations are all mod N. From Equation (2), we have

$$\prod_{i=1}^{s} (v_i \cdot \overline{x}_i)^{d_{i,j}} \;=\; \prod_{i=1}^{s} \left[v_i \cdot \left(v_i^{-e} \cdot \prod_{l=1}^{s} y_l^{c_{l,i}} \right)^d \right]^{d_{i,j}} \;=\; \prod_{i=1}^{s} \left[v_i \cdot v_i^{-1} \cdot \prod_{l=1}^{s} (y_l^d)^{c_{l,i}} \right]^{d_{i,j}} .$$

Simplifying the last expression, we obtain

$$\prod_{i=1}^{s} \prod_{l=1}^{s} (y_l^d)^{c_{l,i} d_{i,j}} \;=\; \prod_{l=1}^{s} \prod_{i=1}^{s} (y_l^d)^{c_{l,i} d_{i,j}} \;=\; \prod_{l=1}^{s} (y_l^d)^{\sum_{i=1}^{s} c_{l,i} d_{i,j}} \;=\; \prod_{l=1}^{s} (y_l^d)^{\det(C) \cdot \delta_{l,j}}$$

where the last equality is by Equation (4). Finally, we use the fact that $\delta_{l,j} = 1$ if $l = j$ and 0 otherwise. This tells us that the above is $(y_j^d)^{\det(C)}$ as desired.

Lemma 7. *Let* N, e *be an RSA public key and* N, d *the corresponding secret key. Let* $\alpha \in \mathbb{N}$ *and* $y, z \in \mathbb{Z}_N^*$. *If* $\gcd(\alpha, e) = 1$ *and* $(y^d)^\alpha \equiv z \pmod{N}$ *then* $(z^a y^b)^e \equiv y \pmod{N}$ *where* a, b *are the unique integers such that* $a\alpha + be = 1$. $\blacksquare$

Proof (Lemma 7). This is a standard calculation:

$$(z^a y^b)^e = (y^{d\alpha})^{ae} y^{be} = y^{\alpha a + be} = y^1 = y$$

where the computations are all mod N.

Next, we consider a question in probabilistic linear algebra. Let $q \ge 2$ and $s \ge 1$ be integers. Let $\mathsf{GLP}(s, q)$ denote the probability that $\gcd(\det(M), q) = 1$ when M is an s by s matrix formed by choosing all entries uniformly and independently from $\mathbb{Z}_q$. Then we have the following lower bounds:

Lemma 8. *Let* $q \ge 2$ *and* $s \ge 1$ *be integers, and assume* $q \le 2^k$. *Then,*

$$\mathsf{GLP}(s, q) \;\ge\; \frac{1}{286^2} \cdot \frac{1}{\ln(k)^2} . \tag{5}$$

Furthermore, if q *is prime, the following better lower bound holds:*

$$\mathsf{GLP}(s, q) \;\ge\; 1 - \frac{1}{q} - \frac{1}{q^2} . \;\blacksquare \tag{6}$$

The proof of the above lemma is in Appendix A.

4 The Equivalence Result

We provide a converse to the claim of Remark 4.

Theorem 9. *Let $n, m\colon \mathbb{N} \to \mathbb{N}$ be polynomially-bounded functions satisfying $m(\cdot) < n(\cdot)$. Then for any adversary B, there exists an adversary A so that*

$$\mathbf{Adv}^{\text{rsa-cti}}_{B,n,m}(k) \;\le\; O(\ln(k)^2) \cdot \mathbf{Adv}^{\text{rsa-kti}}_{A,m}(k)\;, \tag{7}$$

and A has time-complexity

$$T_A(k) \;=\; T_B(k) + O\left(k^3 n(k) m(k) + k^4 m(k) + k^2 m(k)^5 + k m(k)^6\right) \tag{8}$$

where $T_B(\cdot)$ is the time-complexity of B. Furthermore, if KeyGen *is a prime-exponent key generator then Equation (7) can be improved to*

$$\mathbf{Adv}^{\text{rsa-cti}}_{B,n,m}(k) \;\le\; \frac{9}{5} \cdot \mathbf{Adv}^{\text{rsa-kti}}_{A,m}(k)\;. \quad \blacksquare \tag{9}$$

In this theorem, the time-complexity of the adversary refers to the function which on input k returns the execution time of the full associated experiment including the time taken to compute answers to oracle calls, plus the size of the code of the adversary, in some fixed model of computation. This convention simplifies the concrete security considerations in the theorem.

A simple consequence of Theorem 9 and Remark 4 is the following equivalence, which is our main result:

Corollary 10. *Let $n, m\colon \mathbb{N} \to \mathbb{N}$ be polynomially-bounded functions satisfying $m(\cdot) < n(\cdot)$. Then the RSA-KTI[m] problem is hard if and only if the RSA-CTI[n, m] problem is hard.* ∎

Now we present the proof of Theorem 9.

Proof (Theorem 9). The adversary A is depicted in Figure 1. Its input is (N, e, k and) $s = m(k) + 1$ target points $y_1, \ldots, y_s$. Its goal is to compute $y_1^d, \ldots, y_s^d \bmod N$.

Adversary A will begin by computing $n(k)$ points $\overline{y}_1, \ldots, \overline{y}_{n(k)}$ as a (randomized) function of the given points $y_1, \ldots, y_s$. The property we want these to have is that, given the RSA-inverses of *any* s of the points $\overline{y}_1, \ldots, \overline{y}_{n(k)}$, it is possible to extract in polynomial time the RSA-inverses of the original target points, at least with high probability. If such a "reversible embedding" can be implemented then A's work is complete since invoking B on the points $\overline{y}_1, \ldots, \overline{y}_{n(k)}$ will cause the RSA-inverses of some s of these points to be returned. The question is, thus, how to compute and later reverse this "reversible embedding."

Lines 2–5 of Figure 1 show how to compute it. For each j, the point $\overline{y}_j$ is created by first raising each of $y_1, \ldots, y_s$ to a random power and then multiplying the obtained quantities. (This product is then multiplied by a random group element of which A knows the RSA-inverse in order to make sure that $\overline{y}_1, \ldots, \overline{y}_{n(k)}$ are uniformly and independently distributed and thus are appropriate to feed to B.) A detail worth remarking here is the choice of the range from which the exponents $c[j, i]$ are chosen. This is $\mathbb{Z}_q$ where we have set q equal to the encryption exponent e. We will see the reasons for this choice later.

Once the points $\overline{y}_1, \ldots, \overline{y}_{n(k)}$ have been defined, B is invoked. In executing B, adversary A will invoke its own oracle to answer RSA-inversion oracle queries of B. Notice that this means that the number of oracle queries made by A is exactly equal to the number made by B which is $s - 1 = m(k)$. Assuming that B succeeds, A is

Algorithm $A^{(\cdot)^d \bmod N}(N, e, k, y_1, \ldots, y_{m(k)+1})$

1 $q \leftarrow e$; $s \leftarrow m(k) + 1$
2 For $i = 1, \ldots, n(k)$ do
3 $v[i] \xleftarrow{R} \mathbb{Z}_N^*$
4 For $j = 1, \ldots, s$ do $c[j, i] \xleftarrow{R} \mathbb{Z}_q$
5 $\overline{y}_i \leftarrow v[i]^{-e} \prod_{j=1}^{s} y_j^{c[j,i]} \bmod N$
6 $(\pi, \overline{x}_1, \ldots, \overline{x}_s) \leftarrow B^{(\cdot)^d \bmod N}(N, e, k, \overline{y}_1, \ldots, \overline{y}_{n(k)})$
7 For $j = 1, \ldots, s$ do
$v_j \leftarrow v[\pi(j)]$
For $l = 1, \ldots, s$ do $c_{j,l} \leftarrow c[j, \pi(l)]$
8 $C \leftarrow \begin{bmatrix} c_{1,1} & \ldots & c_{1,s} \\ \vdots & & \vdots \\ c_{s,1} & \ldots & c_{s,s} \end{bmatrix}$
9 $\alpha \leftarrow \det(C)$
10 If $\alpha = 0$ then abort
11 Compute a matrix
$D = \begin{bmatrix} d_{1,1} & \ldots & d_{1,s} \\ \vdots & & \vdots \\ d_{s,1} & \ldots & d_{s,s} \end{bmatrix}$
with integer entries such that $C \cdot D = \det(C) \cdot I_s$
12 For $j = 1, \ldots, s$ do
13 $z_j \leftarrow \prod_{i=1}^{s} (v_i \cdot \overline{x}_i)^{d_{i,j}} \bmod N$
14 If $\gcd(\alpha, e) \neq 1$ then abort
15 Compute $a, b \in \mathbb{Z}$ such that $a\alpha + be = 1$ via extended Euclid algorithm
16 For $j = 1, \ldots, s$ do
17 $x_j \leftarrow z_j^a \cdot y_j^b \bmod N$
18 Return $x_1, \ldots, x_s$

Fig. 1. Adversary A of the proof of Theorem 9.

in possession of $\overline{x}_j \equiv \overline{y}_{\pi(j)}^d \pmod N$ for $j = 1, \ldots, s$ where $\pi(j)$ are indices of B's choice that A could not have predicted beforehand. The final step is to recover the RSA-inverses of the original target points.

To this end, A creates the matrix C shown in line 8 of the code. If this matrix has zero determinant then A will not be able to reverse its embedding and aborts. Assuming a non-zero determinant, A would like to invert matrix C. Since the entries are exponents, A would like to work modulo $\phi(N)$ but A does not know this value. Instead, it works over the integers. A can compute a "partial" RSA-inverse, namely an integer matrix D such that $C \cdot D$ is a known integer multiple of the s by s identity matrix I_s. The integer multiple in question is the determinant of C, and thus the matrix D is the adjoint of C. (We will discuss the computation of D more later.) Lines 12–18 show how A then computes $x_1, \ldots, x_s$ which we claim equal $y_1^d, \ldots, y_s^d$.

We now proceed to the detailed analysis. Let NS be the event that $\gcd(\det(C), e) = 1$. Let "A succeeds" denote the event that $x_i = y_i^d$ for all $i = 1, \ldots, s$. Let "B succeeds" denote the event that $\overline{x}_j = \overline{y}_{\pi(j)}^d$ for all $j = 1, \ldots, s$. Then,

$$\begin{aligned} &\Pr[\, A \text{ succeeds} \,] \\ &\quad \geq \Pr[\, A \text{ succeeds} \wedge B \text{ succeeds} \wedge \text{NS} \,] \\ &\quad = \Pr[\, A \text{ succeeds} \mid B \text{ succeeds} \wedge \text{NS} \,] \cdot \Pr[\, B \text{ succeeds} \wedge \text{NS} \,] \,. \end{aligned} \tag{10}$$

Code	**Cost**
"For" loop at line 2	$O(k^3) \cdot n(k) \cdot s$
$\det(C)$	$O(s^4k + s^3k^2)$
Matrix D	$s^2 \cdot O(s^4k + s^3k^2)$
"For" loop at line 12	$O(k^2s) \cdot O(sk)$
Lines 14, 15	$O(sk) \cdot O(k)$
"For" loop at line 16	$O(k^2) \cdot O(k^2s)$
Total	$O(k^3n(k)s + k^4s + k^2s^5 + ks^6)$

Fig. 2. Costs of computations of the algorithm of Figure 1. Recall that $s = m(k) + 1$

We claim that

$$\Pr[\, A \text{ succeeds} \mid B \text{ succeeds} \wedge \text{NS}\,] = 1 \tag{11}$$

$$\Pr[\, B \text{ succeeds} \wedge \text{NS}\,] \geq \mathsf{GLP}(s,q) \cdot \mathbf{Adv}^{\text{rsa-cti}}_{B,n,m}(k)\ . \tag{12}$$

Equations (10), (11), and (12), together with Lemma 8, imply Equation (7). So it remains to verify Equations (11), (12) and the time-complexity claimed in Equation (8). We begin with Equation (11). Lemma 6 tells us that, assuming that B succeeds and that $\det(C) \neq 0$, after line 13 of Figure 1, we have

$$(y_j^d)^{\det(C)} \ \equiv\ z_j \pmod N \tag{13}$$

for $j = 1, \ldots, s$. Assume $\gcd(\alpha, e) = 1$. Then Equation (13) and Lemma 7 imply that at line 17 we have $x_j^e = y_j$ for all $j = 1, \ldots, s$, in other words, A succeeds. Now, note that event NS implies that $\det(C) \neq 0$. This completes the proof of Equation (11).

We now move on to the proof of Equation (12). Due to the random choice of $v[1], \ldots, v[n(k)]$, the points $\overline{y}_1, \ldots, \overline{y}_{n(k)}$ computed at line 5 and then fed to B are uniformly and independently distributed over $\mathbb{Z}_N^*$ regardless of the choices of $c[j,i]$. This means that the events "B succeeds" and NS are independent and also that the probability of the former is the advantage of B. Thus, we have

$$\Pr[\, B \text{ succeeds} \wedge \text{NS}\,] \ =\ \Pr[\,\text{NS}\,] \cdot \Pr[\, B \text{ succeeds}\,] \ =\ \Pr[\,\text{NS}\,] \cdot \mathbf{Adv}^{\text{rsa-cti}}_{B,n,m}(k)\ .$$

By definition of NS we have

$$\Pr[\,\text{NS}\,] \ =\ \mathsf{GLP}(s,q)\ .$$

So Equation (12) follows. Now, if KeyGen is a prime-exponent key generator, we note that $q = e$ (line 1 in Figure 1), so q is a prime, and $q \geq 3$ for RSA. Then, we can apply Equation (6) from Lemma 8 to obtain Equation (9) in Theorem 9 as follows.

$$\Pr[\,\text{NS}\,] \ =\ \mathsf{GLP}(s,q) \ \geq\ 1 - \frac{1}{q} - \frac{1}{q^2} \ =\ 1 - \frac{1}{e} - \frac{1}{e^2} \ \geq\ 1 - \frac{1}{3} - \frac{1}{3^2} \ =\ \frac{5}{9}\ .$$

If KeyGen is not a prime-exponent key generator, we note that $q = e$ and apply Equation (5) from Lemma 8 to obtain Equation (7) in Theorem 9.

We now justify the claim of Equation (8) about the time complexity. The costs of various steps of the algorithm of the adversary A are summarized in Figure 2. We now briefly explain them. As in the code, we let $s = m(k) + 1$. The "For" loop beginning at line 2 involves $n(k) \cdot s$ exponentiations of k-bit exponents which has the cost shown. Computation of determinants is done using the algorithm of [1]. This takes $O(r^4(\log(r) + k) + r^3k^2)$ time to compute the determinant of an r by r integer matrix each of whose entries is at most k-bits long. (Although somewhat faster algorithms

are known [15], they are randomized, and for simplicity, we use a deterministic algorithm.) We use this algorithm in Step 9. In the worst case, e (and hence q) is k-bits long. So the entries of C are at most k-bits long, and the cost of computing $\det(C)$ is $O(s^4(\log(s)+k)+s^3k^2)$, which is $O(s^4k+s^3k^2)$ since $\log(s)=O(k)$. The matrix D is the adjoint matrix of C, namely the transpose of the co-factor matrix of C. We compute it by computing the co-factors using determinants. This involves computing s^2 determinants of submatrices of C so the cost is at most s^2 times the cost of computing the determinant of C. Line 13 involves computing exponentiations modulo N with exponents of the size of entries in D. The Hadamard bound tells us that the entries of D are bounded in size by $O(s(\log(s)+k)$, which simplifies to $O(sk)$, so the cost is this many k-bit multiplications. Euclid's algorithm used for lines 14 and 15 runs in time the product of the lengths of α and e. Finally, the lengths of a, b cannot exceed this time, and they are the exponents in line 17.

5 Alternative Formulations of the Problems

We consider formulations of the one-more-RSA-inversion problems that are an alternative to the parameterized formulations of Section 2, and prove them equivalent to the original ones. This is useful because the alternative formulations are often more convenient in using the one-more-RSA-inversion problems as assumptions in proving the security of other schemes, and, in particular, these formulations are used in Section 6 and in [6, 5].

In these analogues, rather than directly giving an adversary the points to invert as inputs, we allow the adversary to query a *challenge oracle* to obtain the points. A challenge oracle is an oracle that takes no input and simply returns a random challenge point in $\mathbb{Z}_N^*$, where N is the RSA modulus, each time it is queried. (The point is chosen independently and uniformly from $\mathbb{Z}_N^*$ each time the oracle is invoked.) The adversary is given access to this oracle and the RSA-inversion oracle. With this setup we consider the known-target and chosen-target versions of the problem in turn.

In the alternative formulation of the known-target inversion problem, the adversary is considered successful if it outputs correct inverses of all the points returned in response to queries it makes to its challenge oracle, while querying its decryption oracle a number of times that is strictly fewer than the number of times it queries its challenge oracle. This differs from the previous, parameterized formulation of the problem in that the number of target points m is a random variable that depends on the adversary's inputs and coin tosses. This might lead one to think that the adversary's power is increased, but, as we will see, the two formulations end up being equivalent.

Definition 11 (Known-Target Inversion Problem, alternative formulation: RSA-AKTI). Let $k \in \mathbb{N}$ be the security parameter. Let A be an adversary with access to the RSA-inversion oracle $(\cdot)^d \bmod N$ and the challenge oracle O_N. Consider the following experiment:

Experiment $\mathbf{Exp}_A^{\text{rsa-akti}}(k)$

$(N, e, d) \xleftarrow{R} \text{KeyGen}(k)$

$(x_1, \ldots, x_m) \leftarrow A^{(\cdot)^d \bmod N, O_N}(N, e, k)$ where m is the number of queries to O_N

Let $y_1, \ldots, y_m$ be the challenges returned by O_N

If the following are both true then return 1 else return 0

- $\forall i \in \{1, \ldots, m\} : x_i^e \equiv y_i \pmod N$
- A made strictly fewer than m queries to $(\cdot)^d \bmod N$

We define the advantage of A via

$$\mathbf{Adv}^{\text{rsa-akti}}_{A}(k) = \Pr[\,\mathbf{Exp}^{\text{rsa-akti}}_{A}(k) = 1\,]\ .$$

The RSA-AKTI problem is said to be *hard* if the function $\mathbf{Adv}^{\text{rsa-akti}}_{A}(\cdot)$ is negligible for any adversary A whose time-complexity is polynomial in the security parameter k. ▮

In the alternative formulation of the chosen-target inversion problem, the adversary must provide correct inverses of some of the points returned by the challenge oracle while making a number of decryption oracle queries strictly fewer than the number of points it outputs. But the number of points the adversary chooses to invert is up to the adversary, i.e., it is a random variable in the adversary's inputs and coin tosses.

Definition 12 (Chosen-Target Inversion Problem, alternative formulation: RSA-ACTI). Let $k \in \mathbb{N}$ be the security parameter. Let B be an adversary with access to the RSA-inversion oracle $(\cdot)^d \bmod N$ and the challenge oracle O_N. Consider the following experiment:

Experiment $\mathbf{Exp}^{\text{rsa-acti}}_{B}(k)$
$(N, e, d) \xleftarrow{R} \text{KeyGen}(k)$
$(\pi, \overline{x}_1, \ldots, \overline{x}_m) \leftarrow B^{(\cdot)^d \bmod N, O_N}(N, e, k)$ where m is an integer
Let n be the number of queries to O_N
Let $\overline{y}_1, \ldots, \overline{y}_n$ be the challenges returned by O_N
If the following are all true then return 1 else return 0
- $\pi\colon \{1, \ldots, m\} \to \{1, \ldots, n\}$ is injective
- $\forall i \in \{1, \ldots, m\} : \overline{x}_i^e \equiv \overline{y}_{\pi(i)} \pmod N$
- B made strictly fewer than m queries to $(\cdot)^d \bmod N$

We define the advantage of B via

$$\mathbf{Adv}^{\text{rsa-acti}}_{B}(k) = \Pr[\,\mathbf{Exp}^{\text{rsa-acti}}_{B}(k) = 1\,]\ .$$

The RSA-ACTI problem is said to be *hard* if the function $\mathbf{Adv}^{\text{rsa-acti}}_{B}(\cdot)$ is negligible for any adversary B whose time-complexity is polynomial in the security parameter k. ▮

The following theorem states that the original and alternative known-target inversion problems are computationally equivalent.

Theorem 13. *The RSA-KTI problem is hard if and only if the RSA-AKTI problem is hard.* ▮

Proof (Theorem 13). First, for any polynomial-time adversary A, we show that there exists a polynomial-time adversary B and a polynomially-bounded function m such that for all k

$$\mathbf{Adv}^{\text{rsa-akti}}_{A}(k) \leq \mathbf{Adv}^{\text{rsa-kti}}_{B,m}(k)\ .$$

It follows that if the RSA-KTI problem is hard then the RSA-AKTI problem is also hard. The construction is as follows. By assumption, A has running time polynomial in k. So we can choose a polynomial-time computable, polynomially bounded function $m\colon\ \mathbb{N} \to \mathbb{N}$ such that $m(k)$ is strictly more than the running time of A in $\mathbf{Exp}^{\text{rsa-akti}}_{A}(k)$ for all $k \in \mathbb{N}$. We now construct adversary B as shown below.

Algorithm $B^{(\cdot)^d mod N}(N, e, k, y_1, ..., y_{m(k)+1})$
$count \leftarrow 0$
Run A on input (N, e, k), replying to its oracle queries as follows:
If A makes a query to its challenge oracle then
$count \leftarrow count + 1$; Return y_{count} to A
If A makes a query y to its decryption oracle then
Query y to $(\cdot)^d \bmod N$; Return the response to A
Until A halts with some output $(x_1, ..., x_{count})$
For $i = count + 1, ..., m(k) + 1$ do
Query y_i to $(\cdot)^d \bmod N$; Let x_i be the response
Return $(x_1, ..., x_{m(k)+1})$

Our assumption that $m(k)$ exceeds the running time of A means that B will not run out of target points to provide A. If A is successful, it must invert $m' + 1$ points while making at most m' queries to its inversion oracle, for some m'. Regardless of the value of m', adversary B will invert $m(k) + 1$ points while making $m(k)$ queries to its decryption oracle.

Second, let B be any polynomial-time adversary, and $m\colon \mathbb{N} \to \mathbb{N}$ any polynomial-time computable, polynomially-bounded function. We show that there is a polynomial-time adversary A such that for all k

$$\mathbf{Adv}^{\text{rsa-kti}}_{B,m}(k) \leq \mathbf{Adv}^{\text{rsa-akti}}_{A}(k) \ .$$

It follows that if the RSA-AKTI problem is hard then the RSA-KTI[m] problem is hard for all polynomially-bounded functions m and hence the RSA-KTI problem is hard. (If m is not polynomial-time computable, it can be bounded above by a polynomial-time computable function, so restricting our construction to polynomial-time computable functions is enough.) The construction is as follows. The adversary A first computes $m(k)$, using the assumption that m is polynomial-time computable. It then submits $m(k) + 1$ queries to its challenge oracle O_N to obtain $m(k) + 1$ challenges. Then, it runs B with the challenges as inputs, replying to B's queries via its own RSA-inversion oracle. It terminates when B does and outputs what B outputs.

An analogous result is true for the chosen-target versions of the problems.

Theorem 14. *The RSA-CTI problem is hard if and only if the RSA-ACTI problem is hard.* ∎

Proof (Theorem 14). First, for any polynomial-time adversary A, we show that there exists a polynomial-time adversary B and polynomially-bounded functions n, m such that $m(\cdot) < n(\cdot)$ and for all k

$$\mathbf{Adv}^{\text{rsa-acti}}_{A}(k) \leq \mathbf{Adv}^{\text{rsa-cti}}_{B,n,m}(k) \ .$$

It follows that if the RSA-CTI problem is hard then the RSA-ACTI problem is also hard. The construction is as follows. By assumption, A has running time polynomial in k. So we can choose a polynomial-time computable, polynomially bounded function $m\colon \mathbb{N} \to \mathbb{N}$ such that $m(k)$ is strictly more than the running time of A in $\mathbf{Exp}^{\text{rsa-akti}}_{A}(k)$ for all $k \in \mathbb{N}$. We let $n(\cdot) = m(\cdot) + 1$. We now construct adversary B as shown below.

Algorithm $B^{(\cdot)^d mod N}(N, e, k, y_1, ..., y_{n(k)})$
$count \leftarrow 0$
Run A on input (N, e, k), replying to its oracle queries as follows:
If A makes a query to its challenge oracle then

$count \leftarrow count + 1$; Return y_{count} to A
If A makes a query y to its decryption oracle then
Query y to $(\cdot)^d \bmod N$; Return the response to A
Until A halts with some output $(\pi, \overline{x}_1, ..., \overline{x}_{count})$
$i \leftarrow 0$
For $j = count + 1, ..., n(k)$ do
$i \leftarrow i + 1$
If i is not in the range of π then
Query y_i to $(\cdot)^d \bmod N$; Let $\overline{x}_j$ be the response
Set $\pi(j)$ to i
Return $(\pi, \overline{x}_1, ..., \overline{x}_{n(k)})$

Second, let B be any polynomial-time adversary, and n, m: $\mathbb{N} \to \mathbb{N}$ any polynomial-time computable, polynomially-bounded functions satisfying $m(\cdot) < n(\cdot)$. We show that there is a polynomial-time adversary A such that for all k

$$\mathbf{Adv}^{\text{rsa-cti}}_{B,n,m}(k) \;\le\; \mathbf{Adv}^{\text{rsa-acti}}_{A}(k) \; .$$

As in the proof of Theorem 13, it follows that if the RSA-ACTI problem is hard then the RSA-CTI problem is hard. The construction is as follows. The adversary A first computes $n(k), m(k)$, using the assumption that the functions in question are polynomial-time computable. It then submits $n(k)$ queries to its challenge oracle O_N to obtain $n(k)$ challenges. Then, it runs B with the challenges as inputs, replying to B's queries via its own RSA-inversion oracle. It halts when B does and outputs what B outputs.

6 The RSA Blind Signature Scheme

The RSA blind signature scheme [12] consists of three components: the key generation algorithm KeyGen described in Section 2; the *signing protocol* depicted in Figure 3; and the *verification algorithm*. The signer has public key N, e and secret key N, d. Here H: $\{0,1\}^* \to \mathbb{Z}_N^*$ is a public hash function which in our security analysis will be modeled as a random oracle [7]. In that case, the signature schemes is the FDH-RSA scheme of [8]. A message-tag pair (M, x) is said to be valid if $x^e \bmod N$ is equal to $H(M)$. The verification algorithm is the same as that of FDH-RSA: to verify the message-tag pair (M, x) using a public key (N, e), one simply checks if the message-tag pair is valid.

Unforgeability. In the standard formalization of security of a digital signature scheme —-namely unforgeability under adaptive chosen-message attack [18]— the adversary gets to submit messages of its choice to the signer and obtain their signature, and is then considered successful if it can forge the signature of a new message. This formalization does not apply for blind signatures because here nobody submits any messages to the signer to sign, and in fact the user is supposed to use the signer to compute a signature on a message which the signer does not know. Instead, we use the notion of security against one-more-forgery introduced in [27, 28]. The adversary (referred to as a *forger* in this context) is allowed to play the role of the user in the blind signature protocol. After some number of such interactions, it outputs a sequence of message-tag pairs. It wins if the number of these that are valid exceeds the number of protocol instances in which it engaged.

There are numerous possibilities with regard to the manner in which the adversary is allowed to interact with the signer, giving rise to different attack models. Some that

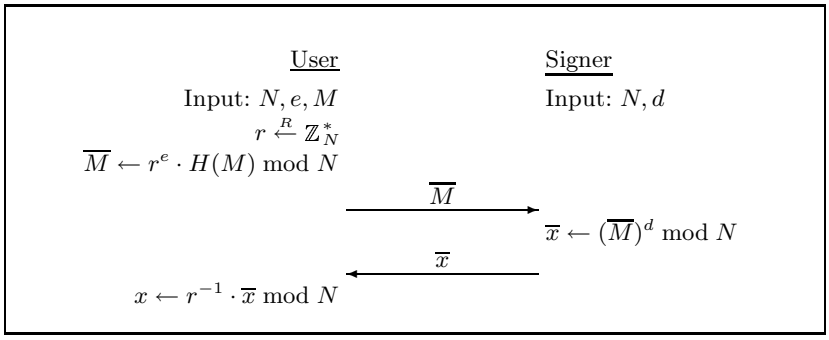

Fig. 3. Blind signing protocol for FDH-RSA

have been considered are the sequential [27, 28] (where the adversary must complete one interaction before beginning another), the parallel [27, 28] or adaptive-interleaved [20] (where the adversary can engage the signer in several concurrent interactions), and a restricted version of the latter called synchronized-parallel [25]. However, in the blind signature protocol for FDH-RSA, the signer has only one move, and in this case the power of all these different types of attacks is the same.

Notice that in its single move the signer simply inverts the RSA function on the value supplied to it by the user in the previous move. Thus, the signer is simply an RSA inversion oracle. With this simplification we can make the following definition for security against one-more forgery which will cover all types of attacks.

Below, we let $[\{0,1\}^* \to \mathbb{Z}_N^*]$ denote the set of all maps from $\{0,1\}^*$ to $\mathbb{Z}_N^*$. It is convenient to let the notation $H \xleftarrow{R} [\{0,1\}^* \to \mathbb{Z}_N^*]$ mean that we select a hash function H at random from this set. The discussion following the definition clarifies how we implement this selection of an object at random from an infinite space.

Definition 15 (Unforgeability of the blind FDH-RSA signature scheme). Let $k \in \mathbb{N}$ be the security parameter. Let F be a forger with access to an RSA-inversion oracle and a hash oracle, denoted $(\cdot)^d \bmod N$ and $H(\cdot)$, respectively. Consider the following experiment:

Experiment $\mathbf{Exp}_F^{\text{rsa-omf}}(k)$
$H \xleftarrow{R} [\{0,1\}^* \to \mathbb{Z}_N^*]$
$(N, e, d) \xleftarrow{R} \text{KeyGen}(k)$
$((M_1, x_1), \ldots, (M_m, x_m)) \leftarrow F^{(\cdot)^d \bmod N, H(\cdot)}(N, e, k)$ where m is an integer
If the following are all true, then return 1 else return 0:
- $\forall i \in \{1, \ldots, m\} : H(M_i) \equiv x_i^e \bmod N$
- Messages $M_1, \ldots, M_m$ are all distinct
- F made strictly fewer than m queries to its RSA-inversion oracle

We define the *advantage* of the forger F via

$$\mathbf{Adv}_F^{\text{rsa-omf}}(k) = \Pr[\,\mathbf{Exp}_F^{\text{rsa-omf}}(k) = 1\,]\,.$$

The FDH-RSA blind signature scheme is said to be *polynomially-secure against one-more forgery* if the function $\mathbf{Adv}_F^{\text{rsa-omf}}(\cdot)$ is negligible for any forger F whose time-complexity is polynomial in the security parameter k. ∎

We need a convention regarding choosing the function H since it is an infinite object. The convention is that we do not actually view it as being chosen all at once,

but rather view it as being built dynamically and stored in a table. Each time a query of M to the hash oracle is made, we charge the cost of the following: check whether a table entry $H(M)$ exists and if so return it; otherwise, pick an element y of $\mathbb{Z}_N^*$ at random, make a table entry $H(M) = y$, and return y. Furthermore, as above, we adopt the convention that the time-complexity refers to the entire experiment. In this regard, the cost of maintaining this table-based implementation of the hash function is included.

Security. We show that the FDH-RSA blind signature scheme is secure as long as the RSA known-target inversion problem is hard.

Theorem 16. *For any forger F attacking the FDH-RSA blind signature scheme, there exists an adversary A for the RSA known-target inversion problem such that*

$$\mathbf{Adv}_F^{\text{rsa-omf}}(k) \;\leq\; O(\ln(k)^2) \cdot \mathbf{Adv}_A^{\text{rsa-akti}}(k) \;,$$

and the time-complexity of A is polynomial in the time-complexity of F. ∎

As a simple corollary, we have:

Corollary 17. *[**Unforgeability of the FDH-RSA blind signature scheme**] If the RSA known-target inversion problem is hard, then the FDH-RSA blind signature scheme is polynomially-secure against one-more forgery.* ∎

Theorem 17 follows from Theorem 9, Theorem 13, Theorem 14, and the following lemma saying that the FDH-RSA blind signature scheme is secure if the RSA *chosen*-target inversion problem is hard.

Lemma 18. *If the RSA chosen-target inversion problem is hard, then the FDH-RSA blind signature scheme is polynomially-secure against one-more forgery. Concretely, for forger F, there exists an adversary B so that*

$$\mathbf{Adv}_F^{\text{rsa-omf}}(k) \;\leq\; \mathbf{Adv}_B^{\text{rsa-acti}}(k)$$

and the time-complexity of B is polynomial in the time-complexity of the forger F. ∎

Proof (Lemma 18). Adversary B uses the forger F to achieve its goal by running F and providing answers to F's oracle queries. In response to hash-oracle queries, B simply returns its own targets to F. RSA-Inversion oracle queries of F are forwarded by B to its own RSA-inversion oracle and the results returned to F.

A detailed description of B is in Figure 4. It uses a subroutine *Find* that looks for a given value in a given array. Specifically, it takes as its inputs an array of values A and a target value a assumed to be in the array, and returns the least index i such that $a = A[i]$.

The simulation is a largely straightforward use of random oracle techniques [7, 8] so we confine the analysis to a few remarks. Note that B simulates hash-oracle queries corresponding to the messages in the message-tag pairs output by F in case these are not already made. This ensures that the advantages of the two algorithms are identical. The time spent by B to maintain the hash-oracle table is the same as that spent in $\mathbf{Exp}_F^{\text{rsa-omf}}(k)$ as per the conventions discussed following Definition 15. We omit the details.

```
Algorithm B^{(·)^d mod N, O_N}(N, e, k)
    count ← 0
    Initialize associative arrays Hash and Ind to empty
    Initialize arrays Msg, X to empty
    Run F on input N, e, k replying to its oracle queries as follows:
        When F submits a hash query M do
            If Hash[M] is undefined then
                count ← count + 1 ; Hash[M] ← O_N ; Msg[count] ← M
            Return Hash[M]
        When F submits an RSA-inversion query y do
            Submit y to the RSA-inversion oracle (·)^d mod N and
            return its response.
    ((M_1, x_1), ..., (M_m, x_m)) ← F
    For j = 1 to m, do
        If Hash[M_j] is undefined then
            count ← count + 1 ; Hash[M_j] ← O_N ; Msg[count] ← M_j
        Ind[j] ← Find(Msg, M_j) ; X[Ind[j]] ← x_j
    Return (Ind, X[Ind[1]], ..., X[Ind[s]])
```

Fig. 4. Adversary B for the proof of Lemma 18.

7 One-More-Discrete-Log: Problems and Results

We describe the discrete-log analogues of the one-more-RSA-inversion problems and then state a theorem analogous to Theorem 9.

The security parameter is denoted by $k \in \mathbb{N}$ as before. A *group generator* is a randomized, poly(k) time algorithm that on input k returns a triple $(\mathbb{G}, q, g)$ such that: $2^{k-1} \leq q < 2^k$; $\mathbb{G}$ is (the description of) a group of order q; and g is a generator of $\mathbb{G}$. Additionally, if q is an odd prime, then we refer to the algorithm as a *prime-order group generator*. Group operations (multiplication and inverse) are assumed to be poly(k)-time. There are numerous possible choices of group generators, so rather than pin one down, we formulate computational problems in which the generator is a parameter. For that purpose, we fix a group generator KeyGen for the rest of this section.

7.1 *Definitions of the Problems*

We recall the standard notion, couching it in a way more suitable for comparison with the new notions.

Definition 19 (Single-Target Discrete Log Problem: DL-ST). Let $k \in \mathbb{N}$ be the security parameter. Let A be an adversary. Consider the following experiment:

$$\begin{array}{l}\text{Experiment } \mathbf{Exp}^{\text{dl-st}}_A(k)\\ \quad (\mathbb{G}, q, g) \xleftarrow{R} \text{KeyGen}(k)\\ \quad x \xleftarrow{R} \mathbb{Z}_q \,;\; y \leftarrow g^x \,;\; z \leftarrow A(\mathbb{G}, q, g, y, k)\\ \quad \text{If } y = g^z \text{ then return 1 else return 0}\end{array}$$

We define the advantage of A via

$$\mathbf{Adv}^{\text{dl-st}}_A(k) = \Pr[\,\mathbf{Exp}^{\text{dl-st}}_A(k) = 1\,]\,.$$

The DL-ST problem —in more standard terminology, the discrete log problem— is said to be *hard* if the function $\mathbf{Adv}^{\text{dl-st}}_{A}(\cdot)$ is negligible for any adversary A whose time-complexity is polynomial in the security parameter k. ▮

We now present the parameterized formulations of the one-more-discrete-logarithm problems in the style of Section 2. Alternative formulations in the style of Section 5 can be easily provided, and can be proved equivalent to the parameterized ones by arguments analogous to those in Section 5. We note that the alternative formulation of the known-target version of the problem is presented and used in [6]. We now proceed to define the known-target inversion problem.

We denote by $\text{dlog}_g(\cdot)$ the oracle that takes input $y \in \mathbb{G}$ and returns its discrete log, namely a value $x \in \mathbb{Z}_q$ such that $y = g^x$. An adversary solving the known-target discrete log problem is given oracle access to $\text{dlog}_g(\cdot)$ and is given $m(k)+1$ targets where $m : \mathbb{N} \to \mathbb{N}$. Its task is to compute the discrete logs of *all* the targets while submitting at most $m(k)$ queries to the oracle.

Definition 20 (Known-Target Discrete Log Problem: DL-KT$[m]$). Let $k \in \mathbb{N}$ be the security parameter, and let $m : \mathbb{N} \to \mathbb{N}$ be a function of k. Let A be an adversary with access to a discrete-log oracle $\text{dlog}_g(\cdot)$. Consider the following experiment:

Experiment $\mathbf{Exp}^{\text{dl-kt}}_{A,m}(k)$
$(\mathbb{G}, q, g) \xleftarrow{R} \text{KeyGen}(k)$
For $i = 1$ to $m(k)+1$ do $x_i \leftarrow \mathbb{Z}_q$; $y_i \leftarrow g^{x_i}$
$(z_1, \ldots, z_{m(k)+1}) \leftarrow A^{\text{dlog}_g(\cdot)}(\mathbb{G}, q, g, k, y_1, \ldots, y_{m(k)+1})$
If the following are both true then return 1 else return 0
– $\forall i \in \{1, \ldots, m(k)+1\} : y_i = g^{z_i}$
– A made at most $m(k)$ oracle queries

We define the advantage of A via

$$\mathbf{Adv}^{\text{dl-kt}}_{A,m}(k) = \Pr[\,\mathbf{Exp}^{\text{dl-kt}}_{A,m}(k) = 1\,]\ .$$

The DL-KT$[m]$ problem is said to be *hard* if the function $\mathbf{Adv}^{\text{dl-kt}}_{A,m}(\cdot)$ is negligible for any adversary A whose time-complexity is polynomial in the security parameter k. The known-target inversion problem is said to be hard if DL-KT$[m]$ is hard for all polynomially-bounded $m(\cdot)$. ▮

Notice that DL-KT[0] is the same as DL-ST. That is, the standard assumption that discrete logarithm problem is hard is exactly the same as the assumption that DL-KT[0] is hard.

We proceed to the chosen-target inversion problem. An adversary solving this problem is given access to a discrete-log oracle as above, and $n(k)$ targets where $n : \mathbb{N} \to \mathbb{N}$. Its task is to compute $m(k)+1$ discrete logs of the given targets, where $m : \mathbb{N} \to \mathbb{N}$ and $m(k) < n(k)$, while submitting at most $m(k)$ queries to the oracle. The choice of which targets to compute the discrete log is up to the adversary. This choice is indicated by the range of the injective map π.

Definition 21 (Chosen-Target Discrete Log Problem: DL-CT$[n, m]$). Let $k \in \mathbb{N}$ be the security parameter, and let $m, n : \mathbb{N} \to \mathbb{N}$ be functions of k such that $m(\cdot) < n(\cdot)$. Let B be an adversary with access to a discrete-log oracle $\text{dlog}_g(\cdot)$. Consider the following experiment:

Experiment $\mathbf{Exp}^{\text{dl-ct}}_{B,n,m}(k)$
$(\mathbb{G}, q, g) \xleftarrow{R} \text{KeyGen}(k)$
For $i = 1$ to $n(k)$ do $\overline{x}_i \xleftarrow{R} \mathbb{Z}_q$; $\overline{y}_i \leftarrow g^{\overline{x}_i}$
$(\pi, \overline{z}_1, \ldots, \overline{z}_{m(k)+1}) \leftarrow B^{\text{dlog}_g(\cdot)}(\mathbb{G}, q, g, k, \overline{y}_1, \ldots, \overline{y}_{n(k)})$
If the following are all true then return 1 else return 0
- $\pi\colon \{1, \ldots, m(k)+1\} \to \{1, \ldots, n(k)\}$ is injective
- $\forall i \in \{1, \ldots, m(k)+1\} : \overline{y}_i = g^{\overline{z}_{\pi(i)}}$
- A made at most $m(k)$ oracle queries

We define the advantage of B via

$$\mathbf{Adv}^{\text{dl-ct}}_{B,n,m}(k) = \Pr[\,\mathbf{Exp}^{\text{dl-ct}}_{B,n,m}(k) = 1\,] \,.$$

The DL-CT$[n, m]$ problem is said to be *hard* if the function $\mathbf{Adv}^{\text{dl-ct}}_{B,n,m}(\cdot)$ is negligible for any adversary B whose time complexity is polynomial in the security parameter k. The chosen-target discrete log problem is said to be hard if DL-CT$[n, m]$ is hard for all polynomially-bounded $n(\cdot)$ and $m(\cdot)$. ▮

7.2 *The Equivalence Result*

The following theorem is the analogue of Corollary 10.

Theorem 22. *Let $n, m\colon \mathbb{N} \to \mathbb{N}$ be polynomially-bounded functions satisfying $m(\cdot) < n(\cdot)$. Then the DL-KT$[m]$ problem is hard if and only if the DL-CT$[n, m]$ problem is hard.* ▮

Proof (Theorem 22). It is easy to see that if the DL-KT$[m]$ problem is hard then so is the DL-CT$[n, m]$ problem. We concentrate on the converse. For any adversary B, we show that there exists an adversary A so that

$$\mathbf{Adv}^{\text{dl-ct}}_{B,n,m}(k) \;\le\; O(\ln(k)^2) \cdot \mathbf{Adv}^{\text{dl-kt}}_{A,m}(k) \tag{14}$$

and A has time-complexity equal to that of B plus poly(k). Furthermore, if KeyGen is a prime-order group generator, then Equation (14) can be improved to

$$\mathbf{Adv}^{\text{dl-ct}}_{B,n,m}(k) \;\le\; \frac{9}{5} \cdot \mathbf{Adv}^{\text{dl-kt}}_{A,m}(k) \,. \tag{15}$$

The adversary A takes $\mathbb{G}, q, g,$ and $s = m(k) + 1$ target points $y_1, \ldots, y_s$. Its goal is to compute $x_1, \ldots, x_s$ such that $y_i = g^{x_i}$ for any $i \in \{1, \ldots, s\}$. The proof follows the same approach as that of Theorem 9. Figure 5 describe the adversary A in detail. We denote the adversary solving the RSA-CTI$[n, m]$ problem by B. The analysis is similar to that in the proof of Theorem 9. As before, let NS be the event that $\gcd(\det(C), q) = 1$. Let "A succeeds" denote the event that $y_i = g^{x_i}$ for all $i = 1, \ldots, s$. Let "B succeeds" denote the event that $\overline{y}_j = g^{\overline{x}_{\pi(j)}}$ for all $j = 1, \ldots, s$. Then,

$$\begin{aligned}
\Pr[\,A \text{ succeeds}\,] &\ge \Pr[\,A \text{ succeeds} \wedge B \text{ succeeds} \wedge \text{NS}\,] \\
&= \Pr[\,A \text{ succeeds} \mid B \text{ succeeds} \wedge \text{NS}\,] \cdot \Pr[\,B \text{ succeeds} \wedge \text{NS}\,] \\
&= 1 \cdot \Pr[\,B \text{ succeeds} \wedge \text{NS}\,] \\
&= \Pr[\,B \text{ succeeds}\,] \cdot \Pr[\,\text{NS}\,] \\
&= \Pr[\,\text{NS}\,] \cdot \mathbf{Adv}^{\text{dl-ct}}_{B,n,m}(k) \,.
\end{aligned}$$

If KeyGen is a prime-order group generator, then we note that $q \ge 3$ and apply Equation (6) from Lemma 8 to obtain Equation (15) as follows.

$$\Pr[\,\text{NS}\,] \;=\; \mathsf{GLP}(s, q) \;\ge\; 1 - \frac{1}{q} - \frac{1}{q^2} \;\ge\; 1 - \frac{1}{3} - \frac{1}{3^2} \;=\; \frac{5}{9} \,.$$

Algorithm $A^{\mathrm{dlog}_g(\cdot)}(\mathbb{G}, q, g, k, y_1, \ldots, y_{m(k)+1})$
$s \leftarrow m(k) + 1$
For $i = 1, \ldots, n(k)$ do
  $v[i] \stackrel{R}{\leftarrow} \mathbb{Z}_q$
  For $j = 1, \ldots, s$ do $c[j, i] \stackrel{R}{\leftarrow} \mathbb{Z}_q$
  $\overline{y}_i \leftarrow g^{v[i]} \prod_{j=1}^{s} y_j^{c[j,i]}$
$(\pi, \overline{x}_1, \ldots, \overline{x}_s) \leftarrow B^{\mathrm{dlog}_g(\cdot)}(\mathbb{G}, q, g, k, \overline{y}_1, \ldots, \overline{y}_{n(k)})$
For $j = 1, \ldots, s$ do
  $v_j \leftarrow v[\pi(j)]$
  For $l = 1, \ldots, s$ do $c_{j,l} \leftarrow c[j, \pi(l)]$

$$C \leftarrow \begin{bmatrix} c_{1,1} & \ldots & c_{1,s} \\ \vdots & & \vdots \\ c_{s,1} & \ldots & c_{s,s} \end{bmatrix}$$

$\alpha \leftarrow \det(C)$
If $\alpha = 0$ then abort
Compute a matrix

$$D = \begin{bmatrix} d_{1,1} & \ldots & d_{1,s} \\ \vdots & & \vdots \\ d_{s,1} & \ldots & d_{s,s} \end{bmatrix}$$

with integer entries such that $C \cdot D = \det(C) \cdot I_s$
For $j = 1, \ldots, s$ do
  $z_j \leftarrow \sum_{i=1}^{s} (\overline{x}_i - v_i) \cdot d_{i,j} \bmod q$
If $\gcd(\alpha, q) \neq 1$ then abort
$\beta \leftarrow \alpha^{-1} \bmod q$
For $j = 1, \ldots, s$ do
  $x_j \leftarrow \beta \cdot z_j \bmod q$
Return $x_1, \ldots, x_s$

Fig. 5. Adversary A of the proof of Theorem 22.

If KeyGen is not a prime-order group generator, then we apply Equation (5) from Lemma 8 to obtain Equation (14).

We have not stated the concrete-security of the reduction, but it is worth making a remark about how concrete-security is measured. Recall that in the RSA case, the time-complexity of an adversary was defined as that of the entire associated experiment, including the time to compute replies to oracle queries. With regard to the decryption oracle, this meant cubic time, since the computation is done by an oracle in possession of the trapdoor information d. This convention regarding measurement of time-complexity must be dropped in the discrete-logarithm case, since the time to compute discrete-logarithms is exponential, and the complexity estimates end up being meaningless if we adopt the same convention as we did for RSA.

Acknowledgments

We thank Keith Conrad for ideas that lead to Lemma 8. We also thank the anonymous referees for the Financial Cryptography 2001 conference, the anonymous referees for the Journal of Cryptology, and Matt Franklin, as the Journal of Cryptology editor who handled our paper.

References

1. J. Abbott, M. Bronstein, and T. Mulders. Fast deterministic computation of determinants of dense matrices. In *Proceedings of ACM International Symposium on Symbolic and Algebraic Computation*, pages 197–204. ACM Press, 1999.
2. W. Adkins and S. Weintraub. *Algebra: An Approach Via Module Theory*, volume 136 of *Graduate Texts in Mathematics*, chapter 4, page 204. Springer-Verlag, Berlin Germany, 1992.
3. N. Barić and B. Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In W. Fumy, editor, *Advances in Cryptology – EUROCRYPT ' 97*, volume 1233 of *Lecture Notes in Computer Science*, pages 480–494. Springer-Verlag, Berlin Germany, May 1997.
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 P. Syverson, editor, *Fifth International Conference on Financial Cryptography – FC ' 01*, volume 2339 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin Germany, February 2002.
5. M. Bellare and G. Neven. Transitive signatures based on factoring and RSA. Manuscript, May 2002.
6. M. Bellare and A. Palacio. GQ and Schnorr identification schemes: Proofs of security against impersonation under active and concurrent attack. In M. Yung, editor, *Advances in Cryptology – CRYPTO ' 02*, Lecture Notes in Computer Science. Springer-Verlag, Berlin Germany, August 2002.
7. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In V. Ashby, editor, *1st ACM Conference on Computer and Communications Security*. ACM Press, November 1993.
8. M. Bellare and P. Rogaway. The exact security of digital signatures—how to sign with RSA and Rabin. In U. Maurer, editor, *Advances in Cryptology – EUROCRYPT ' 96*, volume 1070 of *Lecture Notes in Computer Science*, pages 399–416. Springer-Verlag, Berlin Germany, 12–16 May 1996.
9. M. Bellare and R. Sandhu. The security of practical two-party RSA signature schemes. Cryptology ePrint Archive: Report 2001/060, July 2001. Available via `http://eprint.iacr.org/2001/060/`.
10. D. Boneh. Twenty years of attacks on the RSA cryptosystem. *Notices of the American Mathematical Society*, 46(2):203–213, February 1999.
11. D. Boneh and R. Venkatesan. Breaking RSA may not be equivalent to factoring. In K. Nyberg, editor, *Advances in Cryptology – EUROCRYPT ' 98*, volume 1233 of *Lecture Notes in Computer Science*, pages 59–71. Springer-Verlag, Berlin Germany, 1998.
12. D. Chaum. Blind signatures for untraceable payments. In D. Chaum, R. Rivest, and A. Sherman, editors, *Advances in Cryptology – CRYPTO ' 82*, Lecture Notes in Computer Science, pages 199–203. Plenum Press, New York and London, 1983, August 1982.
13. J.S. Coron. On the exact security of full domain hash. In M. Bellare, editor, *Advances in Cryptology – CRYPTO ' 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 229–235. Springer-Verlag, Berlin Germany, August 2000.
14. R. Cramer and V. Shoup. Signature schemes based on the strong RSA assumption. In *Proceedings of the 6th ACM conference on Computer and communications security*, pages 46–51. ACM Press, November 1999.
15. W. Eberly, M. Giesbrecht, and G. Villard. Computing the determinant and Smith form of an integer matrix. In IEEE, editor, *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, pages 675–685. IEEE, IEEE Computer Society Press, 12–14 November 2000.
16. E. Fujisaki and T. Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In B. Kaliski Jr., editor, *Advances in Cryptology – CRYPTO ' 97*, volume 1294 of *Lecture Notes in Computer Science*, pages 16–30. Springer-Verlag, Berlin Germany, 17–21 August 1997.
17. R. Gennaro, S. Halevi, and T. Rabin. Secure hash-and-sign signatures without the random oracle. In J. Stern, editor, *Advances in Cryptology – EUROCRYPT ' 99*, volume 1592 of *Lecture Notes in Computer Science*, pages 123–139. Springer-Verlag, Berlin Germany, May 1999.
18. S. Goldwasser, S. Micali, and R. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, April 1988. Special issue on cryptography.
19. L. Guillou and J. J. Quisquater. A "paradoxical" identity-based signature scheme resulting from zero-knowledge. In S. Goldwasser, editor, *Advances in Cryptology – CRYPTO ' 88*, volume 403 of *Lecture Notes in Computer Science*, pages 216–231. Springer-Verlag, Berlin Germany, 1989.
20. A. Juels, M. Luby, and R. Ostrovsky. Security of blind digital signatures. In B. Kaliski Jr., editor, *Advances in Cryptology – CRYPTO ' 97*, volume 1294 of *Lecture Notes in Computer Science*, pages 150–164. Springer-Verlag, Berlin Germany, 17–21 August 1997.

21. J. Kilian and E. Petrank. Identity escrow. In H. Krawczyk, editor, *Advances in Cryptology – CRYPTO ' 98*, volume 1462 of *Lecture Notes in Computer Science*, pages 169–185. Springer-Verlag, Berlin Germany, 1998.
22. B. McDonald. *Linear Algebra over Commutative Rings*, volume 87 of *Pure and Applied Mathematics*, chapter I, pages 10, 56. Marcel Dekker, Inc., 1984.
23. S. Micali and R. Rivest. Transitive signature schemes. In B. Preneel, editor, *Topics in Cryptology – CT-RSA ' 02*, volume 2271 of *Lecture Notes in Computer Science*, pages 236–243. Springer-Verlag, Berlin Germany, 2002.
24. M. Michels, M. Stadler, and H. Sun. The security of some variants of the RSA signature scheme. In Y. Deswarte, editor, *Computer Security – ESORICS ' 98*, volume 1485 of *Lecture Notes in Computer Science*, pages 85–96. Springer-Verlag, Berlin Germany, 1998.
25. D. Pointcheval. Strengthened security for blind signatures. In K. Nyberg, editor, *Advances in Cryptology – EUROCRYPT '98*, volume 1403 of *Lecture Notes in Computer Science*, pages 391–405. Springer-Verlag, Berlin Germany, 31–4 June 1998.
26. D. Pointcheval. New public key cryptosystems based on the dependent-RSA problems. In J. Stern, editor, *Advances in Cryptology – EUROCRYPT '99*, volume 1592 of *Lecture Notes in Computer Science*, pages 239–255. Springer-Verlag, Berlin Germany, 1999.
27. D. Pointcheval and J. Stern. Provably secure blind signature schemes. In K. Kim and T. Matsumoto, editors, *Advances in Cryptology – ASIACRYPT ' 96*, volume 1163 of *Lecture Notes in Computer Science*, pages 252–265. Springer-Verlag, Berlin Germany, 1996.
28. D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, 2000.
29. J. B. Rosser and L. Schoenfeld. Approximate formulas for some functions of prime numbers. *Illinois J. Math.*, 6:64–94, 1962.
30. C. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.

A Proof of Lemma 8

Let $q \geq 2$ and $s \geq 1$ be integers. We define the set of s by s matrices over the ring $\mathbb{Z}_q$ and denote it by $\mathrm{M}_s(q)$. This is the set of all s by s matrices with entries in $\mathbb{Z}_q$. The matrices in $\mathrm{M}_s(q)$ are added and multiplied modulo q, meaning that the underlying computations on entries are performed modulo q. A matrix $A \in \mathrm{M}_s(q)$ is said to be *invertible* if there exists a matrix $B \in \mathrm{M}_s(q)$ such that $AB = BA = I_s$. We let $\mathrm{GL}_s(q)$ denote the set of all invertible matrices in $\mathrm{M}_s(q)$. This is the so-called *general linear* group under matrix multiplication modulo q.

We will state various linear algebraic facts without proof. The first of these is that a matrix over a ring is invertible if and only if its determinant is invertible in the underlying ring [2]. Since the ring here is $\mathbb{Z}_q$, this in turn means that a matrix over a ring is invertible if and only if its determinant is relatively prime to q.

Fact A1 [2] Let $q \geq 2$ and $s \geq 1$ be integers. Then,

$$\mathrm{GL}_s(q) = \{\, M \in \mathrm{M}_s(q) \;:\; \gcd(\det(M), q) = 1 \,\} \,. \;\blacksquare$$

We are interested in the probability $\mathsf{GLP}(s, q)$ that $\gcd(\det(M), q) = 1$ when M is chosen uniformly at random from $\mathrm{M}_s(q)$. We first apply Fact A1 to see that

$$\mathsf{GLP}(s, q) \;=\; \frac{|\mathrm{GL}_s(q)|}{|\mathrm{M}_s(q)|} \,.$$

The denominator above is easy to compute. We now turn to estimating the numerator. This will require a few more facts. We say that $q = q_1^{n_1} \cdots q_k^{n_k}$ is the *prime factorization* of q if $2 \leq q_1 < q_2 < \cdots < q_k$ are primes and $n_1, \ldots, n_k \geq 1$ are integers.

Fact A2 [22] Let $q \geq 2$ and $s \geq 1$ be integers, and let $q = q_1^{n_1} \cdots q_k^{n_k}$ be the prime factorization of q. Then,

$$\mathrm{GL}_s(q) \;\simeq\; \mathrm{GL}_s(q_1^{n_1}) \times \cdots \times \mathrm{GL}_s(q_k^{n_k}) \,,$$

where $\simeq$ denotes group isomorphism. ∎

The above reduces the computation of $\mathrm{GL}_s(q)$ to computation of the same quantity for the case where q is a prime power. The following fact gives the exact number of elements for the latter.

Fact A3 [22] Let $q \geq 2$ be a prime and let $s, n \geq 1$ be integers. Then,

$$|\mathrm{GL}_s(q^n)| \;=\; \begin{cases} q^{s^2} \cdot \prod_{j=0}^{s-1} \left(1 - q^{j-s}\right) & \text{if } n = 1 \\ q^{s^2(n-1)} \cdot |\mathrm{GL}_s(q)| & \text{otherwise.} \end{cases} \quad \blacksquare \tag{16}$$

When q is a prime, it is tempting to think that the determinant of a random matrix from $\mathrm{M}_s(q)$ is a random value and hence that $\mathsf{GLP}(s, q) = 1 - 1/q$. This, however, is not true. For example, a simple computation shows that $\mathsf{GLP}(2, q) = 1 - 1/q - 1/q^2 + 1/q^3$ for any prime q. The following lemma provides bounds for the probability of the complement event, namely the probability $\mathsf{SMP}(s, q)$ that $\gcd(\det(M), q) \neq 1$ when M is chosen uniformly at random from $\mathrm{M}_s(q)$. This is the probability that a matrix chosen at random from $\mathrm{M}_s(q)$ is not invertible modulo q.

Lemma 23. *Let $q \geq 2$ be a prime and let $s \geq 1$ be an integer. Then,*

$$\frac{1}{q} \;\leq\; \mathsf{SMP}(s, q) \;\leq\; \frac{1}{q} + \frac{1}{q^2} \,.$$

Recall that $\varphi(\cdot)$ denotes the Euler Phi function. The following lower bound is a corollary of Rosser and Schoenfeld [29, Theorem 15].

Fact A4 [29] Let $n \geq 3$ be an integer. Then,

$$\frac{\varphi(n)}{n} \;\geq\; \frac{1}{286} \cdot \frac{1}{\ln \ln(n)} \,.$$

We conclude the proof of Lemma 8 given the above, and then return to prove Lemma 23.

Proof (Lemma 8). Let $q = q_1^{n_1} \cdots q_k^{n_k}$ be the prime factorization of q. Below, we obtain Equation (18) by Fact A1; Equation (19) by Fact A2; Equation (20) by Fact A3; and Equation (21) by Lemma 23:

$$\begin{aligned}
\mathsf{GLP}(s, q) &= \frac{|\mathrm{GL}_s(q)|}{|\mathrm{M}_s(q)|} && (18) \\
&= \frac{\left|\mathrm{GL}_s(q_1^{n_1}) \times \cdots \times \mathrm{GL}_s(q_k^{n_k})\right|}{q^{s^2}} && (19) \\
&= \frac{\prod_{i=1}^{k} |\mathrm{GL}_s(q_i^{n_i})|}{\prod_{i=1}^{k} q_i^{n_i s^2}} = \prod_{i=1}^{k} \frac{|\mathrm{GL}_s(q_i^{n_i})|}{q_i^{n_i s^2}} \\
&= \prod_{i=1}^{k} \frac{q_i^{(n_i-1)s^2} \cdot |\mathrm{GL}_s(q_i)|}{q_i^{n_i s^2}} && (20) \\
&= \prod_{i=1}^{k} \frac{|\mathrm{GL}_s(q_i)|}{q_i^{s^2}} = \prod_{i=1}^{k} \mathsf{GLP}(s, q_i) \geq \prod_{i=1}^{k} \left(1 - \frac{1}{q_i} - \frac{1}{q_i^2}\right) \,. && (21)
\end{aligned}$$

If q is prime, the lower bound is $1 - \frac{1}{q} - \frac{1}{q^2}$ as claimed. If q is not prime, we derive a lower bound as follows. The inequality $1 - x - x^2 \geq (1-x)^2$ is valid for all real numbers $x \leq 1/2$. Above $q_i \geq 2$, and hence $1/q_i \leq 1/2$, and hence we get

$$\mathsf{GLP}(s,q) \;\geq\; \prod_{i=1}^{k}\left(1-\frac{1}{q_i}\right)^2 \;=\; \left[\prod_{i=1}^{k}\left(1-\frac{1}{q_i}\right)\right]^2 . \tag{22}$$

Let $n = q_1 \cdots q_k$. Applying Fact A4 and letting $\lg(\cdot)$ denote the logarithm to base two, we have

$$\begin{aligned}\prod_{i=1}^{k}\left(1-\frac{1}{q_i}\right) &= \frac{\varphi(n)}{n} \\ &\geq \frac{1}{286}\cdot\frac{1}{\ln\ln(n)} \geq \frac{1}{286}\cdot\frac{1}{\ln\ln(q)} \geq \frac{1}{286}\cdot\frac{1}{\ln\lg(q)} .\end{aligned}$$

Combining the above with Equation (22) completes the proof of Lemma 8.

Proof (Lemma 23). Let $M \in \mathrm{M}_s(q)$. For $i = 1, \ldots, s$, let M_i denote the vector which is the i-th column of M, and let LI_i denote the event that the vectors $M_1, \ldots, M_i$ are linearly independent over $\mathbb{Z}_q$. It is convenient to let LI_0 be the event having probability one. Let $\mathsf{SMP}(s,q,i) = \Pr[\,\neg\mathrm{LI}_i\,]$ for $i = 0, \ldots, s$, and note that $\mathsf{SMP}(s,q) = \mathsf{SMP}(s,q,s)$. Fact A3 implies that

$$\mathsf{GLP}(s,q) \;=\; \prod_{j=0}^{s-1}\left(1-q^{j-s}\right) . \tag{23}$$

We use it to derive the desired lower bound. (The upper bound is derived by a separate inductive argument.) Upper bounding the product term of Equation (23) by a single, smallest term for the product, we obtain

$$\mathsf{SMP}(s,q) \geq 1 - \left(1-\frac{1}{q}\right) = \frac{1}{q} .$$

For the upper bound, we first claim that the following recurrence is true for $i = 0, \ldots, s$:

$$\mathsf{SMP}(s,q,i) \;=\; \begin{cases} 0 & \text{if } i = 0 \\ \dfrac{q^{i-1}}{q^s} + \left(1-\dfrac{q^{i-1}}{q^s}\right)\cdot\mathsf{SMP}(s,q,i-1) & \text{if } i \geq 1 \end{cases} \tag{24}$$

The initial condition is simply by the convention we adopted that $\Pr[\,\mathrm{LI}_0\,] = 1$. The recurrence is justified as follows for $i \geq 1$:

$$\begin{aligned}&\mathsf{SMP}(s,q,i) \\ &= \Pr[\,\neg\mathrm{LI}_i\,] \\ &= \Pr\left[\,\neg\mathrm{LI}_i \mid \mathrm{LI}_{i-1}\,\right]\cdot\Pr[\,\mathrm{LI}_{i-1}\,] + \Pr\left[\,\neg\mathrm{LI}_i \mid \neg\mathrm{LI}_{i-1}\,\right]\cdot\Pr[\,\neg\mathrm{LI}_{i-1}\,] \\ &= \Pr\left[\,\neg\mathrm{LI}_i \mid \mathrm{LI}_{i-1}\,\right]\cdot(1-\mathsf{SMP}(s,q,i-1)) + 1\cdot\mathsf{SMP}(s,q,i-1) \\ &= \Pr\left[\,\neg\mathrm{LI}_i \mid \mathrm{LI}_{i-1}\,\right] + (1-\Pr\left[\,\neg\mathrm{LI}_i \mid \mathrm{LI}_{i-1}\,\right])\cdot\mathsf{SMP}(s,q,i-1) \\ &= \frac{q^{i-1}}{q^s} + \left(1-\frac{q^{i-1}}{q^s}\right)\cdot\mathsf{SMP}(s,q,i-1) .\end{aligned}$$

We claim that

$$\mathsf{SMP}(s,q,i) \;\leq\; \frac{q^i}{q^s}\cdot\frac{1}{q-1} \qquad \text{for } i = 0, \ldots, s . \tag{25}$$

This will be justified below. It already gives us an upper bound on $\mathsf{SMP}(s,q) = \mathsf{SMP}(s,q,s)$, namely $1/(q-1)$, but this is a little worse than our claimed upper bound. To get the latter, we use the recurrence for $i = s$ and use Equation (25) with $i = s-1$. This give us

$$\begin{aligned}\mathsf{SMP}(s,q) \;=\; \mathsf{SMP}(s,q,s) &= \frac{q^{s-1}}{q^s} + \left(1 - \frac{q^{s-1}}{q^s}\right) \cdot \mathsf{SMP}(s,q,s-1) \\ &\leq \frac{q^{s-1}}{q^s} + \left(1 - \frac{q^{s-1}}{q^s}\right) \cdot \frac{q^{s-1}}{q^s} \cdot \frac{1}{q-1}\end{aligned}$$

Simplifying this further, we get

$$\mathsf{SMP}(s,q) \;\leq\; \frac{1}{q} + \left(1 - \frac{1}{q}\right) \cdot \left(\frac{1}{q}\right) \cdot \left(\frac{1}{q-1}\right) \;=\; \frac{1}{q} + \left(\frac{q-1}{q}\right) \cdot \left(\frac{1}{q^2} - \frac{1}{q}\right) \;\leq\; \frac{1}{q} + \frac{1}{q^2}\,.$$

This is the claimed upper bound. It remains to justify Equation (25) which we do by induction on i. When $i = 0$, Equation (25) puts a positive upper bound on $\mathsf{SMP}(s,q,0)$, and hence is certainly true. So assume $i \geq 1$. Substituting into the recurrence of Equation (24), we get

$$\begin{aligned}\mathsf{SMP}(s,q,i) &= \frac{q^{i-1}}{q^s} + \left(1 - \frac{q^{i-1}}{q^s}\right) \cdot \mathsf{SMP}(s,q,i-1) \\ &\leq \frac{q^{i-1}}{q^s} + \mathsf{SMP}(s,q,i-1)\,.\end{aligned}$$

Using the inductive hypothesis and simplifying, we have

$$\mathsf{SMP}(s,q,i) \;\leq\; \frac{q^{i-1}}{q^s} + \frac{q^{i-1}}{q^s}\frac{1}{q-1} \;=\; \frac{q^{i-1}}{q^s}\left(1 + \frac{1}{q-1}\right) \;=\; \frac{q^i}{q^s} \cdot \frac{1}{q-1}$$

as desired. ∎