Another Look at "Provable Security". II

Neal Koblitz[1] and Alfred Menezes[2]

[1] Department of Mathematics, University of Washington
koblitz@math.washington.edu
[2] Department of Combinatorics & Optimization, University of Waterloo
ajmeneze@uwaterloo.ca

Abstract. We discuss the question of how to interpret reduction arguments in cryptography. We give some examples to show the subtlety and difficulty of this question.

1 Introduction

Suppose that one wants to have confidence in the security of a certain cryptographic protocol. In the "provable security" paradigm, the ideal situation is that one has a tight reduction (see §4 for a definition and discussion of tightness) from a mathematical problem that is widely believed to be intractable to a successful attack (of a prescribed type) on the protocol. This means that an adversary who can attack the system must also be able to solve the (supposedly intractable) problem in essentially the same amount of time with essentially the same probability of success. Often, however, the best that researchers have been able to achieve falls short of this ideal. Sometimes reductionist security arguments have been found for modified versions of the protocol, but not for the actual protocol that is used in practice; or for a modified version of the type of attack, but not for the security definition that people really want; or based on a somewhat contrived and unnatural modified version of the mathematical problem that is believed to be hard, but not based on the actual problem that has been extensively studied. In other cases, an asymptotic result is known that cannot be applied to specific parameters without further analysis. In still other cases, one has a reduction, but one can show that there cannot be (or is unlikely to be) a tight reduction.

In this paper we give examples that show the subtle questions that arise when interpreting reduction arguments in cryptography.

2 Equivalence But No Reductionist Proof

In [13], Boneh and Venkatesan showed that an efficient reduction from factoring to the RSA problem (the problem of inverting the function $y = x^e \bmod N$) is unlikely to exist. More precisely, they proved that for small encryption exponent e the existence of an efficient "algebraic" reduction would imply that factoring is easy.

R. Barua and T. Lange (Eds.): INDOCRYPT 2006, LNCS 4329, pp. 148–175, 2006.

The paper [13] appeared at a time of intense rivalry between RSA and elliptic curve cryptography (ECC). As enthusiastic advocates of the latter, we were personally delighted to see the Boneh–Venkatesan result, and we welcomed their interpretation of it — that, in the words of their title, "breaking RSA may not be equivalent to factoring" — as another nail in the coffin of RSA.

However, to be honest, another interpretation is at least as plausible. Both factoring and the RSA problem have been studied intensively for many years. In the general case no one has any idea how to solve the RSA problem without factoring the modulus. Just as our experience leads us to believe that factoring (and certain other problems, such as the elliptic curve discrete logarithm problem) are hard, so also we have good reason to believe that, in practice, the RSA problem *is* equivalent to factoring. Thus, an alternative interpretation of the Boneh–Venkatesan result is that it shows the limited value of reduction arguments, and an alternative title of the paper [13] would have been "Absence of a reduction between two problems may not indicate inequivalence."

Which interpretation one prefers is a matter of opinion, and that opinion may be influenced, as in our own case, by one's biases in favor of or against RSA.

3 Results That Point in Opposite Directions

3.1 Reverse Boneh–Venkatesan

A recent result [16] by D. Brown can be seen as giving support to the alternative interpretation of Boneh–Venkaesan that we described at the end of §2. For small encryption exponents e,[1] Brown proves that if there is an efficient program that, given the RSA modulus N, constructs a straight-line program that efficiently solves the RSA problem,[2] then the program can also be used to efficiently factor N. This suggests that for small e the RSA problem may very well be equivalent to factoring. If one believes this interpretation, then one might conclude that small e are more secure than large e. In contrast, the result of Boneh–Venkatesan could be viewed as suggesting that large values of e are more secure than small ones.

As Brown points out in §5 of [16], his result does not actually contradict Boneh–Venkatesan. His reduction of factoring to a straight-line program for finding e-th roots does not satisfy the conditions of the reductions treated in [13]. His use of the e-th root extractor cannot be modeled by an RSA-oracle, as required in [13], because he applies the straight-line program to ring extensions of $\mathbb{Z}/N\mathbb{Z}$.[3]

Brown's choice of title is a helpful one: "Breaking RSA may be as difficult as factoring." All one has to do is put it together in a disjunction with the title of [13], and one has a statement that cannot lead one astray, and accurately summarizes what is known on the subject.

[1] Brown's result actually applies if e just has a small prime factor.

[2] This essentially means that it constructs a polynomial that inverts the encryption function.

[3] For example, when $e = 3$ the polynomial that inverts cube roots is applied to the ring $\mathbb{Z}/N\mathbb{Z}[X]/(X^2 - u)$, where the Jacobi symbol $\left(\frac{u}{N}\right) = -1$.

3.2 Random Padding Before or After Hashing?

When comparing ElGamal-like signature schemes, one finds that some, such as Schnorr signatures [35], append a random string to the message before evaluating the hash function; and some, such as the Digital Signature Algorithm (DSA) and the Elliptic Curve Digital Signature Algorithm (ECDSA), apply the hash function before the random padding. Is it more secure to do the padding before or after hashing? What do the available "provable security" results tell us about this question?

As we discussed in §5.2 of [27], the proof that forgery of Schnorr signatures is equivalent to solving the discrete log problem (see the sketch in §5.1 of [27] and §8.3 below, and the detailed proof in [33,34]) relies in an essential way on the fact that an attacker must choose the random r before making his hash query. For this reason, the proof does not carry over to DSA, where only the message m and not r is hashed. In §5.2 of [27] we commented that

> ...replacing $H(m, r)$ by $H(m)$ potentially gives more power to a forger, who has control over the choice of k (which determines r) but no control over the (essentially random) hash value. If H depends on r as well as m, the forger's choice of k must come before the determination of the hash value, so the forger doesn't "get the last word."

That was our attempt to give an intuitive explanation of the circumstance that in the random oracle model Schnorr signatures, unlike the closely related DSA signatures, have been tied to the discrete logarithm problem (DLP) through a reduction argument. One could conclude from our comment that it's more secure to do the padding before hashing.

However, we were very much at fault in misleading the reader in this way. In fact, there is another provable security result, due to D. Brown [14,15], that points in the opposite direction. It says: *If the hash function and pseudorandom bit generator satisfy certain reasonable assumptions, then ECDSA is secure against chosen-message attack by a universal forger*[4] *provided that the "adaptive semi-logarithm problem" in the elliptic curve group is hard.*[5] Brown comments in [15] that his security reduction would not work for a modification of ECDSA in which r as well as the message m is hashed. Brown does not claim that the modified version is therefore less secure than the original version of ECDSA with only the message hashed. However, in an informal communication [17] he explained how someone might make such a claim: namely, the inclusion of a random r along with m in the input could be viewed as "giving an attacker extra play

[4] A forger is *universal* (or *selective* in Brown's terminology) if it can forge an arbitrary message that it is given.

[5] A semi-logarithm of a point Q with respect to a basepoint P of prime order p is a pair (t, u) of integers mod p such that $t = f(u^{-1}(P + tQ))$, where the "conversion function" f is the map from points to integers mod p that is used in ECDSA. The adaptive semi-logarithm problem is the problem of finding a semi-logarithm of Q to the base P given an oracle that can find a semi-logarithm of Q to any base of the form eP with $e \neq 1$.

with the hash function," and this could lead to a breach. (But note that both the results in [33,34] and in [14,15] assume that the hash function is strong.)

Once again we have provable security results that suggest opposite answers to a simple down-to-earth question. Is it better to put in the random padding before or after evaluating the hash function? As in the case of the question in §3.1, both answers "before" and "after" can be supported by reduction arguments.

In §8 we shall discuss another question — whether or not forgery of Schnorr-type signatures is equivalent to the DLP — for which different provable security results give evidence for opposite answers.

4 Non-tightness in Reductions

We first give an informal definition of tightness of a reduction. Suppose that we have an algorithm for solving problem $\mathcal{A}$ that takes time at most T and is successful for a proportion at least ϵ of the instances of $\mathcal{A}$, where T and ϵ are functions of the input length. A reduction from a problem $\mathcal{B}$ to $\mathcal{A}$ is an algorithm that calls upon the algorithm for $\mathcal{A}$ a certain number of times and solves $\mathcal{B}$ in time T' for at least a proportion ϵ' of the instances of $\mathcal{B}$. This reduction is said to be *tight* if $T' \approx T$ and $\epsilon' \approx \epsilon$. Roughly speaking, it is *non-tight* if $T' \gg T$ or if $\epsilon' \ll \epsilon$.

Suppose that researchers have been able to obtain a highly non-tight reduction from a hard mathematical problem to breaking a protocol. There are various common ways to respond to this situation:

1. Even a non-tight reduction is better than nothing at all. One should regard the cup as half-full rather than half-empty, derive some reassurance from what one has, and try not to think too much about what one wishes one had.[6]
2. Even though the reduction is not tight, it is reasonable to expect that in the future a tighter reduction will be found.
3. Perhaps a tight reduction cannot be found for the protocol in question, but a small modification of the protocol can be made in such a way as to permit the construction of a tight reduction — and we should regard this reduction as a type of assurance about the original protocol.
4. A tight reduction perhaps can be obtained by relaxing the underlying hard problem (for example, replacing the computational Diffie–Hellman problem by the decision Diffie–Hellman problem).
5. Maybe the notion of security is too strict, and one should relax it a little so as to make possible a tight reduction.

[6] We are reminded of the words of the popular song
If you can't be with the one you love,
Love the one you're with,
(*Stephen Stills*, 1970). The version for cryptographers is:
If you can't prove what you'd love to prove,
Hype whatever you prove.

6. Perhaps the protocol is secure in practice, even though a tight reduction may simply not exist.
7. Perhaps the protocol is in fact insecure, but an attack has not yet been discovered.

These seven points of view are not mutually exclusive. In fact, protocol developers usually adopt some combination of the first six interpretations — but generally not the seventh.

4.1 Insecure But Provably Secure: An Example

We now give an example that is admittedly somewhat artificial. Let us step into a time machine and go back about 25 years to a time when naive index-calculus was pretty much the best factoring algorithm. Let us also suppose that 2^{2a} operations are feasible, but $2^{(2\sqrt{2})a}$ operations are not.

Let N be a c-bit RSA modulus, and let r be an a-bit integer. Let $F = \{p_1, \ldots, p_r\}$ be a factor base consisting of the first r primes. Let 2^b be the expected time needed before a randomly selected x mod N has the property that x^2 mod N is p_r-smooth (this means that it has no prime factors greater than p_r). The usual estimate is that $2^b \approx u^u$, where $u = c/a$. (Actually, it's more like $u = c/(a + \log(a \ln 2))$, where log denotes $\log_2$, but let's ignore second-order terms.)

If x has the property that x^2 mod N is p_r-smooth, then by its "exponent-vector" we mean the vector in $\mathbb{F}_2^r$ whose components ϵ_i are the exponents of p_i in the squarefree part of x^2 mod N.

The basic (naive) index-calculus algorithm involves generating roughly r such x values and then solving an $r \times r$-matrix over $\mathbb{F}_2$. The first part takes roughly $r2^b \approx 2^{a+b}$ operations, and the second part takes roughly 2^{2a} operations. So one usually chooses $b \approx a$. However, in our protocol, in order to be able to give a "proof" of security we'll optimize slightly differently, taking $b \approx 2a$.

Note that for fixed c, the value of a chosen with $b \approx 2a$ is different from the optimal value a' that one would choose to factor N. In the former case one sets $2^{2a} \approx u^u$ (where $u = c/a$) — that is, $2a \approx \frac{c}{a} \log u$ — and in the latter case one sets $a' \approx \frac{c}{a'} \log u'$ (where $u' = c/a'$). Since u' is of the same order of magnitude as u, by dividing these two equations we get approximately $a' \approx \sqrt{2}a$. This leads to the estimate $2^{(2\sqrt{2})a}$ for the number of operations needed to factor N.

We now describe the protocol. Alice wants to prove her identity to Bob, i.e., prove that she knows the factors of her public modulus N. Bob sends her a challenge that consists of s linearly independent vectors in $\mathbb{F}_2^r$, where $0 \leq s \leq r-1$. Alice must respond with an x such that x^2 mod N is p_r-smooth and such that its exponent-vector is not in the subspace S spanned by Bob's challenge vectors. (The idea is to prevent an imposter from giving a correct response by combining earlier responses of Alice; thus, in practice Bob would be sure to include the exponent-vectors of Alice's earlier responses among his challenge

vectors.) Alice can do this quickly, because it is easy to find square roots modulo N if one knows the factorization of N.

We now reduce factoring to impersonating Alice. Let IO be the impersonator-oracle. To factor N, we make r calls to IO (where each time our challenge vectors consist of the exponent-vectors of all the earlier responses of IO) to get a set of relations whose exponent-vectors span $\mathbb{F}_2^r$. After that we merely have to find k more randomly generated x with p_r-smooth $x^2 \bmod N$ in order to have probability $1 - 2^{-k}$ of factoring N. Finding these x's takes time about $k2^b$. Since we have to solve a matrix each time, the time is really $k(2^b + 2^{2a})$. If a call to IO on average takes time T, then the total time to factor N is $T' \approx k(2^b + 2^{2a}) + rT \approx k2^{2a+1} + 2^a T$ since $b = 2a$ and $r \approx 2^a$. We are assuming that factoring N requires $2^{(2\sqrt{2})a}$ operations, and so we obtain the nontrivial lower bound $T \geq 2^{(2\sqrt{2}-1)a}$. Whenever one is able to prove a lower bound for an adversary's running time that, although far short of what one ideally would want, is highly nontrivial and comes close to the limits of practical feasibility, such a result can be viewed as reassuring (see also Remark 2 below).

However, the protocol is insecure, because it can be broken in time roughly $2^b = 2^{2a}$.

This example is unrealistic not only because we're supposing that naive index-calculus is the best factoring algorithm, but also because it should have been obvious from the beginning that the protocol is insecure. We thus state as an open problem:

Problem. Find an example of a natural and realistic protocol that has a plausible (non-tight) reductionist proof of security, and is also insecure when used with commonly accepted parameter sizes.

Remark 1. Either success or failure in solving this problem would be of interest. If someone finds a (non-tightly) provably secure but insecure protocol, then the importance of the tightness question in security reductions will be clearer than ever. On the other hand, if no such example is found after much effort, then practitioners might feel justified in doubting the need for tightness in reductions.

Remark 2. It should be noted that something like this has already been done in the context of symmetric–key message authentication codes (MAC's). In [18] Cary and Venkatesan presented a MAC scheme for which they had a security proof (it was not actually a reductionist proof). Their scheme depended on a parameter l, and for the practical value $l = 32$ their proof showed that a collision cannot be found without at least 2^{27} MAC queries. Even though this figure falls far short of what one ideally would want — namely, 64 bits of security — it could be viewed as providing some assurance that the scheme does in fact have the desired security level. However, in [8] Blackburn and Paterson found an attack that could find a collision using $2^{48.5}$ MAC queries and a forgery using 2^{55} queries. This example shows that the exact guarantees implied by a proof have to be taken seriously, or else one might end up with a cryptosystem that is provably secure and also insecure.

4.2 Coron's Result for RSA Signatures

We first recall the basic RSA signature scheme with full-domain hash function. Suppose that a user Alice with public key (N, e) and secret exponent d wants to sign a message m. She applies a hash function $H(m)$ which takes values in the interval $0 \leq H(m) < N$, and then computes her signature $s = H(m)^d \bmod N$.

When Bob receives the message m and the signature s, he verifies the signature by computing $H(m)$ and then $s^e \bmod N$. If these values are equal, he is satisfied that Alice truly sent the message (because presumably only Alice knows the exponent d that inverts the exponentiation $s \mapsto s^e$) and that the message has not been tampered with (because any other message would presumably have a different hash value).

We now describe a classic reductionist security argument for this signature scheme [6]:

Reductionist security claim. If the problem of inverting $x \mapsto x^e \bmod N$ is intractable, then the RSA signature with full-domain hash function is secure in the random oracle model from chosen-message attack by an existential forger.

Argument. Suppose that we are given an arbitrary integer y, $0 \leq y < N$, and asked to find x such that $y = x^e \bmod N$. The claim follows if we show how we could find x (with high probability) if we had a forger that can mount chosen-message attacks.

So suppose that we have such a forger. We give it Alice's public key (N, e) and wait for its queries. In all cases but one, we respond to the hash query for a message m_i by randomly selecting $x_i \in \{0, 1, \ldots, N-1\}$ and setting the hash value h_i equal to $x_i^e \bmod N$. For just one value m_{i_0} we respond to the hash query by setting $h_{i_0} = y$ (recall that y is the integer whose inverse under the map $x \mapsto x^e \bmod N$ we are required to find). We choose i_0 at random and hope that $m = m_{i_0}$ happens to be the message whose signature will be forged by our existential forger. Any time the forger makes a signature query for a message m_i with $i \neq i_0$, we send x_i as its signature. Notice that this will satisfy the forger, since $x_i^e \equiv h_i \pmod N$. If the forger ends up outputting a valid signature s_{i_0} for m_{i_0}, that means that we have a solution $x = s_{i_0}$ to our original equation $y = x^e \bmod N$ with unknown x. If we guessed wrong and m_{i_0} was not the message that the forger ends up signing, then we won't be able to give a valid response to a signature query for m_{i_0}. The forger either will fail or will give us useless output, and we have to start over again. Suppose that q_h is a bound on the number of queries of the hash function. If we go through the procedure k times, the probability that every single time we fail to solve $y = x^e \bmod N$ for x is at most $(1 - 1/q_h)^k$. For large k, this approaches zero; so with high probability we succeed. This completes the argument.

Notice that the forgery program has to be used roughly $O(q_h)$ times (where q_h is the number of hash queries) in order to find the desired e-th root modulo

N. A result of Coron [19] shows that this can be improved to $O(q_s)$, where q_s denotes a bound on the number of signature queries.[7] (Thus, $q_h = q_s + q'_h$, where q'_h is a bound on the number of hash function queries that are not followed later by a signature query for the same message.)

Moreover, in a later paper [20] Coron essentially proves that his result cannot be improved to give a tight reduction argument; $O(q_s)$ is a lower bound on the number of calls on the forger needed to solve the RSA problem.

From the standpoint of practice (as emphasized, for example, in [5]) this non-tightness is important. What it means is the following. Suppose that you anticipate that a chosen-message attacker can get away with making up to 2^{20} signature queries. You want your system to have 80 bits of security; that is, you want a guarantee that such a forger will require time at least 2^{80}. The results of [19,20] mean that you should use a large enough RSA modulus N so that you're confident that e-th roots modulo N cannot be found in fewer than $2^{100} = 2^{20} \cdot 2^{80}$ operations. Thus, you should use a modulus N of about 1500 bits.

4.3 The Implausible Magic of One Bit

We now look at a construction of Katz and Wang [25], who show that by adding only a single random bit to a message, one can achieve a tight reduction.[8] To sign a message m Alice chooses a random bit b and evaluates the hash function H at m concatenated with b. She then computes $s = (H(m, b))^d \bmod N$; her signature is the pair (s, b). To verify the signature, Bob checks that $s^e = H(m, b) \bmod N$.

Remarkably, Katz and Wang show that the use of a single random bit b is enough to get a tight reduction from the RSA problem to the problem of producing a forgery of a Katz–Wang signature. Namely, suppose that we have a forger in the random oracle model that asks for the signatures of some messages and then produces a valid signature of some other message. Given an arbitrary integer y, the simulator must use the forger to produce x such that $y = x^e \bmod N$. Without loss of generality we may assume that when the forger asks for the hash value $H(m, b)$, it also gets $H(m, b')$ (where b' denotes the complement of b). Now when the forger makes such a query, the simulator selects a random bit c and two random integers t_1 and t_2. If $c = b$, then the simulator responds with $H(m, b) = t_1^e y$ and $H(m, b') = t_2^e$; if $c = b'$, it responds with $H(m, b) = t_2^e$ and $H(m, b') = t_1^e y$. If the forger later asks the simulator to sign the message m, the simulator responds with the corresponding value of t_2. At the end the forger outputs a signature that is either an e-th root of t_2^e or an e-th root of $t_1^e y$ for some t_1 or t_2 that the simulator knows. In the latter case, the simulator has succeeded in its task. Since this happens with probability $1/2$, the simulator is almost certain — with probability $1 - 2^{-k}$ — to find the desired e-th root

[7] In the above argument, instead of responding only to the i_0-th hash query with $h_{i_0} = y$, Coron's idea was to respond to a certain optimal number $i_0, i_1, \ldots$ with $h_{i_j} = y z_j^e$ with z_j random.

[8] We shall describe a slightly simplified version of the Katz–Wang scheme. In particular, we are assuming that Alice never signs the same message twice.

after running the forger k times. This gives us a tight reduction from the RSA problem to the forgery problem.

From the standpoint of "practice-oriented provable security" the Katz–Wang modification provides a much better guarantee than did the RSA signature without the added bit. Namely, in order to get 80 bits of security one need only choose N large enough so that finding e-th roots modulo N requires 2^{80} operations — that is, one needs roughly a 1000-bit N. Thus, the appending of a random bit to the message allows us to shave 500 bits off our modulus!

This defies common sense. How could such a "magic bit" have any significant impact on the true security of a cryptosystem, let alone such a dramatic impact? This example shows that whether or not a cryptographic protocol lends itself to a tight security reduction argument is not necessarily related to the true security of the protocol.

Does tightness matter in a reductionist security argument? Perhaps not, if, as in this case, a protocol with a non-tight reduction can be modified in a trivial way to get one that has a tight reduction. On the other hand, the example in §4.1 shows that in some circumstances a non-tight reduction might be worthless. Thus, the question of how to interpret a non-tight reductionist security argument has no easy answer.

One interpretation of Coron's lower bound on tightness is that if the RSA problem has s_1 bits of security and if we suppose that an attacker could make 2^{s_2} signature queries, then RSA signatures with full-domain hash have only $s_1 - s_2$ bits of security. However, such a conclusion seems unwarranted in light of the Katz–Wang construction. Rather, it is reasonable to view Coron's lower bound on tightness as a result that casts doubt not on the security of the basic RSA signature scheme, but rather on the usefulness of reduction arguments as a measure of security of a protocol. This point of view is similar to the alternative interpretation of Boneh–Venkatesan's result that we proposed in §2.

5 Equivalence But No Tight Reduction

Let $\mathcal{P}$ denote a presumably hard problem underlying a cryptographic protocol; that is, solving an instance of $\mathcal{P}$ will recover a user's private key. For example, the RSA version of factorization is the problem $\mathcal{P}$ whose input is a product N of two unknown k-bit primes and whose output is the factorization of N.

Let $\mathcal{P}_m$ denote the problem whose input is an m-tuple of distinct inputs for $\mathcal{P}$ of the same bitlength and whose output is the solution to $\mathcal{P}$ for any one of the inputs. In the cryptographic context, m might be the number of users. In that case, solving $\mathcal{P}_m$ means finding the private key of any one of the users, while solving $\mathcal{P}$ means finding the private key of a specified user. We call the former "existential key recovery" and the latter "universal key recovery." A desirable property of a cryptosystem is that these two problems be equivalent — in other words, that it be no easier to recover the private key of a user of the attacker's choice than to recover the private key of a user that is specified to the attacker.

To see how this issue might arise in practice, let's suppose that in a certain cryptosystem a small proportion — say, 10^{-5} — of the randomly assigned private keys are vulnerable to a certain attack. From the standpoint of an individual user, the system is secure: she is 99.999% sure that her secret is safe. However, from the standpoint of the system administrator, who is answerable to a million users, the system is insecure because an attacker is almost certain (see below) to eventually obtain the private key of one or more of the users, who will then sue the administrator. Thus, a system administrator has to be worried about existential key recovery, whereas an individual user might care only about universal key recovery.

5.1 The RSA Factorization Problem

In the case of RSA, is $\mathcal{P}_m$ equivalent to $\mathcal{P}$? (For now we are asking about algorithms that solve all instances of a problem; soon we shall consider algorithms that solve a non-negligible proportion of all instances.) It is unlikely that there is an efficient reduction from $\mathcal{P}$ to $\mathcal{P}_m$. Such a reduction would imply that the following cannot be true: for every k there are a small number $r_k < m$ of moduli N that are much harder to factor than any other $2k$-bit N. On the other hand, all of our knowledge and experience with factoring algorithms support the belief that, in fact, these two problems are in practice equivalent, and that RSA does enjoy the property that existential and universal private key recovery are equivalent.

When studying the security of a protocol, one usually wants to consider algorithms that solve only a certain non-negligible proportion of the instances.[9] In this case there is an easy reduction from $\mathcal{P}$ to $\mathcal{P}_m$: given an input to $\mathcal{P}$, randomly choose $m-1$ other inputs to form an input to $\mathcal{P}_m$. One can check that this transforms an algorithm that solves a non-negligible proportion of instances of $\mathcal{P}_m$ to one that solves a non-negligible proportion of instances of $\mathcal{P}$.

However, the proportion of instances solved can be dramatically different. An algorithm $\mathcal{A}$ that solves ϵ of the instances of $\mathcal{P}$, where ϵ is small but not negligible, gives rise to an algorithm $\mathcal{A}_m$ that solves $\nu = 1-(1-\epsilon)^m$ of the instances of $\mathcal{P}_m$ (this is the probability that at least one of the m components of the input can be solved by $\mathcal{A}$). For small ϵ and large m, $\nu \approx 1 - e^{-\epsilon m}$. For example, if $\epsilon = 10^{-5}$ and $m = 10^6$, then ν is greater than 99.99%. Thus, from a theoretical point of view there seems to be a significant distance between universal private key recovery $\mathcal{P}$ and existential private key recovery $\mathcal{P}_m$ for many systems such as RSA. In other words, we know of no reductionist argument to show that if RSA is secure from the standpoint of an individual user, then it must also be secure from the standpoint of the system administrator.

[9] In this section probabilities are always taken over the set of problem instances (of a given size), and not over sets of possible choices (coin tosses) made in the execution of an algorithm. If for a given problem instance the algorithm succeeds for a non-negligible proportion of sequences of coin tosses, then we suppose that the algorithm is iterated enough times so that it is almost certain to solve the problem instance.

But once again, all of our experience and intuition suggest that there is no real distance between the two versions of the RSA factoring problem. This is because for all of the known subexponential-time factoring algorithms, including the number field sieve, the running time is believed not to be substantially different for (a) a randomly chosen instance, (b) an instance of average difficulty, and (c) a hardest possible instance. No one knows how to prove such a claim; indeed, no one can even give a rigorous proof of the $L_{1/3}$ running time for the number field sieve. And even if the claim could be proved for the current fastest factoring algorithm, we would be very far from proving that there could never be a faster algorithm for which there was a vast difference between average-case and hardest-case running times. This is why there is no hope of proving the tight equivalence of universal and existential private key recovery for RSA.

5.2 A Non-cryptographic Example

Consider the problem $\mathcal{P}$ of finding all the prime factors of an arbitrary integer N. Let us say that N is "k-easy" if it has at most one prime divisor greater than 2^k. If k is small, then $\mathcal{P}$ in that case can be solved efficiently by first using trial division, perhaps in conjunction with the Lenstra elliptic curve factoring algorithm, to pull out the prime factors $< 2^k$, and then applying a primality test to what's left over if it's greater than 1.

It is not hard to see that the proportion ϵ of n-bit integers N that are k-easy is at least k/n. Namely, for $1 \le j < 2^k$ consider N that are of the form pj for primes p. The number of such n-bit integers is asymptotic to

$$\frac{2^{n-1}}{j \ln(2^n/j)} > \frac{2^{n-1}}{\ln 2^n} \frac{1}{j}.$$

Thus, the proportion of n-bit integers that are k-easy is greater than

$$\frac{1}{\ln 2^n} \sum_{1 \le j < 2^k} \frac{1}{j} \approx \frac{\ln 2^k}{\ln 2^n} = \frac{k}{n}.$$

As an example, let's take $n = 2000$, $k = 20$. Then $\epsilon \ge 0.01$. We saw that for $m = 1000$ more than 99.99% of all instances of $\mathcal{P}_m$ can be quickly solved. In contrast, a significant proportion of the instances of $\mathcal{P}$ are outside our reach. Obviously, it is not feasible to factor a 2000-bit RSA modulus. But there is a much larger set of 2000-bit integers that cannot be completely factored with current technology. Namely, let $S_{\ge 1}$ denote the set of integers that have at least one prime factor in the interval $[2^{300}, 2^{500}]$ and at least one prime factor greater than 2^{500}. At present a number in $S_{\ge 1}$ cannot feasibly be factored, even using a combination of the elliptic curve factorization method and the number field sieve; and a heuristic argument, which we now give, shows that at least 25% of all 2000-bit integers N lie in $S_{\ge 1}$.

To see this, let S_k denote the set of integers that have exactly k prime factors in $[2^{300}, 2^{500}]$ and at least one prime factor greater than 2^{500}. Writing a 2000-bit

$N \in S_1$ in the form $N = lm$ with l a prime in $[2^{300}, 2^{500}]$ and $m \in S_0$, we see that the number of such N is equal to

$$\sum_{l \text{ prime in } [2^{300},2^{500}]} \#\left(S_0 \bigcap \left[\frac{1}{l}2^{1999}, \frac{1}{l}2^{2000}\right]\right).$$

The probability that an integer in the latter interval satisfies the two conditions defining S_0 is at least equal to

$$\text{Prob(not divisible by any prime } p \in [2^{300}, 2^{500}]) - \text{Prob}(2^{500} - \text{smooth})$$

$$\approx \prod_{p \in [2^{300},2^{500}]} (1 - \frac{1}{p}) - u^{-u},$$

where $u = (2000 - \log_2 l)/500 \geq 3$. The product is equal to $\exp \sum \ln(1 - \frac{1}{p}) \approx \exp \sum(-1/p) \approx \exp(-\ln\ln 2^{500} + \ln\ln 2^{300}) = 0.6$, and so the probability that an integer in $[\frac{1}{l}2^{1999}, \frac{1}{l}2^{2000}]$ lies in S_0 is greater than 50%. Thus, the proportion of 2000-bit integers N that lie in $S_{\geq 1} \supset S_1$ is at least

$$\frac{1}{2} \sum_{l \text{ prime in } [2^{300},2^{500}]} \frac{1}{l} \approx \frac{1}{2}(\ln\ln 2^{500} - \ln\ln 2^{300}) = \frac{1}{2}\ln(5/3) \approx 0.25,$$

as claimed.

This problem $\mathcal{P}$ does not seem to have any cryptographic significance: it is hard to imagine a protocol whose security is based on the difficulty of completely factoring a randomly chosen integer. Rather, its interest lies in the fact that, despite its apparent resemblance to the RSA factoring problem, it spectacularly fails to have a certain property — tight equivalence of existential and universal solvability — that intuitively seems to be a characteristic of RSA factoring. This example also suggests that it is probably hopeless to try to *prove* that universal and existential private key recovery are tightly equivalent for RSA.

5.3 Use Different Elliptic Curves or the Same One?

Let us look at universal versus existential private key recovery in the case of elliptic curve cryptography (ECC). Suppose that each user chooses an elliptic curve E over a finite field $\mathbb{F}_q$, a subgroup of $E(\mathbb{F}_q)$ whose order is a k-bit prime p, a basepoint P in the subgroup, and a secret key $x \bmod p$; the public key is $Q = xP$. Let $\mathcal{P}$ denote the elliptic curve discrete logarithm problem (ECDLP), that is, the problem of recovering the secret key x from the public information. Let $\mathcal{P}_m$ denote the problem whose input is an m-tuple of ECDLP inputs with distinct orders p of the subgroups and whose output is any one of the m discrete logarithms. Once again, it seems intuitively clear that $\mathcal{P}_m$ is as hard as $\mathcal{P}$, although it is very unlikely that a tight reduction from $\mathcal{P}$ to $\mathcal{P}_m$ could be found.

In contrast, suppose that everyone uses the same elliptic curve group, and only the private/public key pairs (x, Q) differ. In that case ECC *provably* enjoys the property of tight equivalence of existential and universal private key recovery. The reason is that the ECDLP on a fixed group is "self-reducible." That means that, given an instance we want to solve, we can easily create an m-tuple of distinct random instances such that the solution to any one of them gives us the solution to the problem we wanted to solve. Namely, given an input Q, we randomly choose m distinct integers y_i modulo p and set $Q_i = y_i Q$. A $\mathcal{P}_m$-oracle will solve one of the ECDLP instances with input Q_i. Once we know its discrete log x_i, we immediately find $x = y_i^{-1} x_i \bmod p$. This shows that for the ECDLP on a fixed curve the universal private key recovery problem $\mathcal{P}$ reduces (tightly) to the existential private key recovery problem $\mathcal{P}_m$.

Thus, if we want a cryptosystem with the *provable* security property of tight equivalence of existential and universal private key recovery, then we should not only choose ECC in preference to RSA, but also insist that all users work with the same elliptic curve group.

Needless to say, we are not suggesting that this would be a good reason to choose one type of cryptography over another. On the contrary, what this example shows is that it is sometimes foolish to use the existence or absence of a tight reductionist security argument as a guide to determine which version of a cryptosystem is preferable.

Remark 3. We should also recall the problematic history of attempts to construct cryptosystems whose security is based on a problem for which the average cases and the hardest cases are *provably* equivalent.[10] This was finally done by Ajtai and Dwork [2] in 1997. However, the following year Nguyen and Stern [30] found an attack that recovers the secret key in the Ajtai–Dwork system unless parameters are chosen that are too large to be practical (see also [31]).

6 Pseudorandom Bit Generators

A pseudorandom bit generator G is a function — actually, a family of functions parameterized by n and $M \gg n$ — that takes as input a random sequence of n bits (called the "seed") and outputs a sequence of M bits that appear to be random. More precisely, G is said to be *asymptotically secure in the sense of indistinguishability* if there is no polynomial time statistical test that can distinguish (by a non-negligible margin) between its output and random output. An alternative and at first glance weaker notion of security is that of the "next bit" test: that there is no value of j for which there exists a polynomial time algorithm that, given the first $j-1$ bits, can predict the j-th bit with greater than $\frac{1}{2} + \epsilon$ chance of success (where ϵ is non-negligible as a function of n). A theorem of Yao (see [26], pp. 170-171) shows that these two notions of security

[10] Discrete-log-based systems do not have this property because the underlying problem is self-reducible only after the group has been fixed; there is clearly no way to reduce one instance to another when the groups have different orders.

are equivalent. However, that theorem is non-tight in the sense that ϵ-tolerance for the next bit test corresponds only to $(M\epsilon)$-tolerance for indistinguishability.

If one wants to analyze the security of a pseudorandom bit generator more concretely, one has to use a more precise definition than the asymptotic one. Thus, for given values of n and M, G is said to be (T, ϵ)-*secure in the sense of indistinguishability* if there is no algorithm (statistical test) with running time bounded by T such that the probability of a "yes" answer in response to the output of G and the probability of a "yes" answer in response to a truly random sequence of M bits differ in absolute value by at least ϵ. The relation between indistinguishability and the next bit test is that we have to know that our generator is $(T, \epsilon/M)$-secure in the next bit sense in order to conclude that it is (T, ϵ)-secure in the sense of indistinguishability.

6.1 The Blum–Blum–Shub Generator

Let N be an n-bit product of two large primes that are each $\equiv 3 \pmod 4$ (such an N is called a "Blum integer"), and choose a (small) integer j. The Blum–Blum–Shub (BBS) pseudorandom bit generator G takes a random x mod N and produces $M = jk$ bits as follows. Let $x_0 = x$, and for $i = 1, \ldots, k$ let[11]

$$x_i = \min\{x_{i-1}^2 \bmod N,\ N - (x_{i-1}^2 \bmod N)\}.$$

Then the output of G consists of the j least significant bits of x_i, $i = 1, \ldots, k$.

Obviously, the larger j is, the faster G generates M bits. However, the possibility of distinguishing the generated sequence from a truly random sequence becomes greater as j grows. In [41] and [3] it was proved that $j = O(\log \log N)$ bits can be securely extracted in each iteration, under the assumption that factoring is intractable.

This asymptotic result was used to justify recommended values of j. For example, in 1994 the Internet Engineering Task Force [21] made the following recommendation (in this and the following quote the modulus is denoted by n rather than N):

> Currently the generator which has the strongest public proof of strength is called the Blum Blum Shub generator... If you use no more than the $\log_2(\log_2(s_i))$ low order bits, then predicting any additional bits from a sequence generated in this manner is provable [sic] as hard as factoring n.

This recommendation has been repeated more recently, for example, in the book by Young and Yung ([43], p. 68):

> The Blum–Blum–Shub PRBG is also regarded as being secure when the $\log_2(\log_2(n))$ least significant bits...are used (instead of just the least significant bit). So, when n is a 768-bit composite, the 9 least significant bits can be used in the pseudorandom bit stream.

[11] The original generator described in [9] has $j = 1$ and $x_i = x_{i-1}^2 \bmod N$.

Let us compare this recommendation with the best security bounds that are known. In what follows we set

$$L(n) \approx 2.8 \cdot 10^{-3} \exp\left(1.9229(n \ln 2)^{1/3}(\ln(n \ln 2))^{2/3}\right),$$

which is the heuristic expected running time for the number field sieve to factor a random n-bit Blum integer (here the constant $2.8 \cdot 10^{-3}$, which is taken from [40], was obtained from the reported running time for factoring a 512-bit integer), and we assume that no algorithm can factor such an integer in expected time less than $L(n)$.

For the $j = 1$ version of Blum–Blum–Shub the best concrete security result (for large n) is due to Fischlin and Schnorr [22], who showed that the BBS generator is (T, ϵ)-secure in the sense of indistinguishability if

$$T \leq \frac{L(n)(\epsilon/M)^2}{6n \log n} - \frac{2^7 n(\epsilon/M)^{-2} \log(8n(\epsilon/M)^{-1})}{\log n}, \tag{1}$$

where log denotes $\log_2$ here and in the sequel.

For $j > 1$ the Fischlin–Schnorr inequality (1) was generalized by Sidorenko and Schoenmakers [40], who showed that the BBS generator is (T, ϵ)-secure if

$$T \leq \frac{L(n)}{36n(\log n)\delta^{-2}} - 2^{2j+9} n\delta^{-4}, \tag{2}$$

where $\delta = (2^j - 1)^{-1}(\epsilon/M)$. For large n this is an improvement over the inequality

$$T \leq \frac{L(n)(\epsilon/M)^8}{2^{4j+27} n^3}, \tag{3}$$

which is what follows from the security proof in [3].

Returning to the parameters recommended in [21] and [43], we take $n = 768$ and $j = 9$. Suppose we further take $M = 10^7$ and $\epsilon = 0.01$. According to inequality (2), the BBS generator is secure against an adversary whose time is bounded by -2^{192}. (Yes, that's a negative sign!) In this case we get a "better" result from inequality (3), which bounds the adversary's time by 2^{-264}. (Yes, that's a negative exponent!) These less-than-reassuring security guarantees are not improved much by changing M and ϵ. For example, if $M = 2^{15}$ and $\epsilon = 0.5$, we get $T \leq -2^{136}$ and $T \leq 2^{-134}$ from (2) and (3), respectively. Thus, depending on whether we use (2) or (3), the adversary's running time is bounded either by a negative number or by 10^{-40} clock cycles!

Nor does the recommendation in [21] and [43] fare well for larger values of n. In Table 1, the first column lists some values of n; the second column gives $L(n)$ to the nearest power of 2 (this is the bound on the adversary's running time that would result from a tight reduction); the third column gives the corresponding right-hand side of inequality (2); and the fourth column gives the right-hand side of (3). Here we are taking $j = \lfloor \log n \rfloor$, $M = 10^7$, and $\epsilon = 0.01$.

Table 1. The BBS generator: bounds on the adversary's running time with $j = \lfloor \log n \rfloor$

n	$L(n)$	Bound from (2)	Bound from (3)
1024	2^{78}	-2^{199}	2^{-258}
2048	2^{108}	-2^{206}	2^{-235}
3072	2^{130}	-2^{206}	2^{-215}
7680	2^{195}	-2^{213}	2^{-158}
15360	2^{261}	-2^{220}	2^{-99}

Thus, the asymptotic result in [3,41], which seemed to guarantee that we could securely extract $j = \lfloor \log n \rfloor$ bits in each iteration, does not seem to deliver in practice what it promises in theory.

Suppose that we retreat from the idea of getting $j = \lfloor \log n \rfloor$ bits from each iteration, and instead use the BBS generator to give just $j = 1$ bit per iteration. Now the security guarantees given by the inequalities (1) and (3) are better, but not by as much as one might hope. Table 2 gives the corresponding right-hand sides of (1) (in the third column) and (3) (in the fourth column) for $j = 1$, $M = 10^7$, and $\epsilon = 0.01$.

Table 2. The BBS generator: bounds on the adversary's running time with $j = 1$

n	$L(n)$	Bound from (1)	Bound from (3)
1024	2^{78}	-2^{79}	2^{-222}
2048	2^{108}	-2^{80}	2^{-194}
3072	2^{130}	-2^{80}	2^{-175}
7680	2^{195}	2^{115}	2^{-114}
15360	2^{261}	2^{181}	2^{-51}

The cross-over point at which the Fischlin–Schorr inequality starts to give a meaningful security guarantee is about $n = 5000$ (for which the right-hand side of (1) is roughly 2^{84}). Unfortunately, it is not very efficient to have to perform a 5000-bit modular squaring for each bit of the pseudorandom sequence.

Remark 4. The recommended value $j = \log(\log N)$ in [21] and [43] was obtained by taking the asymptotic result $j = O(\log(\log N))$ and setting the implied constant C in the big-O equal to 1. The choice $C = 1$ is arbitrary. In many asymptotic results in number theory the implicit constant is much greater, so with equal justification one might decide to take $C = 100$. It is amusing to note that if one did that with 1000-bit N, one would get a completely insecure BBS generator. Since $j = 100 \log(\log N) = 1000$, one would be using all the bits of x_i. From the output an attacker could easily determine N (by setting $N_1 = x_2 \pm x_1^2$, $N_i = \gcd(N_{i-1}, x_{i+1} \pm x_i^2)$, so that $N_i = N$ for $i \geq i_0$ for quite small i_0), after which the sequence would be deterministic for the attacker.

6.2 The Gennaro Generator

Let p be an n-bit prime of the form $2q + 1$ with q prime, and let c be an integer such that $c \gg \log n$. Let g be a generating element of $\mathbb{F}_p^*$. The Gennaro pseudorandom bit generator G takes a random x mod $p-1$ and produces $M = (n-c-1)k$ bits as follows (see [23]). Let $x \mapsto \tilde{x}$ be the function on n-bit integers $x = \sum_{l=0}^{n-1} s_l 2^l$ given by $\tilde{x} = s_0 + \sum_{l=n-c}^{n-1} s_l 2^l$. Let $x_0 = x$, and for $i = 1, \ldots, k$ let $x_i = g^{\tilde{x}_{i-1}} \bmod p$. Then the output of G consists of the 2nd through $(n-c)$-th bits of x_i, $i = 1, \ldots, k$ (these are the bits that are ignored in $\tilde{x}_i$).

In comparison with the BBS generator, each iteration of the exponentiation $x_i = g^{\tilde{x}_{i-1}} \bmod p$ takes longer than modular squaring. However, one gets many more bits each time. For example, with the parameters $n = 1024$ and $c = 160$ that are recommended in [24] each iteration gives 863 bits.

In [24], Howgrave-Graham, Dyer, and Gennaro compare the Gennaro generator (with $n = 1024$ and $c = 160$) with a SHA-1 based pseudorandom bit generator (namely, the ANSI X9.17 generator) that lacks a proof of security:

> ...SHA-1 based pseudorandom number generation is still considerably faster than the one based on discrete logarithms. However, the difference, a factor of less than 4 on this hardware, may be considered not too high a price to pay by some who wish to have a "provably secure," rather than a "seemingly secure" (i.e., one that has withstood cryptographic attack thus far) system for pseudorandom number generation.

The proof of security for the Gennaro generator is given in §4 of [23]. Interestingly, Gennaro uses the next bit test rather than the indistinguishability criterion to derive his results. However, it is the latter criterion rather than the next bit test that is the widely accepted notion of security of a pseudorandom bit generator. As mentioned above, to pass from the next bit test to indistinguishability, one must replace ϵ by ϵ/M in the inequalities. One finds [39] that Gennaro's proof then gives the following inequality for the adversary's time:

$$T \leq \frac{L(n)(n-c)^3}{16c(\ln c)(M/\epsilon)^3}. \tag{4}$$

For $n = 1024$, $c = 160$, $M = 10^7$, and $\epsilon = 0.01$, the right-hand side of (4) is 18. Thus, the security guarantees that come with the Gennaro generator are not a whole lot more reassuring than the ones in §6.1.

We conclude this section by repeating the comment we made in §5.5 of [27]:

> Unfortunately, this type of analysis [incorporating the measure of non-tightness into recommendations for parameter sizes] is generally missing from papers that argue for a new protocol on the basis of a "proof" of its security. Typically, authors of such papers trumpet the advantage that their protocol has over competing ones that lack a proof of security (or that have a proof of security only in the random oracle model), then give a non-tight reductionist argument, and at the end give key-length recommendations that would make sense if their proof had been

tight. They fail to inform the potential users of their protocol of the true security level that is guaranteed by the "proof" if, say, a 1024-bit prime is used. It seems to us that cryptographers should be consistent. If one really believes that reductionist security arguments are very important, then one should give recommendations for parameter sizes based on an honest analysis of the security argument, even if it means admitting that efficiency must be sacrificed.

7 Short Signatures

In the early days of provable security work, researchers were content to give asymptotic results with polynomial-time reductions. In recent years, they have increasingly recognized the importance of detailed analyses of their reductions that allow them to state their results in terms of specified bounds, probabilities, and running times.

But regrettably, they often fail to follow through with interpretations in practical terms of the formulas and bounds in their lemmas and theorems. As a result, even the best researchers sometimes publish results that, when analyzed in a concrete manner, turn out to be meaningless in practice. In this section we give an example of this.

First we recall that when analyzing the security of a signature scheme against chosen-message attack in the random oracle model, one always has two different types of oracle queries — signature queries and hash function queries — each with a corresponding bound on the number of queries that an attacker can make.[12] In practice, since signature queries require a response from the target of the attack, to some extent they can be limited. So it is reasonable to suppose that the bound q_s is of the order of a million or a billion. In contrast, a query to the hash oracle corresponds in practice to simply evaluating a publicly available function. There is no justification for supposing that an attacker's hash queries will be limited in number by anything other than her total running time. Thus, to be safe one should think of q_h as being 2^{80}, or at the very least 2^{50}.

We now give an overview of three signature schemes proposed by Boneh-Lynn-Shacham [12] and Boneh-Boyen [11]. All three use bilinear pairings to obtain short signatures whose security against chosen-message attack is supported by reductionist arguments. Let k denote the security parameter; in practice, usually $k \approx 80$. For efficient implementation it is generally assumed that the group order q is approximately 2^{2k}, which is large enough to prevent squareroot attacks on the discrete log problem.

In the Boneh-Lynn-Shacham (BLS) signature scheme the signatures then have length only about $2k$. In [12] this scheme is shown to be secure against chosen-message attack in the random oracle model if the Computational Diffie-Hellman problem is hard.

[12] We shall continue to use the notation q_s and q_h for these bounds, even though we are also using q to denote the prime group order. We apologize to the reader for our over-use of the letter q.

In [11] Boneh and Boyen propose two alternatives to the BLS scheme. The first one (referred to below as the "BB signature scheme") has roughly twice the signature length of BLS, namely, $4k$, but it can be proven secure against chosen-message attack without using the random oracle model, assuming that the so-called Strong Diffie-Hellman problem (SDH) is intractable. The second signature scheme proposed in the paper (the "BB hash-signature scheme") is a variant of the first one in which the message must be hashed. Its proof of security uses the random oracle assumption. Like the BLS scheme, the BB hash-signature scheme has signature length roughly $2k$ rather than $4k$; moreover, it has the advantage over BLS that verification is roughly twice as fast.

The proofs in [11] are clear and readable, in part because the authors introduce a simplified version of the BB scheme (the "basic" BB scheme) in order to formulate an auxiliary lemma (Lemma 1) that is used to prove the security of both the full BB scheme (without random oracles) and the BB hash-signature scheme (with random oracles). What concerns us is the second of these results (Theorem 2).

We now describe our reason for doubting the value of that result. We shall give Lemma 1 and Theorem 2 of [11] in a slightly simplified form where we omit mention of the probabilities ϵ and ϵ', which are not relevant to our discussion. The underlying hard problem SDH for both BB schemes is parameterized by an integer that we shall denote q'_s.

Lemma 1. Suppose that q'_s-SDH cannot be solved in time less than t'. Then the basic signature scheme is secure against a weak chosen-message attack by an existential forger whose signature queries are bounded by q''_s and whose running time is bounded by t'', provided that

$$q''_s < q'_s \qquad \text{and} \qquad t'' \leq t' - \Theta({q'_s}^2 T),$$

where T is the maximum time for a group exponentiation.

Theorem 2. Suppose that the basic signature scheme referred to in Lemma 1 is existentially unforgeable under a weak chosen-message attack with bounds q''_s and t''. Then the corresponding hash-signature scheme is secure in the random oracle model against an adaptive chosen-message attack by an existential forger whose signature queries are bounded by q_s, whose hash queries are bounded by q_h, and whose running time is bounded by t, provided that

$$q_s + q_h < q''_s \qquad \text{and} \qquad t \leq t'' - o(t'').$$

Casual readers are likely to view this theorem as a fairly precise and definitive security guarantee, especially since the authors comment: "Note that the security reduction in Theorem 2 is tight... Proofs of signature schemes in the random oracle model are often far less tight." Readers are not likely to go to the trouble of comparing the statement of the theorem with that of Lemma 1, particularly since in [11] several pages of text separate the lemma from the theorem. But such a comparison must be made if we want to avoid ending up in the embarrassing

situation of the previous section (see Tables 1 and 2), where the adversary's running time was bounded by a negative number.

If we put the two statements side by side and compare them, we see that in order for the bound on the adversary's running time to be a positive number it is necessary that

$$q_h^2 < t' \approx 2^k,$$

where k is the security parameter. In practice, this means that we need $q_h \ll 2^{40}$.[13] Thus, there is no security guarantee at all for the hash-signature scheme in Theorem 2 unless one assumes that the adversary is severely limited in the number of hash values she can obtain.

The conclusion of all this is not, of course, that the signature scheme in Theorem 2 of [11] is necessarily insecure, but rather that the provable security result for it has no meaning if parameters are chosen for efficient implementation.

8 The Paillier–Vergnaud Results for Schnorr Signatures

In [32] Paillier and Vergnaud prove that it is unlikely that a reduction — more precisely, an "algebraic" reduction — can be found from the Discrete Logarithm Problem (DLP) to forging Schnorr signatures. After describing this result and its proof, we compare it with various positive results that suggest equivalence between forgery of Schnorr-type signatures and the DLP.

8.1 Schnorr Signatures

We first recall the Schnorr signature scheme [35].

Schnorr key generation. Let q be a large prime, and let p be a prime such that $p \equiv 1 \pmod{q}$. Let g be a generator of the cyclic subgroup G of order q in $\mathbb{F}_p^*$. Let H be a hash function that takes values in the interval $[1, q-1]$. Each user Alice constructs her keys by selecting a random integer x in the interval $[1, q-1]$ and computing $y = g^x \bmod p$. Alice's public key is y; her private key is x.

Schnorr signature generation. To sign a message m, Alice must do the following:

1. Select a random integer k in the interval $[1, q-1]$.
2. Compute $r = g^k \bmod p$, and set $h = H(m, r)$.
3. Set $s = k + hx \bmod q$.

The signature for the message is the pair of integers (h, s).

Schnorr signature verification. To verify Alice's signature (h, s) on a message m, Bob must do the following:

1. Obtain an authenticated copy of Alice's public key y.
2. Verify that h and s are integers in the interval $[0, q-1]$.

[13] If we had a 160-bit group order and took $q_h = 2^{50}$, then Theorem 2 and Lemma 1 would give us the bound $t \leq -2^{100}$ for the adversary's running time.

3. Compute $u = g^s y^{-h} \bmod p$ and $v = H(m, u)$.
4. Accept the signature if and only if $v = h$.

8.2 Paillier–Vergnaud

Before giving the Paillier–Vergnaud result, we need some preliminaries. First, suppose that we have a group G generated by g. By the "discrete log" of $y \in G$ we mean a solution x to the equation $g^x = y$. In [32] the "one-more DLP" problem, denoted n-DLP, is defined as follows.

n-DLP. Given $r_0, r_1, \ldots, r_n \in G$ and a discrete log oracle $DL(\cdot)$ that can be called upon n times, find the discrete logs of all $n+1$ elements r_i.

Second, by an "algebraic" reduction $\mathcal{R}$ from the DLP to forgery, Paillier and Vergnaud mean a reduction that is able to perform group operations but is not able to use special features of the way that group elements are represented. In addition, they suppose that the choices made while carrying out $\mathcal{R}$ are accessible to whomever is running the reduction algorithm (in the proof below this is the n-DLP solver). With these definitions, they prove the following result.

Theorem. Suppose that G is a group of order q generated by g. Suppose that $\mathcal{R}$ is an algebraic reduction from the DLP to universal forgery with a key-only attack that makes n calls to the forger. Then n-DLP is easy.

Proof. Let $r_0, r_1, \ldots, r_n \in G$ be an instance of n-DLP. We are required to find all $n+1$ discrete logs, and we can call upon the oracle $DL(\cdot)$ n times. The reduction $\mathcal{R}$ will find the discrete logarithm of any element if it is given a forger that will break n different instances (chosen by $\mathcal{R}$) of the Schnorr signature scheme. We ask $\mathcal{R}$ to find the discrete log of r_0. Then n times the reduction algorithm produces a Schnorr public key y_i and a message m_i. Each time we simulate the forger by choosing $r = r_i$, computing the hash value $h_i = H(m_i, r_i)$, and then setting s_i equal to the discrete log of $r_i y_i^{h_i}$, which we determine from the oracle:

$$s_i = DL(r_i y_i^{h_i}).$$

We send (h_i, s_i), which is a valid signature for m_i with public key y_i, to $\mathcal{R}$. Finally, $\mathcal{R}$ outputs the discrete log x_0 of r_0.

In order to compute the public key y_i, $\mathcal{R}$ must have performed group operations starting with the only two group elements that it was given, namely, g and r_0. Thus, for some integer values α_i and β_i that are accessible to us, we have $y_i = g^{\alpha_i} r_0^{\beta_i}$. Once we learn x_0 (which is the output of $\mathcal{R}$), we can compute

$$x_i = s_i - h_i(\alpha_i + x_0 \beta_i) \bmod q,$$

which is the discrete logarithm of r_i, $i = 1, \ldots, n$. We now know the discrete logs of all the $n+1$ values $r_0, \ldots, r_n$. This completes the proof.

Paillier and Vergnaud proved similar results for other signature schemes based on the DLP, such as DSA and ECDSA. In the latter cases they had to modify the n-DLP slightly: the discrete log oracle is able to give the queried discrete logs to different bases g_i.

Intuitively, the "one-more DLP" problem seems to be equivalent to the DLP, even though there is an obvious reduction in just one direction. Thus, the Paillier–Vergnaud results can be paraphrased as follows: *A reduction from the DLP to forgery is unlikely unless the DLP is easy.* In this sense the above theorem has the same flavor as the result of Boneh and Venkatesan [13] discussed in §2. As in that case, one possible interpretation of Paillier–Vergnaud is that there might be a security weakness in Schnorr-type signatures. Indeed, that interpretation is suggested by the title "Discrete-log-based signatures may not be equivalent to discrete log" and by the claim in the Introduction that "our work disproves that Schnorr, ElGamal, DSA, GQ, etc. are maximally secure."[14]

On the other hand, as in §2, an alternative explanation is that their work gives a further illustration of the limitations of reduction arguments. It is instructive to compare the negative result of Paillier–Vergnaud concerning the existence of reductions with the following two positive reductionist security results for Schnorr-type signature schemes.

8.3 Random Oracle Reductions

Reductionist security claim. In the Schnorr signature scheme, if the hash function is modeled by a random oracle, then the DLP reduces to universal forgery.

Argument. Suppose that the adversary can forge a signature for m. After it gets $h = H(m, r)$, suppose that it is suddenly given a second hash function H'. Since a hash function has no special properties that the forger can take advantage of, whatever method it used will work equally well with H replaced by H'. In other words, we are using the random oracle model for the hash function. So the forger uses $h' = H'(m, r)$ as well as $h = H(m, r)$ and produces two valid signatures (h, s) and (h', s') for m, with the same r but with $h' \neq h$. Note that the value of k is the same in both cases, since r is the same. By subtracting the two values $s \equiv k + xh$ and $s' \equiv k + xh' \pmod{q}$ and then dividing by $h' - h$, one can use the forger's output to immediately find the discrete log x.[15]

The above argument is imprecise. Strictly speaking, we should allow for the possibility that a forger gets $H(m, r)$ for several different values of r and signs only one of them. In that case we guess which value will be signed, and run the forger program several times with random guesses until our guess is correct. We described a rigorous argument (for a stronger version of the above claim) in §5 of [27], and full details can be found in [33,34].

Note that the need to run the forger many times leads to a non-tight reduction. In [34] it is shown that it suffices to call on the forger approximately q_h times, where q_h is a bound on the number of hash function queries. In [32] Paillier and Vergnaud prove that, roughly speaking, an algebraic reduction in the random oracle model cannot be tighter than $\sqrt{q_h}$. Much as Coron did in the case

[14] Paillier and Vergnaud do acknowledge, however, that their work leads to "no actual attack or weakness of either of these signature schemes."

[15] Note that one does not need to know k.

of RSA signatures, Paillier and Vergnaud establish a lower bound on tightness of the reduction.

What do we make of the circumstance that, apparently, no tight reduction from the DLP to forgery is possible in the random oracle model, and no reduction at all is likely in a standard model? As usual, several interpretations are possible. Perhaps this shows that reductions in the random oracle model are dangerous, because they lead to security results that cannot be achieved in a standard model. On the other hand, perhaps we can conclude that the random oracle model should be used, because it can often come closer to achieving what our intuition suggests should be possible. And what about the non-tightness? Should we ignore it, or should we adjust our recommendations for key sizes so that we have, say, 80 bits of security after taking into account the non-tightness factor?

8.4 Brown's Result for ECDSA

Finally, we discuss another positive result that concerns ECDSA. We shall state without proof an informal version of a theorem of D. Brown [14,15].

Theorem. Suppose that the elliptic curve is modeled by a generic group. Then the problem of finding a collision for the hash function reduces to forgery of ECDSA signatures.

Brown's theorem falls outside the framework of the results in [32]. It is a reduction not from the DLP to forgery, but rather from collision finding to forgery. And it is a tight reduction. By making the generic group assumption, one is essentially assuming that the DLP is hard (see [36]). If the hash function is collision-resistant, then the assumed hardness of the DLP (more precisely, the generic group assumption) implies hardness of forgery. However, in [14] there is no reduction from the DLP to forgery.

Both Brown and Paillier–Vergnaud make similar assumptions about the group. The latter authors implicitly assume that n-DLP is hard, and they assume that a reduction uses the group in a "generic way," that is, computes group operations without exploiting any special features of the encodings of group elements. Similarly, Brown assumes that the elliptic curve group is for all practical purposes like a generic group, and, in particular, the DLP is hard.

But their conclusions are opposite one another. Paillier and Vergnaud prove that no reduction is possible in the standard model, and no tight reduction is possible in the random oracle model. Brown gives a tight reduction — of a different sort than the ones considered in [32] — which proves security of ECDSA subject to his assumptions.

So *is* forgery of Schnorr-type signatures equivalent to the DLP? The best answer we can give is to quote a famous statement by a recent American president: it all depends on what the definition of "is" is.[16]

[16] The context was an explanation of his earlier statement that "there *is* no sexual relationship with Ms. Lewinsky." A statement to the effect that "there *is* no relationship of equivalence between the DLP and forgery of discrete-log-based signatures" is, in our judgment, equally implausible.

9 Conclusions

In his 1998 survey article "Why chosen ciphertext security matters" [37], Shoup explained the rationale for attaching great importance to reductionist security arguments:

> This is the preferred approach of modern, mathematical cryptography. Here, one shows with mathematical rigor that any attacker that can break the cryptosystem can be transformed into an efficient program to solve the underlying well-studied problem (e.g., factoring large numbers) that is widely believed to be very hard. Turning this logic around: if the "hardness assumption" is correct as presumed, the cryptosystem is secure. This approach is about the best we can do. If we can prove security in this way, then we essentially rule out all possible shortcuts, even ones *we have not yet even imagined*. The only way to attack the cryptosystem is a full-frontal attack on the underlying hard problem. Period. (p. 15; emphasis in original)

Later in [37] Shoup concluded: "Practical cryptosystems that are provably secure are available, and there is very little excuse for not using them." One of the two systems whose use he advocated because they had proofs of security was RSA-OAEP [7].

Unfortunately, history has not been kind to the bold opinion quoted above about the reliability of provable security results. In 2001, Shoup himself [38] found a flaw in the purported proof of security of general OAEP by Bellare and Rogaway. The same year, Manger [29] mounted a successful chosen-ciphertext attack on RSA-OAEP. Interestingly, it was not the flaw in the Bellare–Rogaway proof (which was later patched for RSA-OAEP) that made Manger's attack possible. Rather, Manger found a shortcut that was "not yet even imagined" in 1998, when Shoup wrote his survey.

It is often difficult to determine what meaning, if any, a reductionist security argument has for practical cryptography. In recent years, researchers have become more aware of the importance of concrete analysis of their reductions. But while they often take great pains to prove precise inequalities, they rarely make any effort to explain what their mathematically precise security results actually mean in practice.

For example, in [1] the authors construct a certain type of password-based key exchange system and give proofs of security in the random oracle model based on hardness of the computational Diffie–Hellman (CDH) problem. Here is the (slightly edited) text of their basic result (Corollary 1 of Theorem 1, pp. 201-202 of [1]) that establishes the relation between the "advantage" of an adversary in breaking their SPAKE1 protocol and the advantage of an adversary in solving the CDH:

> *Corollary 1.* Let G be a represent group of order p, and let $\mathcal{D}$ be a uniformly distributed dictionary of size $|\mathcal{D}|$. Let SPAKE1 be the above password-based encrypted key exchange protocol associated with these primitives. Then for any numbers t, q_{start}, q^A_{send}, q^B_{send}, q_H, q_{exe},

$$\mathrm{Adv}^{\mathrm{ake}}_{\mathrm{SPAKE},\mathcal{D}}(t, q_{\mathrm{start}}, q^A_{\mathrm{send}}, q^B_{\mathrm{send}}, q_H, q_{\mathrm{exe}})$$

$$\leq 2 \cdot \left(\frac{q^A_{\mathrm{send}} + q^B_{\mathrm{send}}}{|\mathcal{D}|} + \sqrt[6]{\frac{2^{14}}{|\mathcal{D}|^2} \mathrm{Adv}^{\mathrm{CDH}}_G(t') + \frac{2^{15} q_H^4}{|\mathcal{D}|^2 p}} \right)$$

$$+2 \cdot \left(\frac{(q_{\mathrm{exe}} + q_{\mathrm{send}})^2}{2p} + q_H \mathrm{Adv}^{\mathrm{CDH}}_G(t + 2q_{\mathrm{exe}}\tau + 3\tau) \right),$$

where q_H represents the number of queries to the H oracle; q_{exe} represents the number of queries to the Execute oracle; q_{start} and q^A_{send} represent the number of queries to the Send oracle with respect to the initiator A; q^B_{send} represents the number of queries to the Send oracle with respect to the responder B; $q_{\mathrm{send}} = q^A_{\mathrm{send}} + q^B_{\mathrm{send}} + q_{\mathrm{start}}$; $t' = 4t + O((q_{\mathrm{start}} + q_H)\tau)$; and τ is the time to compute one exponentiation in G.

The paper [1] includes a proof of this bewildering and rather intimidating inequality. But the paper gives no indication of what meaning, if any, it would have in practice. The reader who might want to use the protocol and would like to find parameters that satisfy security guarantees and at the same time allow a reasonably efficient implementation is left to fend for herself.

In the provable security literature the hapless reader is increasingly likely to encounter complicated inequalities involving more than half a dozen variables. (For other examples, see Theorem 5 in [28] and Theorems 2 and 3 in [4].) The practical significance of these inequalities is almost never explained. Indeed, one has to wonder what the purpose is of publishing them in such an elaborate, undigested form, with no interpretation given. Whatever the authors' intent might have been, there can be little doubt that the effect is not to enlighten their readers, but only to mesmerize them.

* * *

Embarking on a study of the field of "provable security," before long one begins to feel that one has entered a realm that could only have been imagined by Lewis Carroll, and that the Alice of cryptographic fame has merged with the heroine of Carroll's books:

> Alice felt dreadfully puzzled. The Hatter's remark seemed to her to have no sort of meaning in it, and yet it was certainly English. (*Alice's Adventures in Wonderland* and *Through the Looking-Glass*, London: Oxford Univ. Press, 1971, p. 62.)

The Dormouse proclaims that his random bit generator is provably secure against an adversary whose computational power is bounded by a negative number. The Mad Hatter responds that he has a generator that is provably secure against an adversary whose computational resources are bounded by 10^{-40} clock cycles. The White Knight is heralded for blazing new trails, but upon further examination one notices that he's riding backwards. The Program Committee is made up of Red Queens screaming "Off with their heads!" whenever authors submit a paper with no provable security theorem.

Lewis Carroll's Alice wakes up at the end of the book and realizes that it has all been just a dream. For the cryptographic Alice, however, the return to the real world might not be so easy.

Acknowledgments

We would like to thank Andrey Sidorenko for his valuable comments on pseudorandom bit generators and Bart Preneel for answering our queries about the provable security of MAC algorithms. We also wish to thank Ian Blake and Dan Brown for reading and commenting on earlier drafts of the paper. Needless to say, all the opinions expressed in this article are the sole responsibility of the authors.

References

1. M. Abdalla and D. Pointcheval, Simple password-based encrypted key exchange protocols, *Topics in Cryptology – CT-RSA 2005*, LNCS 3376, Springer-Verlag, 2005, pp. 191-208.
2. M. Ajtai and C. Dwork, A public-key cryptosystem with worst-case/average-case equivalence, *Proc. 29th Symp. Theory of Computing*, A.C.M., 1997, pp. 284-293.
3. W. Alexi, B. Chor, O. Goldreich, and C. P. Schnorr, RSA and Rabin functions: Certain parts are as hard as the whole, *SIAM J. Computing*, **17** (1988), pp. 194-209.
4. P. Barreto, B. Libert, N. McCullagh, and J.-J. Quisquater, Efficient and provably-secure identity-based signatures and signcryption from bilinear maps, *Advances in Cryptology – Asiacrypt 2005*, LNCS 3788, Springer-Verlag, 2005, pp. 515-532.
5. M. Bellare, Practice-oriented provable-security, *Proc. First International Workshop on Information Security (ISW '97)*, LNCS 1396, Springer-Verlag, 1998, pp. 221-231.
6. M. Bellare and P. Rogaway, Random oracles are practical: a paradigm for designing efficient protocols, *Proc. First Annual Conf. Computer and Communications Security*, ACM, 1993, pp. 62-73.
7. M. Bellare and P. Rogaway, Optimal asymmetric encryption — how to encrypt with RSA, *Advances in Cryptology – Eurocrypt '94*, LNCS 950, Springer-Verlag, 1994, pp. 92-111.
8. S. Blackburn and K. Paterson, Cryptanalysis of a message authentication code due to Cary and Venkatesan, *Fast Software Encryption 2004*, LNCS 3017, Springer-Verlag, 2004, pp. 446-453.
9. L. Blum, M. Blum, and M. Shub, A simple unpredictable pseudo-random number generator, *SIAM J. Computing*, **15** (1986), pp. 364-383.
10. M. Blum and S. Micali, How to generate cryptographically strong sequences of pseudo-random bits, *SIAM J. Computing*, **13** (1984), pp. 850-864.
11. D. Boneh and X. Boyen, Short signatures without random oracles, *Advances in Cryptology – Eurocrypt 2004*, LNCS 3027, Springer-Verlag, 2004, pp. 56-73.
12. D. Boneh, B. Lynn, and H. Shacham, Short signatures from the Weil pairing, *Advances in Cryptology – Asiacrypt 2001*, LNCS 2248, Springer-Verlag, 2001, pp. 514-532.

13. D. Boneh and R. Venkatesan, Breaking RSA may not be equivalent to factoring, *Advances in Cryptology – Eurocrypt '98*, LNCS 1233, Springer-Verlag, 1998, pp. 59-71.
14. D. Brown, Generic groups, collision resistance, and ECDSA, *Designs, Codes and Cryptography*, **35** (2005), pp. 119-152.
15. D. Brown, On the provable security of ECDSA, in I. Blake, G. Seroussi, and N. Smart, eds., *Advances in Elliptic Curve Cryptography*, Cambridge University Press, 2005, pp. 21-40.
16. D. Brown, Breaking RSA may be as difficult as factoring, http://eprint.iacr.org/2005/380
17. D. Brown, unpublished communication, February 2006.
18. M. Cary and R. Venkatesan, A message authentication code based on unimodular matrix groups, *Advances in Cryptology – Crypto 2003*, LNCS 2729, Springer-Verlag, 2003, pp. 500-512.
19. J.-S. Coron, On the exact security of full domain hash, *Advances in Cryptology – Crypto 2000*, LNCS 1880, Springer-Verlag, 2000, pp. 229-235.
20. J.-S. Coron, Optimal security proofs for PSS and other signature schemes, *Advances in Cryptology – Eurocrypt 2002*, LNCS 2332, Springer-Verlag, 2002, pp. 272-287.
21. D. Eastlake, S. Crocker, and J. Schiller, RFC 1750 – Randomness Recommendations for Security, available from http://www.ietf.org/rfc/rfc1750.txt
22. R. Fischlin and C. P. Schnorr, Stronger security proofs for RSA and Rabin bits, *J. Cryptology*, **13** (2000), pp. 221-244.
23. R. Gennaro, An improved pseudo-random generator based on the discrete log problem, *J. Cryptology*, **18** (2005), pp. 91-110.
24. N. Howgrave-Graham, J. Dyer, and R. Gennaro, Pseudo-random number generation on the IBM 4758 Secure Crypto Coprocessor, *Workshop on Cryptographic Hardware and Embedded Systems* (CHES 2001), LNCS 2162, Springer-Verlag, 2001, pp. 93-102.
25. J. Katz and N. Wang, Efficiency improvements for signature schemes with tight security reductions, *10th ACM Conf. Computer and Communications Security*, 2003, pp. 155-164.
26. D. Knuth, *Seminumerical Algorithms*, vol. 2 of *Art of Computer Programming*, 3rd ed., Addison-Wesley, 1997.
27. N. Koblitz and A. Menezes, Another look at "provable security," to appear in *J. Cryptology*; available from http://eprint.iacr.org/2004/152.
28. P. Mackenzie and S. Patel, Hard bits of the discrete log with applications to password authentication, *Topics in Cryptology – CT-RSA 2005*, LNCS 3376, Springer-Verlag, 2005, pp. 209-226.
29. J. Manger, A chosen ciphertext attack on RSA Optimal Asymmetric Encryption Padding (OAEP) as standardized in PKCS #1 v2.0, *Advances in Cryptology – Crypto 2001*, LNCS 2139, Springer-Verlag, 2001, pp. 230-238.
30. P. Q. Nguyen and J. Stern, Cryptanalysis of the Ajtai–Dwork cryptosystem, *Advances in Cryptology – Crypto '98*, LNCS 1462, Springer-Verlag, 1998, pp. 223-242.
31. P. Q. Nguyen and J. Stern, The two faces of lattices in cryptology, *Cryptography and Lattices – Proc. CALC 2001*, LNCS 2146, Springer-Verlag, 2001, pp. 146-180.
32. P. Paillier and D. Vergnaud, Discrete-log-based signatures may not be equivalent to discrete log, *Advances in Cryptology – Asiacrypt 2005*, LNCS 3788, Springer-Verlag, 2005, pp. 1-20.
33. D. Pointcheval and J. Stern, Security proofs for signature schemes, *Advances in Cryptology – Eurocrypt '96*, LNCS 1070, Springer-Verlag, 1996, pp. 387-398.

34. D. Pointcheval and J. Stern, Security arguments for digital signatures and blind signatures, *J. Cryptology*, **13** (2000), pp. 361-396.
35. C. P. Schnorr, Efficient signature generation for smart cards, *J. Cryptology*, **4** (1991), pp. 161-174.
36. V. Shoup, Lower bounds for discrete logarithms and related problems, *Advances in Cryptology – Eurocrypt '97*, LNCS 1233, Springer-Verlag, 1997, pp. 256-266.
37. V. Shoup, Why chosen ciphertext security matters, IBM Research Report RZ 3076 (#93122) 23/11/1998.
38. V. Shoup, OAEP reconsidered, *Advances in Cryptology – Crypto 2001*, LNCS 2139, Springer-Verlag, 2001, pp. 239-259.
39. A. Sidorenko, unpublished communication, March 2006.
40. A. Sidorenko and B. Schoenmakers, Concrete security of the Blum–Blum–Shub pseudorandom generator, *Cryptography and Coding 2005*, LNCS 3796, Springer-Verlag, 2005, pp. 355-375.
41. U. V. Vazirani and V. V. Vazirani, Efficient and secure pseudo-random number generation, *Proc. IEEE 25th Annual Symp. Foundations of Computer Science*, 1984, pp. 458-463.
42. A. Yao, Theory and applications of trapdoor functions, *Proc. IEEE 23rd Annual Symp. Foundations of Computer Science*, 1982, pp. 80-91.
43. A. Young and M. Yung, *Malicious Cryptography: Exposing Cryptovirology*, Wiley, 2004.