J. Cryptology (2007) 20: 3–37
DOI: 10.1007/s00145-005-0432-z

Journal of
CRYPTOLOGY

Another Look at "Provable Security"

Neal Koblitz
Department of Mathematics, Box 354350,
University of Washington,
Seattle, WA 98195, U.S.A.
koblitz@math.washington.edu

Alfred J. Menezes
Department of Combinatorics & Optimization,
University of Waterloo,
Waterloo, Ontario, Canada N2L 3G1
ajmeneze@uwaterloo.ca

Communicated by Johannes Buchmann

Received 16 August 2004 and revised 8 May 2005
Online publication 7 January 2006

Abstract. We give an informal analysis and critique of several typical "provable security" results. In some cases there are intuitive but convincing arguments for rejecting the conclusions suggested by the formal terminology and "proofs," whereas in other cases the formalism seems to be consistent with common sense. We discuss the reasons why the search for mathematically convincing theoretical evidence to support the security of public-key systems has been an important theme of researchers. However, we argue that the theorem-proof paradigm of theoretical mathematics is often of limited relevance here and frequently leads to papers that are confusing and misleading. Because our paper is aimed at the general mathematical public, it is self-contained and as jargon-free as possible.

Keywords. Cryptography, Public key, Provable security.

1. Introduction

Suppose that someone is using public-key cryptography to protect credit card numbers during online purchases, maintain confidentiality of medical records, or safeguard national security information. How can she be sure that the system is secure? What type of evidence could convince her that a malicious adversary could not somehow break into the system and learn her secret?

At first glance it seems that this question has a straightforward answer. At the heart of any public-key cryptosystem is a "one-way function"—a function $y = f(x)$ that is easy to evaluate but for which it is computationally infeasible to find the inverse $x = f^{-1}(y)$. For example, the system might be based on the function $y = x^e \pmod{n}$, where n

is an integer whose prime factors are secret and e is a constant exponent (this is the so-called "RSA function"); or it might be based on the function $y = g^x$, where g is a fixed generator of a group of prime order in which the so-called "discrete logarithm" (the inverse of this function) is believed to be hard to find. Then in order to have confidence in the security of the system we have to be convinced that no one knows any algorithm that could invert the one-way function in a reasonable amount of time.

Indeed, a large proportion of all of the mathematical research in public-key cryptography is concerned with algorithms for inverting the most important one-way functions. Hundreds of papers in mathematics as well as cryptography journals have been devoted to index calculus methods for factoring integers and for finding the discrete logarithm in the multiplicative group of a finite field, to improved Pollard-ρ algorithms and Weil descent methods for finding discrete logarithms on elliptic curves, and to searches for "weak parameters," i.e., RSA moduli n that are a little easier to factor than most, finite fields over which the elliptic curve discrete logarithm problem is slightly easier to solve, and so on.

Many mathematicians working in cryptography regard the question of security of a type of public-key system as equivalent to non-invertibility of the underlying one-way function. This is in fact the impression conveyed in some of the mathematically oriented introductions to cryptography. The first books on cryptography that the two of us wrote in our naive youth[1] [39], [45] suffer from this defect: the sections on security deal only with the problem of inverting the one-way function.

The problem with this limited view of security is that it fails to anticipate most of the attacks on a cryptographic system that are likely to occur. The underlying one-way function is a basic ingredient in the system (in crypto jargon one calls it a "primitive" or sometimes an "atomic primitive"), but it has to be incorporated into a particular set of instructions (called a "protocol") in order to accomplish a cryptographic objective. Throughout the history of public-key cryptography almost all of the effective attacks on the most popular systems have succeeded not by inverting the one-way function, but rather by finding a weakness in the protocol.

For example, suppose that Alice is receiving messages that have been encrypted using RSA. The plaintext messages have to adhere to a certain format, and if a decrypted message is not in that form Alice's computer transmits an error message to the sender. This seems innocuous enough. However, Bleichenbacher [10] showed that the error messages sometimes might compromise security.

Bleichenbacher's idea can be illustrated if we consider a simplified version of the protocol that he attacked in [10]. Suppose that we are using RSA with a 1024-bit modulus n to send a 128-bit secret key m (for use in symmetric encryption). We decide to pad m by putting a random number r in front of it, but since this does not take up the full 1024 bits, we just fill in zero-bits to the left of r and m. When Alice receives our ciphertext, she decrypts it, checks that it has the right form with zero-bits at the left end—if not, she informs us that there was an error and asks us to resend—and then deletes the zero-bits and r to obtain m. In that case Bleichenbacher can break the system—in the sense of finding the plaintext message—by sending a series of carefully chosen ciphertexts

[1] In the case of the first author, "youth" refers not to chronological age at the time, but rather to the short period that he had been working in cryptography.

(certain "perturbations" of the ciphertext he wants to decipher) and keeping a record of which ones are rejected because their eth root modulo n is not of the proper form, that is, does not have the prescribed number of zero-bits.

In order to protect Alice and her friends from clever adversaries who are out to steal their secrets, we clearly need much more elaborate criteria for security than just non-invertibility of the underlying one-way function.

1.1. *The First System with Reductionist Security—Rabin Encryption*

Soon after the earliest papers [24], [53] on public-key cryptography appeared, many people started to realize that breaking a system was not necessarily equivalent to solving the underlying mathematical problem. For example, the RSA function $y = x^e \pmod n$ was constructed to be easily invertible by someone who knows the prime factors p and q of $n = pq$, but not by someone who does not. Number theorists thought that it was highly unlikely that someone would find a quicker way than factoring n to find eth roots modulo n. However, no one could really say for sure, and much more recently, work by Boneh and Venkatesan [16] suggests that inverting the RSA function might *not* be equivalent to factoring.

In 1979 Rabin [51] produced an encryption function that could be *proved* to be invertible only by someone who could factor n. His system is similar to RSA, except that the exponent is 2 rather than an integer e prime to $\varphi(n)$, where φ denotes the Euler phi-function. For n a product of two primes the squaring map is 4-to-1 rather than 1-to-1 on the residues modulo n, so Rabin finds all four square roots of a ciphertext y (and in practice chooses the plaintext that makes sense to the message recipient).[2]

The most important feature of Rabin's encryption scheme was that it was in some sense "provably" secure, that is, it had a reductionist security property.

Reductionist security claim. Someone who can find messages m from the ciphertext y must also be able to factor n.

Argument. Informally, the reason is that finding m means being able to find all four square roots of y, because any one of them could be the true plaintext m. Those square roots are $\pm m$ and $\pm \varepsilon m$, where ε is a residue mod n that is $\equiv 1 \pmod p$ and $\equiv -1 \pmod q$. That means that someone who can find messages must know the value of ε, in which case n can be factored quickly using the Euclidean algorithm, since $\gcd(n, \varepsilon - 1) = p$.

We also give a slightly more formal argument, called a "reduction," which will be the prototype for all the reductionist security arguments that come later. The idea of a reduction argument is that you can show that hardness of one problem $\mathcal{P}_1$ implies hardness of another problem $\mathcal{P}_2$—or, equivalently, that "easiness" of $\mathcal{P}_2$ would imply easiness of $\mathcal{P}_1$—by showing that anyone who had an algorithm to solve $\mathcal{P}_2$ could use it to solve $\mathcal{P}_1$ with relatively little additional effort; in that case one says that $\mathcal{P}_1$ reduces to $\mathcal{P}_2$. The most familiar use of reductions is in the theory of NP-completeness [28], where $\mathcal{P}_1$ is a well-known NP-complete problem such as 3SAT and $\mathcal{P}_2$ is another NP problem that you want to prove is NP-complete. In cryptography, $\mathcal{P}_1$ is a mathematical problem

[2] Williams [63] developed a variant of Rabin encryption in which a plaintext is modified in a simple manner so that the plaintext can be uniquely recovered from its square, that is, from the ciphertext.

such as factoring that is assumed to be difficult, and $\mathcal{P}_2$ is a certain specified type of successful attack on our cryptographic system.

Returning to Rabin encryption, we suppose that there exists an "adversary" that takes n and y as input and produces one of the square roots of y modulo n. We think of the adversary as a computer program, and we show how someone (whom we call Sam) who has that program could use it to factor n quickly. That is, the adversary is a set of instructions, available for anyone to use, that produce a successful attack on the encryption scheme. If we thought of it as a person, we might wonder whether it could change its behavior or refuse to cooperate with Sam the Simulator. If we think of it as a computer program, we will not be tempted to ask such silly questions.

What Sam does is the following. He chooses a random residue x, sets $y = x^2 \pmod{n}$, and inputs that value of y to the adversary. The adversary outputs a square root m of y mod n. With probability 1/2 the root m is $\pm\varepsilon x$, and in that case Sam can immediately compute $\varepsilon = \pm x/m$ and then factor n. If, on the other hand, $m = \pm x$, then the value of m will not help him factor n, and he tries again starting with a new value of x. There is only a $1/2^k$ chance that he will fail to factor n in k or fewer tries. We say that this argument "reduces" factoring n to breaking Rabin encryption mod n (where "breaking" means recovering plaintext messages). Rabin's scheme was the first public-key system to be proposed that was accompanied with a reductionist security argument. Users of Rabin encryption could be certain that no one could recover plaintexts unless they knew the factorization of n.

1.2. *Chosen-Ciphertext Attacks*

Soon after Rabin proposed his encryption scheme, Rivest (see [63]) pointed out that, ironically, the very feature that gave it an extra measure of security would also lead to total collapse if it were confronted with a different type of adversary, called a "chosen-ciphertext" attacker. Namely, suppose that the adversary could somehow fool Alice into decrypting a ciphertext of its own choosing. The adversary could then follow the same procedure that Sam used in the previous paragraph to factor n. An adversary who could trick Alice into deciphering k chosen ciphertexts would have a $1 - 2^{-k}$ probability of factoring n.

The original RSA enciphering scheme (which is often called "naive RSA" or "basic RSA" or "the RSA primitive") is also vulnerable to chosen-ciphertext attack. Namely, suppose that the adversary wants to find a message m from the ciphertext $y = m^e \pmod{n}$. If it is allowed to ask Alice for a single decryption, then it can take random $\widetilde{m}$, compute $y' = y\widetilde{m}^e \pmod{n}$, send Alice y' to decrypt, and divide the resulting m' by $\widetilde{m}$ to recover m. This is a serious weakness, but it is not as bad as the one that Rabin encryption suffers from, since the adversary succeeds in obtaining just one message. In the attack on Rabin it gets the factorization of n, and hence can read all subsequent secret communications to Alice.

At this point the reader might wonder why anyone should worry about a chosen-ciphertext attack. Cannot we assume that Alice knows enough not to give out decryptions to strangers?

In the first place, one can imagine scenarios where Alice might think that such a decryption request is reasonable—for example, if the system crashed and the attacker

appears to be a legitimate user who is trying to recover lost data, or if the attacker is someone who had legitimate business with Alice in the past, but now wants to steal some data sent to her by someone else, etc.

In the second place, if a system is secure against chosen-ciphertext attacks, then it is also secure against partial chosen-ciphertext attacks, that is, when the adversary obtains only part of the information that it would get in a full chosen-ciphertext attack. Such partial chosen-ciphertext attacks are very real possibilities. The best-known example is Bleichenbacher's attack [10] that we described earlier. He showed how to compromise the security of an RSA protocol that had been approved by standards bodies and is still used today in real-world applications. If an RSA protocol had been used that had been shown to be secure against chosen-ciphertext attacks, then Bleichenbacher's method would not have succeeded against it.

1.3. *Some Basic Concepts*

Starting in the 1980s it became clear that there is a lot more to security of a public-key cryptographic system than just having a one-way function, and researchers with a background in theoretical computer science set out to systematically develop precise definitions and appropriate "models" of security for various types of cryptographic protocols.

One of the seminal ideas of that period was probabilistic encryption [31], [32]. In public-key cryptography, where everyone has the information needed to encipher, deterministic encryption—in which a given plaintext is enciphered into one and only one possible ciphertext—has the drawback that the system is useless if the message is known to belong to a small set. To avoid this defect, one should use an encryption function $f_r(m)$ that depends on a random integer r as well as the message m. For example, one could append r to m before applying the RSA function; this is called an RSA "padding."

Another reason for the development of probabilistic encryption was to avoid the problem pointed out by Rivest (see the beginning of Section 1.2). That is, people wanted to be able to prove reductionist security results that do not come back to bite us in the way that the equivalence between factoring and Rabin decryption had done. The key point here is that the introduction of randomness into encryption greatly reduces the power of a chosen-ciphertext attacker. Such an adversary no longer gets Alice to give him the complete solution to the problem of inverting the basic encryption function. For example, in Rabin encryption, if Alice pads a message m with random r before squaring mod n, then the chosen-ciphertext attacker learns only m and not the full square root modulo n, which is the concatenation of m and r.

In the context of probabilistic encryption with a passive adversary (one that does not request decryptions of chosen ciphertexts), Goldwasser and Micali [31], [32] were able to define two strong notions of security. The first was that of *semantic security*. This notion can also be carried over to the chosen-ciphertext setting, where, informally speaking, it means that the attacker is unable to obtain any information at all (except for its bitlength) about the plaintext m^* that was encrypted to produce a ciphertext y^*, even if it is allowed to request the decryption of any ciphertexts of its choosing except for y^*. Another security notion in [31], [32] is called *indistinguishability*. This notion can be extended to cover chosen-ciphertext attacks using ideas from [48] and [52]. In that

setting, indistinguishability means that the attacker chooses two messages m_0 and m_1, one of which is then encrypted: $y^* = f_r(m_b)$, $b \in \{0, 1\}$. Even though the attacker is allowed to request the decryption of any ciphertexts it wants (except for y^*) both before and after it chooses m_0, m_1, it cannot guess which of the two messages was encrypted with significantly more than $1/2$ chance of success. These two strong notions of security are closely related; in fact, in [31], [32], and [46] they were proved to be equivalent against a passive adversary. However, for a long time it was not clear whether or not they are equivalent under active attacks.

It is quite surprising that the equivalence of semantic security and indistinguishability in the case of chosen-ciphertext attacks was not considered in the research literature until 2002. After all, semantic security is really the natural notion of what one should strive for in public-key encryption, while indistinguishability is seemingly an artificial notion. However, in practice it has been much easier to prove a public-key encryption scheme to be secure using indistinguishability than using the more natural definition, and so all proofs in the literature use it. The equivalence of indistinguishability and semantic security under chosen-ciphertext attacks has purportedly been proved in [61] and [30]. If these proofs are correct, then the matter has finally been settled.

The second important theoretical advance in the mid-1980s was the first work [33], [34] to give a definition of what it means for digital signatures to be secure. That definition has stood the test of time and is still widely used today. Goldwasser et al. replace "chosen-ciphertext" (used for encryption schemes) by "chosen-message" and replace semantic security/indistinguishability by the idea of an existential forger. That is, a signature scheme is said to be *secure against chosen-message attack by an existential forger* if an adversary that has been allowed to request valid signatures for messages m_i of its choosing is unable to produce a valid signature for any message that is different from all of the m_i. A typical reductionist security result for a signature scheme says that it is secure in this sense provided that certain assumptions (about hardness of an underlying mathematical problem, randomness of some numbers, and properties of hash functions) hold.

The idea that emerged in the 1980s of systematically using reduction arguments to convince oneself of the security of encryption and signature schemes is elegant and powerful. However, it is important always to keep in mind the limitations of the method. Obviously, it cannot guarantee resistance against attacks that are not included in the security definition. In particular, the usual security definitions do not account for attacks that are based on certain features of the physical implementation of a cryptographic protocol. Such *side-channel attacks* utilize information leaked by the computing devices during the execution of private-key operations such as decryption and signature generation. The kind of information that can be exploited includes execution time [40], power consumption [41], electromagnetic radiation [1], induced errors [13], and error messages [42].

Finally, we should mention two fundamental contributions in the 1990s to the theoretical study of security issues, both by Bellare and Rogaway. In [6] they studied the use of the "random oracle model" for hash functions in reductionist security arguments. The systematic use of the assumption that hash functions can be treated as random functions made it possible for security results to be obtained for many efficient and practical schemes. We shall have more to say about the random oracle model in later sections.

In addition, Bellare and Rogaway developed the notion of "practice-oriented provable security" (see [3]). As a result of their work, reductionist security arguments started to be translated into an exact, quantitative form, leading, for example, to specific recommendations about keylengths. The objective of the work has been to move the subject away from its roots in highly theoretical computer science and closer to real-world applications.

1.4. *Outline of the Paper*

This paper was written with four objectives in mind:

- to offer a point of view on provable security that differs from the prevailing one;
- to make the case that this field is as much an art as a science;
- to raise technical points about interpretation of the proofs that hopefully will cause people to think more deeply about what the reductionist arguments actually mean;
- to provide a tutorial introduction for non-experts to an area that is usually impenetrable to outsiders.

The main body of the paper consists of informal (but accurate) descriptions and analyses of the reductionist security arguments for four important practical public-key cryptographic systems. Two of them (in Sections 2 and 4) are encryption schemes (one based on RSA and one based on the so-called "discrete log" problem in the multiplicative group of a finite field), and two (in Sections 3 and 5) are signature schemes (one based on RSA and one on discrete logs). By presenting these constructions and results with as few technicalities as possible, we hope to make them accessible to the broad mathematical public. At the same time, some of the conclusions we draw from our analyses are in sharp disagreement with prevailing views.

In Section 6 we discuss some recent work that purportedly undermines the random oracle model, but which we argue actually supports it, and in Sections 7 and 8 we end with some technical conclusions and some informal remarks about whether "proving" security is an art or a science.

2. Cramer–Shoup Encryption

We start by describing the basic ElGamal encryption scheme [25]. Let G be the subgroup of prime order q of the multiplicative group of the prime field of p elements, where $q|p-1$, and let $g \in G$ be a fixed element (not the identity). (In practice, p might be a 1024-bit prime and q a 160-bit prime.) We suppose that p, q, and g are publicly known. Alice chooses a random integer $z, 0 < z < q$, as her private key; her public key is $e = g^z$.

The sender Bob's message m is an element of the finite field. To encrypt m, he first chooses a random $r, 0 < r < q$, and computes $u = g^r$ and $w = e^r m$. He sends the pair (u, w) to Alice, who deciphers it by dividing w by u^z (where she uses her secret key to compute $u^z = g^{zr} = e^r$).

Like naive RSA, this naive version of ElGamal is vulnerable to chosen-ciphertext attack. Namely, the attacker, who has the ciphertext (u, w) and wants to learn what m is, chooses r' and $\widetilde{m}$ at random, sets $u' = ug^{r'}$ and $w' = we^{r'}\widetilde{m}$, and gets Alice to decrypt (u', w') for him. She sends the attacker $m\widetilde{m}$, after which he simply divides by $\widetilde{m}$ to obtain m.

A first attempt to fix this problem might be to require that any ciphertext include a string of bits that can be used to test that it was formed correctly, that is, that whoever is sending the ciphertext knew the plaintext message from which it was generated. This string of bits is a sort of "fingerprint" of the message, called its "hash value."

More precisely, a *hash function* H is a map from strings of σ bits to strings of ν bits, where generally σ is much greater than ν, that is easy to compute but not to invert (this is called the "one-way" property). That is, given h, it is not feasible to find any m such that $h = H(m)$. Usually a somewhat stronger property, called "collision-resistance," is assumed. This means that it is not feasible to find any pair m, m' such that $H(m) = H(m')$.

If the ElGamal ciphertext includes $h = H(m)$, then Alice can verify that $(g^r, e^r m, h)$ is a valid ciphertext by checking that $h = H(m)$ after she decrypts. If the attacker sends her $(g^{r+r'}, e^{r+r'} m\widetilde{m}, h)$, then she will reject it, because $H(m\widetilde{m}) \neq h$. Note that we are assuming that the attacker will not be able to find any $\widetilde{m}$ such that $H(m\widetilde{m}) = h$.

The problem with this solution is that it fails completely in a situation where the message m is known to be one of a small set $\{m_i\}$ (for example, it is either "yes" or "no," or it is the date when an invasion is planned). In that case all the adversary needs to do is run through the values $H(m_i)$ until it comes to h.

Ideally, we would like the encryption system to have the property that, even if the adversary knows that the plaintext is one of two messages m_0 or m_1, and even if the adversary is allowed to choose the two messages and then see an encryption y^* of one of them, still it would not have appreciably more than a 50% chance of determining whether m_0 or m_1 had been encrypted—even if it can ask for decryptions of any ciphertexts of its choosing both before and after selecting the messages m_0, m_1 (except that after it receives the target ciphertext y^*, it is not, of course, allowed to ask for y^* to be decrypted). As mentioned in the Introduction, if the encryption scheme is secure in this sense, we say that it is *indistinguishability-secure from chosen-ciphertext attack*.

Before stating the Cramer–Shoup result, we need to describe three number-theoretic problems in the group G of prime order q:

The Discrete Logarithm Problem. Given an element $g \neq 1$ of G and another element u, find $x \in \{0, 1, \ldots, q-1\}$ such that $u = g^x$. (The solution x is called the "discrete log" of u to the base g and is often denoted either $\log_g u$ or else $\mathrm{ind}_g u$.)

The Computational Diffie–Hellman Problem. Given elements $g \neq 1$, u_1 and u_2 in G, find u_3 such that $x_3 \equiv x_1 x_2 \pmod{q}$, where $x_i = \log_g u_i$.

The Decision Diffie–Hellman Problem. Given a 4-tuple (g_1, g_2, u_1, u_2), determine whether or not $\log_{g_1} u_1 = \log_{g_2} u_2$. (A formulation that is easily seen to be equivalent is the following: Given a 4-tuple (g, u_1, u_2, u_3), determine whether or not $x_3 \equiv x_1 x_2 \pmod{q}$, where $x_i = \log_g u_i$.)

These three problems are listed in order of decreasing difficulty, in the sense that an algorithm that finds discrete logarithms can be used to solve the Computational Diffie–Hellman Problem immediately, and an algorithm that solves the Computational Diffie–Hellman Problem will also answer the decision version of the problem. The reverse implications are not known, although there is evidence that the first two problems may

be equivalent [43], [14] and the last two probably are not (the papers [11] and [37] give examples of groups where the Decision Diffie–Hellman Problem is easy but the Computational Diffie–Hellman Problem is believed to be hard).

2.1. *The Cramer–Shoup Encryption Scheme and Security Claim*

We are now ready to state the main result of Cramer and Shoup.

Reductionist security claim. If the Decision Diffie–Hellman Problem is hard in the group G and if the hash function H is collision-resistant,[3] then the modified ElGamal encryption scheme described below is indistinguishability-secure from chosen-ciphertext attack.

Description of the Cramer–Shoup encryption scheme. Let $x = (x_1, x_2)$, $y = (y_1, y_2)$, and $z = (z_1, z_2)$ denote pairs of integers between 0 and $q - 1$; let $g = (g_1, g_2)$ and $u = (u_1, u_2)$ denote pairs of elements of G; and let r denote a random integer between 1 and $q - 1$. We use the notation $g^x = g_1^{x_1} g_2^{x_2}$, $g^{rx} = g_1^{rx_1} g_2^{rx_2}$, and so on.

The group G of prime order q in the field of p elements (where $q|p - 1$) and two random non-identity elements g_1, g_2 are publicly known. We suppose that the collision-resistant hash function H takes triples of integers mod p to integers between 0 and $q - 1$ (that is, it takes bit-strings of length $3\lceil \log_2 p \rceil$ to bit-strings of length $\lceil \log_2 q \rceil$—in practice, this might mean that it maps 3072-bit numbers to 160-bit numbers). Alice's private key consists of three randomly generated pairs x, y, z, and her public key consists of the three group elements $c = g^x$, $d = g^y$, and $e = g^z$.

To send a message $m \in G$, Bob chooses a random r, sets $u_1 = g_1^r$, $u_2 = g_2^r$, and $w = e^r m$, and computes the hash value $h = H(u_1, u_2, w)$ of the concatenation of these three mod p integers. He computes $v = c^r d^{rh}$, and sends Alice the ciphertext 4-tuple (u_1, u_2, w, v). In this ciphertext the element v allows Alice to check that Bob enciphered the message properly (after which she is confident that he is not really an adversary using the attack on naive ElGamal that was described above); w contains the message m "disguised" by the "mask" e^r; and u_1 and u_2 are the clues she needs to remove the mask.

More precisely, to decipher the 4-tuple (u_1, u_2, w, v) Alice first computes $h = H(u_1, u_2, w)$ and uses her secret key to find u^{x+hy} (that is, $u_1^{x_1+hy_1} u_2^{x_2+hy_2}$), which should be equal to v (because $u^{x+hy} = g^{rx+ryh} = c^r d^{rh}$). If it is not equal to v, Alice rejects the message. If the ciphertext passes this test, she proceeds to decrypt by dividing w by u^z. Since $u^z = g^{rz} = e^r$ and $w = e^r m$, this gives her the plaintext m. This concludes the description of the cryptosystem.[4]

We say that a 4-tuple (u_1, u_2, w, v) is invalid—not a possible ciphertext—if $\log_{g_1} u_1 \neq \log_{g_2} u_2$. It is important to note that an invalid ciphertext will almost certainly be rejected by Alice. Cramer and Shoup explain this by regarding the two pairs x, y as elements of a four-dimensional vector space V over the field of q elements. Alice's public key—the relations $c = g^x$, $d = g^y$—constrain x, y to a two-dimensional subspace S. If $r = \log_{g_1} u_1 = \log_{g_2} u_2$, then the verification equation $v = u^{x+hy}$ holds either everywhere on

[3] Cramer and Shoup used a slightly weaker assumption in [23], namely, that H is a member of a universal one-way hash family; however, collision-resistance is just as good in practice. Note that they do *not* make the random oracle assumption.

[4] The actual Cramer–Shoup cryptosystem in [23] is the special case with $z_2 = 0$.

S or nowhere on S (depending on the value of v), whereas if $\log_{g_1} u_1 \neq \log_{g_2} u_2$, then the equation $v = u^{x+hy}$ imposes another independent linear condition on x, y (regardless of the value of v). In the latter case the 4-tuple (u_1, u_2, w, v) is accepted only if x, y lie on a particular line in S, and the chance of that is $1/q$, which is negligible. Note also that the rejection of such a ciphertext does not give an adversary any significant amount of information about (x, y); the adversary learns only that it cannot lie on the particular line of the plane S.

2.2. *The Reductionist Security Argument*

Reductionist security argument. We must show that if there is an adversary (which, as mentioned before, should be thought of as a computer program) that makes decryption queries and is eventually able to distinguish whether a target ciphertext y^* is an encryption of m_0 or m_1, then someone (whom we call Sam) is able to use this adversary program to determine whether or not a 4-tuple (g_1, g_2, u_1, u_2) has the Diffie–Hellman property $\log_{g_1} u_1 = \log_{g_2} u_2$. Sam can simulate attacks by the adversary, during which he wants to choose his input and responses to answer the Diffie–Hellman question.

So suppose that Sam is given a 4-tuple (g_1, g_2, u_1, u_2). He starts by playing the role of Alice generating her keys; that is, he chooses three random pairs x, y, z and sets $c = g^x$, $d = g^y$, $e = g^z$. He sends the public key (c, d, e) (and also g_1, g_2) to the adversary, which makes decryption queries (u_1', u_2', w', v'). Sam decrypts just as Alice would, rejecting the message unless $v' = u'^{x+h'y}$, where $h' = H(u_1', u_2', w')$, in which case he responds with the decryption w'/u'^z. When the adversary outputs the two plaintexts m_0 and m_1 for the distinguishability test, Sam chooses $b \in \{0, 1\}$ at random and sets the target ciphertext equal to $y^* = (u_1, u_2, w, v)$ with $w = u^z m_b$ and $v = u^{x+yh}$, where $h = H(u_1, u_2, w)$.

Now the adversary must decide whether y^* is the encryption of m_0 or m_1. If (g_1, g_2, u_1, u_2) has the Diffie–Hellman property, then y^* is a true encryption of m_b (with $r = \log_{g_1} u_1$ as Bob's random number), and so the adversary will have a probability significantly greater than $1/2$ of correctly guessing b.

What if the 4-tuple (g_1, g_2, u_1, u_2) does not have the Diffie–Hellman property? One possibility is that the adversary could fail; for example, it could output an error message, or it could fail to terminate during its expected running time. On the other hand, the adversary might function in the usual way and produce meaningful output whether or not the 4-tuple has the Diffie-Hellman property.

In that case the adversary can ask for decryptions of ciphertexts $y' = (u_1', u_2', w', v')$ that are not identical to $y^* = (u_1, u_2, w, v)$ in the hope of learning whether y^* encrypts m_0 or m_1. If, for example, the adversary could find $w' \neq w$ such that $H(u_1, u_2, w') = h = H(u_1, u_2, w)$, then it could set $y' = (u_1, u_2, w', v)$, which would pass Sam's test (since $v = u^{x+hy}$). Sam would give the adversary the decryption m', from which it could compute $m_b = m'w/w'$ just as in the chosen-ciphertext attack on naive ElGamal. The adversary's success would lead Sam falsely to believe that his 4-tuple has the Diffie–Hellman property (see the concluding paragraph of this argument). Fortunately, the adversary has negligible probability of finding $w' \neq w$ such that $H(u_1, u_2, w') = H(u_1, u_2, w)$ because the hash function has been assumed to be collision-resistant. That is, the only way the adversary could get the same hash value $h = H(u_1, u_2, w)$ is to take $y' = (u_1, u_2, w, v')$ with $v' \neq v$ (since $y' \neq y^*$), in which case Sam would reject y' because $v' \neq u^{x+hy}$.

Whenever the adversary asks for a decryption of an invalid 4-tuple (u'_1, u'_2, w', v'), Sam rejects it (except with negligible probability); this follows by an argument similar to the one at the end of Section 2.1. If the adversary asks for the decryption of a valid 4-tuple (u'_1, u'_2, w', v'), where $r' = \log_{g_1} u'_1 = \log_{g_2} u'_2$, then it learns the decryption $m' = w'/{u'}^z = w'/e^{r'}$ (it learns this provided that y' passes the test $v' = {u'}^{x+h'y}$). Nothing in the adversary's view, even after its decryption queries, constrains Sam's private key pair $z = (z_1, z_2)$ further than the single relation $e = g^z$ coming from the public key. In other words, all values of z in the plane over the field of q elements that lie on a certain line will agree with the adversary's view. However, only one point on that line—namely, the value of z that also satisfies the relation $w = u^z m_b$, which is independent of the relation $e = g^z$ because $\log_{g_1} u_1 \neq \log_{g_2} u_2$—is consistent with the choice of $b \in \{0, 1\}$. This means that (except with negligible probability) what the adversary sees is independent of b. So if Sam's 4-tuple fails to have the Diffie–Hellman property, the adversary will have only a 50% chance of guessing b.

In order to decide whether or not his 4-tuple has the Diffie–Hellman property, Sam will run this simulated attack several times with different x, y, z. If the adversary correctly determines b significantly more than half the time, Sam is almost certain that the 4-tuple has the Diffie–Hellman property. Otherwise, he is almost certain that it does not. This completes the reduction argument.

Cramer and Shoup designed their encryption scheme specifically so that they could get the above reductionist security argument to work. The desire to prove a certain type of security result motivated their construction. Instead of first setting up a natural cryptographic system and then trying to prove something about it, it has become increasingly common for protocol designers to start with a reductionist security objective that they want to achieve and then use it to guide them in their construction of a scheme.

Cramer–Shoup encryption attracted a lot of attention when it was proposed in 1998 because it was the first practical scheme for which a strong security property could be demonstrated under a "standard" hash function assumption (such as collision-resistance) rather than the stronger "random oracle model." However, this weakening of the hash function assumption came at a price: Cramer and Shoup needed to make a rather strong number-theoretic assumption, namely, hardness of the Decision Diffie–Hellman Problem. As mentioned on p. 11, this decision problem is likely to be strictly easier than the Computational Diffie–Hellman Problem; in fact, promising new cryptographic protocols have recently been developed (see, for example, [15]) based on the "gap" in difficulty between these two problems in certain groups. On the other hand, these "Diffie–Hellman gap groups" seem to be very exceptional (the best examples are supersingular elliptic curves and certain families of ordinary elliptic curves [37], [47]). In the groups that Cramer and Shoup would use, as far as we know there is no way to solve Decision Diffie–Hellman that is faster than finding discrete logarithms. So their assumption that the Decision Diffie–Hellman Problem is hard seems reasonable in the setting of their scheme.[5]

[5] It is worth noting that Shoup [56] has given a variant of the Cramer–Shoup encryption scheme for which he can show indistinguishability under chosen-ciphertext attack either with the assumptions in Section 2.1 or else with the assumption that the Computational Diffie–Hellman Problem is hard and the hash function is given by a random oracle (that is, with a weaker number-theoretic assumption and a stronger hash function assumption).

3. The Security of RSA Signatures

We first recall how Alice signs a message using the basic RSA system. Her public key consists of a modulus n (which is a product of two primes) and an encryption exponent e (which is relatively prime to $\varphi(n)$); and her secret key is a decryption exponent d (which is the inverse of e modulo $\varphi(n)$). To sign a message m (which is an arbitrary sequence of bits), Alice first applies a publicly known and easily computable hash function $H(m)$ which takes values in the interval $0 \leq H(m) < n$. We say that this is a "full-domain" hash function because its values range through the full interval $[0, n)$ rather than a smaller subinterval. Just as in encryption schemes, the hash function acts like a "fingerprint"; in practice, one usually assumes that it is computationally infeasible to find two different messages that have the same hash value ("collision-resistance"). In what follows we assume a stronger property, namely, that the hash function is chosen at random from the set of all possible functions taking values in the set $\{0, 1, \ldots, n-1\}$. We then say that we are "in the random oracle model."

Alice computes the least non-negative residue of $H(m)^d$ modulo n. That value s is her signature. Suppose that Bob receives the message m and the signature s. He computes $H(m)$ and also the least residue of s^e modulo n. If $H(m) \equiv s^e \pmod{n}$, then 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 one of the classic "provable security" results for the above RSA signature [6]. Our version is less formalistic than the published proofs, but it is essentially complete.

Recall from Section 1.3 the notion of an "existential forger" that is allowed to make "chosen-message attacks in the random oracle model." This means that the forger (which, as before, we should think of as a computer program, not a person) initially knows only Alice's public key (n, e). It is permitted to question Alice by sending her a sequence of (a bounded number of) test-messages m_i. In each case, if it is the first time the particular message has been queried, Alice responds by sending its hash value; if it is the second time, then she sends the signature.[6] At the end, with high probability the forger will be able to produce a valid signature for one of the messages m_i that Alice has not signed.

Notice that in the random oracle model the forger does not have an algorithm to produce a hash value, but rather must obtain these values one message at a time through queries.

Reductionist security claim. With the above notation and definitions, if the problem of inverting $x \mapsto x^e \pmod{n}$ 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 $y \equiv x^e \pmod{n}$. 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.

[6] There is no loss of generality here in supposing that Alice always supplies the hash value of a message before signing it. If she did not, then the adversary could easily compute the hash value from the signature.

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 $x_i^e \pmod n$. 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 $x \mapsto x^e \pmod n$ 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 a message m_i with $i \neq i_0$ is queried a second time, 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 $y \equiv x^e \pmod n$ with unknown x. If we guessed wrong and m_{i_0} was not the message that the forger ends up signing, then we will not 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 is the 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 $y \equiv x^e \pmod n$ for x is at most $(1 - 1/q)^k$. For large k, this approaches zero; so with high probability we succeed. This completes the argument.

3.1. *An Informal Analysis*

We now reexamine the claim and the argument to support it in a stripped down form, without superfluous features and cryptographic terminology. Because of the assumption regarding randomness of the hash function, the choice of messages m_i is irrelevant. What the forger has to work with is a random sequence of values h_i $(i \neq i_0)$ along with the corresponding x_i (which are the eth roots mod n), and the forger is required to produce the eth root mod n of a random h_{i_0}. The security claim is that this is no easier than producing the eth root mod n of a random number without having the sequence of pairs (h_i, x_i). The proof amounts to the rather trivial observation that, since both the h_i and the x_i are randomly distributed over the set of integers $\{0, 1, \ldots, n-1\}$, you can obtain an equally good sequence of pairs (h_i, x_i) by starting with the random x_i and finding $h_i = x_i^e \pmod n$. In other words, a sequence of random $(h_i, h_i^d \bmod n)$ is indistinguishable from a sequence of random $(x_i^e \bmod n, x_i)$. It makes no difference whether you look at your sequence of pairs left-to-right or right-to-left. Thus, the proof boils down to the following tautology: *the problem of solving an equation is equivalent to the problem of solving the equation in the presence of some additional data* (h_i, x_i) *that are irrelevant to the problem and that anyone can easily generate*.

There is a difference between the conclusions of the formal argument and the informal, "stripped down" one. In the reduction we saw that the forgery program would have to be used roughly $O(q)$ times (where q is the number of hash queries) in order to find the desired eth root modulo n. A result of Coron [21] shows that this can be improved to $O(q_s)$, where q_s denotes a bound on the number of signature queries.[7] (Thus, $q = q_s + q_h$, 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.) On the other hand, the informal argument

[7] In the above argument, instead of responding only to the i_0th hash query with $h_{i_0} = y$, Coron's idea was to respond to a certain optimal number $i_0, i_1, \ldots$ with $h_{i_j} = yz_j^e$ with z_j random.

basically says that, since the queries are of no help to the forger, the amount of time needed for the forgery is the same as for solving the RSA eth root problem.

From the standpoint of practice (as emphasized, for example, in [3]) this difference is important. What it means is the following. Suppose that you are using a large enough RSA modulus n so that you are confident that eth roots modulo n cannot be found in fewer than 2^{80} operations. Suppose that you anticipate a chosen-message attack where the attacker can get away with making up to 2^{20} signature queries. Then Coron's result says that you can be sure only that a successful forger will require time at least 2^{60}, whereas the informal argument says that the forger will need as much time, namely 2^{80}, as to find an eth root modulo n.

Assuming that Coron's result cannot be improved to give a tight reduction argument (which he essentially proves to be the case in a later paper [22]), we are confronted with a discrepancy between the informal argument and the result coming from formal reduction. Who is right? What is going on here?

3.2. *A Tale of Two RSA Problems*

We can shed light on this question by looking at the basic RSA problem (given n, e, and y, you are asked to find x with $x^e \equiv y \pmod{n}$) and the following variant, which we call RSA1—or, more precisely, RSA1(q_s, q_h). It can easily be seen that RSA1(q_s, q_h) and the forgery problem are tightly equivalent.

Given n, e, and a set of $q_s + q_h$ values y_i chosen at random from the set $\{0, 1, \ldots, n-1\}$, *you are permitted* (*at whatever stages of your work that you choose*) *to select up to q_s of those y_i for which you will be given solutions x_i to $x_i^e \equiv y_i \pmod{n}$. You must produce a solution $x_i^e \equiv y_i \pmod{n}$ for one of the remaining y_i.*

Obviously, if you can solve RSA, you can solve RSA1 in essentially the same time. It is easy to see the converse when $q_s = 0$. Namely, given n, e, and y, you create an instance of RSA1$(0, q_h)$ by choosing random $z_1, \ldots, z_{q_h}$ and setting $y_i = yz_i^e$. An answer to RSA1$(0, q_h)$ is a solution $x_i^e \equiv y_i \pmod{n}$ for one of the y_i. Then just set $x = x_i/z_i$.

The above informal argument for the equivalence of forgery and the RSA problem boils down to the claim that the two problems RSA and RSA1(q_s, q_h) are indistinguishable in practice, in the sense that if you can solve RSA1(q_s, q_h) (for some fixed q_s, q_h), then you can solve the RSA problem in essentially the same amount of time. However, the best reduction known from RSA to RSA1—the one obtained by translating Coron's construction [21] to the terminology of these two problems—leads to the weaker conclusion that you can solve RSA in time of order $O(q_s)$ times the amount of time needed to solve RSA1.

Is it reasonable to conjecture the equivalence in practice of RSA and RSA1? It seems to us that in this context it is. After all, a very strong assumption is already being made—that the RSA problem is intractable. In practice, what people mean by making that assumption is that the integer factorization problem is hard, and that there is no way to solve the RSA problem that is faster than factoring n. This assumption is being made in the face of the dramatic result of Boneh and Venkatesan [16] that makes it doubtful that there is a reduction of factorization to the RSA problem. Despite this negative result,

though, people continue making the assumption that in practice the RSA problem is indistinguishable from factorization. We maintain that the equivalence in practice of RSA and RSA1 is at least as plausible as the equivalence of RSA and factoring.

Notice that in neither case is the word "equivalence" meant in the sense of a reduction argument. Because of [22] it is not reasonable to hope for a tight reduction from RSA to RSA1, and because of [16] it is not reasonable to hope for a reduction from factoring to RSA. However, in both cases it *is* reasonable to believe that in the foreseeable future no one will find a way to solve RSA1 without being able to solve RSA in essentially the same amount of time and no one will find a way to solve RSA without being able to factor the modulus.

Tight formal reductions are nice to have. However, sometimes in cryptography one wants to assume that $\mathcal{P}_1$ is as hard as $\mathcal{P}_2$ even if there is no prospect of constructing such a reduction from $\mathcal{P}_2$ to $\mathcal{P}_1$.

3.3. *Pass on PSS*

If we are correct about the equivalence in practice of the RSA and RSA1 problems, then one consequence is that the Probabilistic Signature Scheme (PSS) version of RSA signatures [8]—which was constructed in order to have a tight reduction argument between the RSA problem and chosen-message existential forgery—gives no more security against chosen-message attacks than does the original full-domain hash version.

Let us take an informal look at the question of the relative security of the two signature schemes. We first describe (a slightly simplified version of) PSS. Here the signer first pads the message m with a random sequence of bits r and then proceeds as before. That is, she applies the hash function to (m, r) and then gets s by raising $H(m, r)$ to the power of her decryption exponent d modulo n. Her signature is not just s, but the pair (s, r), since Bob has to know r in order to verify that $s^e \equiv H(m, r) \pmod{n}$.

The basic reason why one can make a tight reductionist security argument for PSS is that the simulator can distinguish between two types of hash function queries—those that follow a signature query and are needed in order to verify the signature, and those that do not. In the former case the r in the (m, r) that is queried was supplied by the simulator, and in the latter case it is chosen by the adversary. The simulator answers the first type of hash query by setting $H(m, r) \equiv s^e \pmod{n}$; he answers the second type of query by choosing random z and setting $H(m, r) \equiv z^e y \pmod{n}$, where y is the input to the RSA problem (see the proof of equivalence of RSA and RSA1$(0, q_h)$ at the beginning of Section 3.2). The reduction argument is tight because the simulator no longer has to guess which message m the adversary will end up signing—he knows that it will be one of the messages for which the adversary, not the simulator, chooses the random padding.

If all we care about is formalistic "proofs," then we might easily be misled into thinking that the PSS system is provably more secure than the original RSA signature scheme. If we use the language of practice-oriented provable security, then we might say that by switching to PSS we have gained a factor of q_s in the time a forger requires to complete its nefarious task. As we have seen, such a conclusion is highly debatable.

Unless one believes that there is a difference in real-world intractability between the RSA problem and the RSA1 problem, there is no point in wasting time and energy on

replacing classical full-domain hash RSA by PSS. One possible reason, in fact, for not adopting PSS is that it requires an additional cryptographic assumption—randomness of some r-value—that is absent from the original full-domain hash RSA. Since randomness is often poorly implemented in practice, it is wise to avoid such a step if it is easy to do so, as in this case. Our conclusion is that, despite a quarter century of research on RSA, the simple "hash and exponentiate" signature protocol that has been known since the late 1970s (with the condition that the image of the hash function be the full set of residues) seems still to be the one to use.[8]

3.4. *A Variant of PSS*

Before leaving the topic of RSA signature schemes, we look at a recent construction of Katz and Wang [38] that is similar to PSS but more efficient. They show that instead of the random string r one need only take a single random bit.

More precisely,[9] 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 $s = (H(m, b))^d$ modulo n; her signature is the pair (s, b). To verify the signature, Bob checks that $s^e \equiv H(m, b) \pmod{n}$.

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 $y \equiv x^e \pmod{n}$. 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 eth root of t_2^e or an eth 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 eth root after running the forger k times. This gives us a tight reduction from the RSA problem to the forgery problem.

In order to better see the relationship between Katz–Wang, PSS, and the basic RSA signature, we consider the following variant of the RSA problem, which we call RSA2(q_s, q_h):

Given n, e, and a set of $q_s + q_h$ pairs of values (y_i, z_i) chosen at random from the set $\{0, 1, \ldots, n-1\}$, you are permitted (at whatever stages of your work that you choose)

[8] Our purpose here is to highlight the comparison between PSS and full-domain hash RSA. There may, however, be other RSA-type signature schemes that are superior to both. For example, Bernstein [9] argues in favor of a version of Rabin signatures [51] for which he gives a tight reductionist security argument.

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

to select up to q_s of those pairs for which you will be given the eth root modulo n of exactly one (randomly selected) element of the pair. You must produce an eth root of either element in one of the remaining pairs.

The RSA2 problem has the same relationship to Katz–Wang as the RSA1 problem has to the basic RSA signature. The above argument gives a tight reduction from the RSA problem to RSA2, but we have no tight reduction from the RSA problem to RSA1. Thus, from the standpoint of reduction arguments RSA2 is as hard as the RSA problem, whereas RSA1 might be easier than both by the factor q_s, which might be quite large, say $q_s \approx 2^{20}$. However, when we put the two problems RSA1 and RSA2 side by side and compare them, we see that it defies common sense to suggest that in practice someone would find RSA1 easier to solve than RSA2. By the same token, it seems to us implausible that the Katz–Wang system could be appreciably more secure than the original RSA signature with full-domain hash function.

4. The Search for Optimal RSA Encryption

In 1994, Bellare and Rogaway [7] proposed a protocol for encrypting messages that they called Optimal Asymmetric Encryption Padding (OAEP). Their method was mainly intended to be used with the RSA function $y = x^e \pmod n$—in which case it was called RSA-OAEP—but it could also be used with other trapdoor one-way functions. This was the first time that a practical public-key encryption scheme was proposed along with an accompanying reductionist security argument (see below).

Here is how OAEP works. We send a plaintext m of μ bits with ν zero-bits appended. Let m^0 denote the resulting σ-bit string, where $\sigma = \mu + \nu$. The purpose of the zero-bits is to ensure that only an insignificant proportion of all σ-bit strings are of the required form. As in Cramer–Shoup encryption (see Section 2), Alice performs a test before accepting a message: she first checks that a deciphered bit-string m^0 is of the proper form and, if not, rejects it.

In addition to her RSA public key (n, e) and private key d, Alice chooses a random function G from ν-bit strings to σ-bit strings and a random function H from σ-bit strings to ν-bit strings. The two functions G and H are public knowledge; that is, the sender Bob can compute them. A reasonable choice of μ, ν, and $\sigma = \mu + \nu$ might be 864, 80, and 944, respectively, if Alice is using a 1024-bit modulus n. The bitlength of the modulus should be $\sigma + \nu = \mu + 2\nu$.

If Bob wants to send a μ-bit message m to Alice, he first chooses a ν-bit random number r. He evaluates $G(r)$, sets m^0 equal to m with ν zeros appended, and computes $s = m^0 \oplus G(r)$ (this is the XOR of m^0 and $G(r)$, that is, the componentwise sum modulo 2). Next, he evaluates $H(s)$, computes $t = r \oplus H(s)$, and sets x equal to s concatenated with t. See Fig. 1. Finally, he exponentiates $y = x^e \pmod n$; his ciphertext is y.

Alice starts her decryption as in naive RSA, using her secret exponent d to find x. She applies H to s, which is the first σ bits of x, and takes the XOR of $H(s)$ with the last ν bits of x; this gives her r. She then evaluates $G(r)$ and takes the XOR of $G(r)$ with s to get m^0. If m^0 does not have ν zero-bits at the end, she rejects the message. If m^0

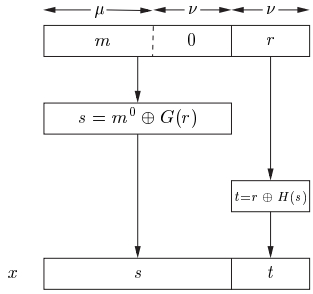

Fig. 1. OAEP padding.

does have the expected number of zero-bits, she deletes them to recover m. Notice that the steps in Alice's decryption after the exponentiation are very simple, much like the "back substitution" stage in an elementary linear algebra problem.

The idea of XOR-ing m^0 with $G(r)$ and then XOR-ing r with $H(m^0 \oplus G(r))$ is a familiar construction from classical symmetric cryptography, called a "two-round Feistel cipher." (The famous DES is a 16-round Feistel cipher.) Its purpose is to scramble everything thoroughly, while allowing rapid and simple decryption. The other two features of OAEP are the ν zero-bits (used to give Alice a way to check legitimacy of the text) and the ν bits of padding r.

4.1. *Reductionist Security of RSA-OAEP*

We now give an informal version of the original reductionist security argument of Bellare and Rogaway [7]. Note that G and H are assumed to be random functions; that is, the argument uses the "random oracle" assumption.

Reductionist security claim. The encryption system RSA-OAEP is indistinguishability-secure against chosen-ciphertext attack in the random oracle model if the RSA problem is intractable.

Argument. What the claim says is the following. Suppose that there is an adversary (which again should be thought of as a computer program) that takes the RSA public key (n, e) as input, makes some queries for values of the functions G and H and for decryptions of various ciphertexts that it chooses, and then selects two plaintext messages m_0 and m_1. It is now given an encryption y^* of one of the m_b (with $b \in \{0, 1\}$ chosen at random), after which it makes some more queries (but it is not, of course, allowed to ask for the decryption of y^*), and finally guesses b with probability significantly greater than $1/2$ of being correct. If such an adversary exists, then Sam the Solver (also known as Sam the Simulator), who has been given an arbitrary y, will be able to interact with the adversary in such a way as to determine the solution x of $x^e \equiv y \pmod{n}$. We make the reductionist security argument by describing how Sam sets up a simulated attack that he uses to find x.

The description of what Sam does in this simulated attack is simple. He sets y^* equal to the value y for which he must find the eth root modulo n, and he uses random values for $G(r)$ and $H(s)$ when answering queries for such values, unless the same value of r

or s has been queried before, in which case he must give the same answer. Sam keeps a record of all r and s that are queried and all of his answers $G(r)$ and $H(s)$ to those queries.

In response to the attacker's request for the decryption of some $y \neq y^*$, Sam runs through the previously queried $(r, G(r))$ and $(s, H(s))$. In each case he checks whether the concatenation of s and $r \oplus H(s)$ gives y when raised to the eth power modulo n. If it does, then he sets $m^0 = s \oplus G(r)$ and checks whether the v rightmost bits of m^0 are zero. If so, he deletes those zero-bits and outputs the resulting message m; if not, he outputs the words "invalid ciphertext." If Sam finishes running through all of the previously queried pairs (r, s) and none leads to a valid decryption m, then Sam answers the decryption query with "invalid ciphertext."

At some point the attacker sends Sam m_0 and m_1, and Sam responds by sending y^* (which is the integer whose eth root modulo n Sam wants to find). One question that might occur to the reader is: What if y^* is not the encryption of m_0 or m_1? In that case it would not be proper input to the adversary. The answer is that the adversary must function as if the values of G and H are values chosen at random at the time that a query is made; this is what the random oracle assumption says. The values of G and H that are used in the encryption of m_b almost certainly have not been specified at the time when y^* is given. Since it is easy to see that there exist values s^*, $H(s^*)$, r^*, and $G(r^*)$ such that the concatenation of s^* with $r^* \oplus H(s^*)$ is the eth root of y^* modulo n and such that $s^* \oplus G(r^*)$ is m_b^0, the adversary must be able to guess m_b when the target ciphertext is y^*.

We are now ready to conclude the Bellare–Rogaway argument. Since the target message m_b^0 is equal to $s^* \oplus G(r^*)$ with $G(r^*)$ random (where s^* is the first σ bits of the eth root of y^* modulo n), the only way the adversary would know any information about m_b^0 (and hence m_b) is to have queried $G(r^*)$. This means that the adversary must have known what value of $r = r^*$ to query. Since $r^* = t^* \oplus H(s^*)$ (where t^* is the last v bits of the eth root of y^* modulo n), the adversary must also have queried $H(s^*)$. Now Sam has a record of the r, s, $G(r)$, $H(s)$ from all of the queries. He runs through all pairs (r, s) in his list of queries and in each case checks to see whether the concatenation of s and $r \oplus H(s)$ gives y^* when raised to the eth power modulo n. When he gets to the pair (r^*, s^*), he finds the desired solution to the RSA problem. This concludes the argument for indistinguishability-security of RSA-OAEP against chosen-ciphertext attack.

If we think for a minute about this argument, we are struck by how much it seems to give for little effort. Not only do we get a strong form of security against powerful attackers, but the argument does not use any specific features of RSA or any assumptions about our parameters μ and v; in fact, the reductionist security claim seems to apply equally to the OAEP construction with RSA replaced by any other trapdoor one-way function (and, indeed, such a generalization was given in [7]). We might get suspicious, and remind ourselves of the adage, "If it's too good to be true, then it probably isn't." Our first response might be to doubt the validity of the random oracle model, which seems to allow Sam to "cheat" by leaving the values of the functions to be defined later (whereas in practice an algorithm for computing them is publicly available from the beginning).

Another possible response is to point out the lack of tightness in the reduction. That is, Sam must check all pairs (r, s) that were queried until he comes to (r^*, s^*). Since the

number of queries could be of the same order of magnitude as the adversary's running time t, this means that the upper bound on the number of steps Sam needs to solve the RSA problem is $O(t^2)$. In other words, if we want a guarantee that the adversary will need at least 2^{80} steps to succeed in its attack (in which case we say that we have "80 bits of security"), the above argument says that we should ensure that the RSA problem requires at least 2^{160} steps. According to present estimates of the running time of the number field sieve factoring algorithm, we should use roughly a 5000-bit RSA modulus to achieve 80 bits of security. This would be pretty inefficient. So the practical guarantee we get is not very useful.

4.2. *Shoup's Objection*

However, the best objection to the "too good to be true" reductionist security argument in [7] was discovered only 7 years later, and it caught everyone by surprise. In 2001 Shoup [57] showed that the argument is fallacious, because it assumes that the adversary cannot work with the input y^* to get useful information.

As Shoup pointed out, it is, for example, conceivable that the adversary is able to find s^* (which is the first σ bits of the solution x^* to the equation $x^{*e} \equiv y^* \pmod{n}$) and is also able to solve the following problem: *Given y^* and a subset S of at most $\nu + 1$ bit positions, determine y' such that the bits of the e*th *roots modulo n of y^* and y' agree except on the subset S, where they disagree*. No one knows how to solve this problem without being able to find eth roots modulo n, but the possibility cannot be ruled out. In such a case the adversary would find s^* and query $H(s^*)$. It would then set s' equal to s^* with the first bit switched, query $H(s')$, and choose the set S of at most $\nu + 1$ bits to consist of the first bit along with the bits in the last ν positions that correspond to one-bits of $H(s^*) \oplus H(s')$ (in other words, the positions where the bits of $H(s^*)$ and $H(s')$ differ). The adversary would find y' such that the solution x' to $x'^e \equiv y' \pmod{n}$ differs from x^* in precisely the bits of S. It would then ask for the decryption of y'. It is easy to see that the answer to this decryption query must be the secret plaintext m_b with its first bit switched. Thus, the adversary will have succeeded without ever needing the random r^* used for the target ciphertext. So the argument that $G(r^*)$ must have been queried is wrong.

Fortunately, Shoup [57] also showed that not all was lost for RSA-OAEP. First, the original argument of Bellare and Rogaway was partly correct, in that the adversary must query $H(s^*)$. Moreover, Shoup showed that if ν is much less than σ (which would be true in practice) and if $e = 3$ (a restriction that was later removed by Fujisaki et al. [27]), then the Bellare–Rogaway reductionist security claim is valid for RSA-OAEP (but not for OAEP with other trapdoor one-way functions). The crucial point is that if you know s^* and y^*, then the equation $x^{*3} = (2^\nu s^* + t^*)^3 \equiv y^* \pmod{n}$ can be solved for t^* using Coppersmith's method [20] for finding small roots of polynomials modulo n.

In addition, Shoup [57] proposed a modification of OAEP, which he called OAEP+ ("optimal asymmetric encryption padding plus"), for which he showed that the original Bellare–Rogaway argument is valid. He modified OAEP by replacing the string of ν zeros in m^0 by $\widetilde{H}(m, r)$, where $\widetilde{H}$ is another random function from strings of σ bits to strings of ν bits, and defining s as $m \oplus G(r)$ concatenated with $\widetilde{H}(m, r)$, where G now maps ν-bit strings to μ-bit strings.

4.3. *Boneh Brings Us Back to Rabin*

At this point it might have appeared that the search for optimal RSA encryption had ended with OAEP+. However, Boneh [12] was able to improve upon Shoup's result. He simplified the construction by reducing the number of Feistel rounds from two to one, and he showed that the reductionist security claim still holds. In other words, Boneh would apply the RSA function to $((m, \widetilde{H}(m, r)) \oplus G(r),\ r)$.

However, Boneh shows that it is actually much better to apply Rabin encryption (squaring modulo n, see Section 1.1) rather than the RSA function, for two reasons: (1) the assumption that finding eth roots modulo n is hard is replaced by the weaker and more natural assumption that factoring n is hard (this was why Rabin introduced his encryption method in 1979); and (2) the reduction argument is tight.

Of the three objections to OAEP mentioned above—use of the random oracle model in the security argument, lack of tightness in the reduction, and a fallacy in the original reductionist security argument—only the first one would apply to Boneh–Rabin. However, in Section 6 we describe evidence that we believe supports the reliability of the random oracle assumption. Thus, in our view Boneh–Rabin is currently the optimal choice for an RSA-type encryption system.[10]

It is ironic that after a quarter century we come full circle and return to Rabin encryption, which was the very first public-key system designed to have a reductionist security property. Boneh shows that all it needed was some random padding and hashing and just one Feistel round to go from being totally insecure to totally secure against chosen-ciphertext attack.

Unfortunately, the Boneh–Rabin system will probably not be widely used in the real world, at least not any time soon. The first reason is psychological. Cryptographers still think of Rabin encryption as the classic example of complete vulnerability to chosen-ciphertext attack. There is a saying, "Once bitten, twice shy." Even if Boneh's randomized Rabin is immune from chosen-ciphertext attack, people will wonder whether perhaps there is some other extreme vulnerability that is hiding around the corner. This fear is not logical (since there is no basis for believing that such a possibility is more likely for Boneh–Rabin than for any other system), but it is powerful nonetheless.

4.4. *The Credibility Problem*

The second reason is that "provable security" results seem to have much less credibility in the real world than one might expect. Non-specialists usually find the terminology and formalistic proofs hard to penetrate, and they do not read the papers. Moreover, the strange history of OAEP—where a "proof" was accepted for 7 years before a fallacy was noticed—hardly inspires confidence. Stern et al. [59] comment:

> Methods from *provable security*, developed over the last twenty years, have been recently extensively used to support emerging standards. However, the fact that proofs also need time to be validated through public discussion was somehow overlooked. This became clear when Shoup found that there was a gap in the widely believed security proof of OAEP against

[10] A slight disadvantage of Boneh–Rabin is that the plaintext m has bitlength at most half that of the modulus.

> adaptive chosen-ciphertext attacks.... the use of provable security is more subtle than it appears, and flaws in security proofs themselves might have a devastating effect on the trustworthiness of cryptography.

One has to wonder how many "proofs" of security are ever read carefully with a critical eye. The purported proof of Bellare and Rogaway in [7] was short and well written, and the result attracted much interest (and caused OAEP to be included in the SET electronic payment standard of MasterCard and Visa [3]). If this proof went essentially unexamined for 7 years, one cannot help asking whether the lengthy and often poorly written "proofs" of less famous security claims are ever read carefully by anyone.

In theoretical mathematics, one of the reasons why theorems engender confidence and trust is that the proof of a major result is almost always scrutinized carefully by referees and others before publication. The most famous example of this occurred in 1993, when Andrew Wiles submitted his 200-page manuscript purporting to prove Fermat's Last Theorem. Within 2 months a referee found a subtle gap in the long, extremely difficult proof—a gap that was fixed a little over a year later by Taylor and Wiles [60], [62]. A more recent example of the scrutiny that a dramatic new result gets in mathematics is the response to the theorem that "Primes is in P" of Agrawal et al. [2]. Their proof, while ingenious, was relatively short and elementary; within a few days of the paper's posting, several of the world's top number theorists had meticulously gone over it with a fine-tooth comb and found it to be correct.

We would feel a little more at ease with "provable security" results if the same tradition of careful examination of all important papers existed in theoretical cryptography.

* * *

The last three sections have dealt with Diffie–Hellman type encryption (that is, encryption using a discrete-log-based primitive), RSA-type signatures, and RSA encryption. We next consider a signature scheme based on a discrete-log primitive.

5. Schnorr Signatures and the Forking Lemma

5.1. *The Equivalence of Schnorr Signature Forgery and Discrete Logs*

We first describe Schnorr's method [55] for signing a message. As in Section 2, let q be a large prime, and let p be an even larger prime such that $p \equiv 1 \pmod{q}$. In practice, roughly $p \approx 2^{1024}$ and $q \approx 2^{160}$. Let g be a generator of the cyclic subgroup G of order q in the multiplicative group of integers mod p. (It is easy to find such a generator by raising a random integer to the $((p-1)/q)$th power modulo p.) Let H be a hash function that takes values in the set $\{0, 1, \ldots, q-1\}$. To prepare for signing messages, Alice chooses a secret random integer x, $0 < x < q$, which is her private key, and computes $g^x \pmod{p}$, which is her public key. To sign a message m, Alice first chooses a random k, $0 < k < q$, where k must be chosen again for each new message. She computes $r = g^k \pmod{p}$, where r is regarded as a non-negative integer less than p, and evaluates the hash function H at the message m concatenated with r; we set $h = H(m, r)$. Finally, she sets s equal to the least positive residue of $k + hx$ modulo q. Her signature is the pair (h, s).

To verify the signature, Bob divides g^s by $(g^x)^h$. (Note that g and g^x are publicly known, and he knows h and s from the signature.) If Alice formed the signature correctly, the result agrees with $g^k = r$, and hence

$$H(m, g^s(g^x)^{-h}) = h.$$

If this equality holds, Bob accepts the signature.

Clearly, if a prospective forger can solve the discrete logarithm problem in G, then it can forge signatures, because it can determine x from the public key g^x. As mentioned in the Introduction, an important question to ask at this point is whether the converse is true. There is a strong argument that it is.

Reductionist security claim. If an adversary can forge Schnorr signatures on all messages, then it can find discrete logarithms (in essentially the same amount of time that it takes to forge a signature).

Argument. Suppose that the adversary can forge a signature for m. When it computes $h = H(m, r)$, suppose that it is suddenly given a second hash function H'. Since, by assumption, 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'. (A more formal proof would use the random oracle model for the hash function.) So the forger computes $h' = H'(m, r)$ 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 $h' \neq h$.[11] Note that the value of k is the same in both cases, since r is the same. By subtracting the two congruences $s \equiv k + xh$ and $s' \equiv k + xh' \pmod{q}$ and then dividing by $h' - h$, the forger finds the discrete log x.[12]

This security result is fairly weak, because we have not allowed chosen-message attacks, and the adversary is assumed to be able to forge a signature on an arbitrary message. Later we describe the more complicated argument that is needed to get a stronger security result. However, before discussing the so-called "forking lemma," we make a historical digression.

5.2. *DSA and NSA*

In the early 1990s, the U.S. government's National Institute of Standards and Technology (NIST), following the advice of the National Security Agency (NSA), proposed a Digital Signature Algorithm (DSA) that combined features of an earlier scheme of ElGamal [25] with the Schnorr signature described above. There were two main differences with Schnorr's scheme: first, in DSA the hash function $h = H(m)$ is evaluated only as a function of the message m and does not depend on the randomly generated r. Second, the congruence $s \equiv k^{-1}(h + xr) \pmod{q}$ (rather than $s \equiv k + xh$) is used.

At the time, the proposed standard—which soon after became the first digital signature algorithm ever approved by the industrial standards bodies—encountered stiff

[11] Strictly speaking, we should allow for the possibility that the 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 omit these details. In Section 5.4 we give a more rigorous argument than here for a stronger security result for Schnorr signatures.

[12] Note that the forger does not need to know k.

opposition, especially from advocates of RSA signatures and from people who mistrusted the NSA's motives. Some of the leading cryptographers of the day tried hard to find weaknesses in the NIST proposal. A summary of the most important objections and the responses to them was published in the Crypto '92 proceedings [17]. The opposition was unable to find any significant defects in the system.

In retrospect, it is amazing that none of the DSA opponents noticed that when the Schnorr signature was modified, the equivalence with discrete logarithms was lost. In other words, there is no argument known that shows that the ability to forge DSA signatures implies the ability to find discrete logs. In particular, if you try to repeat the argument used above for the Schnorr signature, you find that the forgery program can still be made to produce two different signatures for m with different h, but cannot be forced to use the same value of r, since it does not have to choose r before the hash function is evaluated.[13]

This reductionist security failure is a much more serious matter than any of the issues that the anti-DSA people raised in 1992 [17]. It is also surprising that apparently none of the NSA cryptographers noticed this possible objection to DSA; if they had, they could have easily fixed it (without any significant loss of efficiency) by having the signer evaluate the hash function at (m, r) rather than just at m.

The debate over DSA lasted many months and pitted powerful companies and institutions against one another. Reputations and large sums of money were at stake. How could everyone have missed an elementary observation that could have changed the course of the debate?

5.3. *The "Splitting Lemma"*

The "splitting lemma" is the name cryptographers have given to a simple but useful observation that we shall need when we modify the reductionist security argument in Section 5.1 to allow for a more limited type of forgery. We shall want to know that even if a forger succeeds only with a certain non-negligible probability, it can still find discrete logs.

Suppose that we have two sets A and B, with a elements in A and b elements in B. Suppose that εab of all pairs (α, β), where $\alpha \in A$, $\beta \in B$, have a certain property (here $0 < \varepsilon < 1$ is the probability that a random pair has the property). We say that a pair with the property is a "good" pair. In our later application a pair is "good" when the algorithm uses that pair to produce a useful output. Let $A_0 \subset A$ be defined as the set of elements α_0 such that a pair (α_0, β) (with $\alpha_0 \in A$ fixed and $\beta \in B$ varying) has a probability $\geq \varepsilon/2$ of being good.

The "splitting lemma" says that there are at least $\varepsilon a/2$ elements in A_0; in other words, an element $\alpha \in A$ has a probability at least $\varepsilon/2$ of being in A_0. To see this, suppose the contrary. We count the number of good pairs (α, β) with $\alpha \in A_0$ and the number with $\alpha \notin A_0$. By assumption, the first number is at most $\#(A_0)b < (\varepsilon a/2)b$. By the definition

[13] Aside from the question of whether or not equivalence can be shown between forging signatures and solving the discrete log problem, it should have been clear to people in 1992 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 does not "get the last word."

of A_0, the second number is at most $\#(A \backslash A_0)\varepsilon b/2 \leq a\varepsilon b/2$. Then there are fewer than $(\varepsilon a/2)b + a\varepsilon b/2 = \varepsilon ab$ good pairs in all, and this is a contradiction.

5.4. *The "Forking Lemma"*

Following [49] and [50], we return to the forger in Section 5.1, but now make a weaker and more realistic assumption, namely, that the signature scheme is attacked by a probabilistic chosen-message existential forger in the random oracle model; the reductionist security claim is then that such an attack would imply the ability to find discrete logarithms. The term "chosen-message existential forger in the random oracle model" was explained in Section 3. The word "probabilistic" means that the forger (which, as before, we should think of as a computer program, not a person) is supplied with a long sequence of random bits that it uses to make choices at various points in its work. We consider the set of all random functions (that is, random values given in response to hash and signature queries) and all sequences of random bits, and assume that with non-negligible probability at least ε the forger is successful in producing a forged signature.[14]

Reductionist security claim. If the Schnorr signature succumbs to attack by a probabilistic chosen-message existential forger in the random oracle model, then discrete logs can be found.

Argument. Suppose that we have such a forger, which we want to use to find the discrete logarithm x of the public key g^x. The forger is allowed to make a bounded number of signing inquiries and hash function inquiries. Our response to any such query is to give a random value h (if it is a hash query) or pair of random values (h, s) (if it is a signature query), except that the same value of h must be given if the forger asks for $H(m, r)$ twice. Note that since $r = g^s(g^x)^{-h}$ (see Section 5.1), the random choice of s implies a random choice of $r \in G$. In the unlikely event that r turns out to be a value such that we already responded to a query for $H(m, r)$ with the same m and r and gave a different $h' = H(m, r)$, then we have to restart the whole procedure. The same applies if we answered an earlier query for a signature of the same message m by giving (h', s') with $g^{s'}(g^x)^{-h'}$ equal to r but $h' \neq h$. However, the probability of either unfortunate coincidence occurring is negligible.

Let q_h be a bound on the number of pairs (m_j, r_j) for which the forger queries the hash function. We choose a random index $j \leq q_h$ and hope that (m_j, r_j) happens to be the one for which the forger produces a signature. By the definition of our forger, with some non-negligible probability ε it will forge a signature, and so with probability at least ε/q_h it will forge a signature for the jth pair for which it queries the hash function.

We use two copies of the forger (that is, two copies of the same computer program). We give both forgers the public key g^x, the same sequence of random bits, and the same random answers to their queries for hash function values and signatures until they both ask for $H(m_j, r_j)$. At that point we give two independent random answers to the hash function query, and from then on use different sequences of random bits and different

[14] In general, the probability space should be taken to be the ordered *triples* consisting of function, sequence of random bits, public key. However, the discrete log problem has a convenient property, called *self-reducibility*, that allows us not to vary the key. Roughly speaking, if we can find the discrete log of y for certain "easy" y-values, then, given an instance y' that we want to solve, we can always shift it to one of the easy instances by exponentiating y' by known amounts.

random function values. (Hence the name "forking lemma.") We hope that both forgers produce signatures corresponding to the pair (m_j, r_j). If they do not (or if we are unable to proceed because no jth hash function query is made), then we start over again. If they do output signatures (h_1, s_1) and (h_2, s_2), then almost certainly $h_1 \neq h_2$. As before, once we have two congruences $k \equiv s_1 - h_1 x \equiv s_2 - h_2 x \pmod{q}$ with $h_1 \neq h_2$, we immediately find x.

We need to know that there is a non-negligible probability that this will all work as hoped. Let S denote the set of all possible sequences of random bits and random function values during the course of the above procedure, and let $J = \{1, 2, \ldots, q_h\}$. For each $j \in J$ let ε_j denote the probability that a sequence $s \in S$ leads to a forgery of the jth message. By assumption, $\sum_{j \in J} \varepsilon_j = \varepsilon$, where ε denotes the forger's probability of success.

We now use the "splitting lemma." Let A be the set of possible sequences of random bits and random function values that take the forgers up to the point where they ask for $H(m_j, r_j)$; and let B be the set of possible random bits and random function values after that. Suppose that there are a elements in A and b elements in B. Then $S = A \times B$, and $\#S = ab$. For $\varepsilon_j ab$ values of $s \in S$ the forger produces a valid signature for (m_j, r_j). Applying the "splitting lemma," we can say that there are at least $\varepsilon_j a/2$ elements of A that have the following property: the remaining part of the forgery algorithm (starting with the jth hash query) has probability at least $\varepsilon_j/2$ of leading to a signature for (m_j, r_j). For each such element of A the probability that both copies of the forger lead to signatures for (m_j, r_j) is at least $(\varepsilon_j/2)^2$. In summary, the probability that an element of $A \times (B \times B)$ will lead to two different signatures for (m_j, r_j) is at least $(\varepsilon_j/2)^3$.

Since j is chosen uniformly at random from J, it follows that the probability that the above procedure leads to two signatures of the same message is at least $(1/8q_h) \sum_{j \in J} \varepsilon_j^3$. The minimum of the sum $\sum \varepsilon_j^3$ subject to the condition that $\sum \varepsilon_j = \varepsilon$ is achieved when all the ε_j are equal, that is, when $\varepsilon_j = \varepsilon/q_h$ for all j.[15] Thus, the probability of success in one iteration of the procedure is at least $1/8q_h \cdot q_h \cdot (\varepsilon/q_h)^3 = (\varepsilon/2q_h)^3$. If we repeat the procedure k times, the probability that we fail to find the discrete logarithm of x is at most $(1 - (\varepsilon/2q_h)^3)^k$, which approaches zero. This concludes the argument.

5.5. *Tightness*

In order to arrive at a "practice-oriented" interpretation of the above result in the sense of [3], we have to examine the "tightness" of the reductionist security argument. The version of this argument given in the previous subsection is not tight at all. Namely, let t denote the running time (number of steps) of the forger program, and let T be a lower bound on the amount of time we believe it takes to solve the discrete logarithm problem in the group G. If we have to run the forger program k times in order to find (with high probability) the discrete logarithm, then we set $T \approx kt$ to get an estimate for the

[15] It is an easy exercise (using partial derivatives) to show that for any $\ell \neq 1$ the sum $x_1^\ell + x_2^\ell + \cdots + x_{q_h-1}^\ell + (\varepsilon - x_1 - x_2 - \cdots - x_{q_h-1})^\ell$ (with non-negative terms $x_1, x_2, \ldots, x_{q_h-1}, \varepsilon - x_1 - x_2 - \cdots - x_{q_h-1}$) reaches its minimum when all the terms are equal.

minimum time t that the forger takes. Since $(1-(\varepsilon/2q_h)^3)^k$ is small for $k = O(q_h^3/\varepsilon^3)$, it follows that we should set $T \approx tq_h^3/\varepsilon^3$.

For simplicity let us suppose that ε is not very small, for example, $\varepsilon > 0.1$ (in other words, the forger has at least a 10% chance of success). In that case we can neglect the ε^{-3} term (that is, set $\varepsilon = 1$) when making a rough estimate of the magnitude of t.

Now q_h could be of the same magnitude as t; there is no justification for assuming that the number of hash queries the forger makes is bounded by anything other than the forger's running time. Setting $q_h = t$ and $\varepsilon = 1$, we get $T \approx t^4$. The practical consequence is that to get a guarantee of 80 bits of security, we would have to choose q and p in Schnorr's signature scheme large enough so that the discrete log problem in G requires time roughly 2^{320}.

In [50] Pointcheval and Stern give a tighter reduction. We state their result when $q_h = t$ and $\varepsilon = 1$, so that we can compare with the conclusion in the previous paragraph. In that case they show that $T < 2^{17}t^2$; that is, to get 80 bits of security p and q must be chosen so that $T \approx 2^{177}$. According to current estimates of the amount of time required to solve the discrete log problem in a generic group of q elements and in the multiplicative group of the field of p elements using the best available algorithms (these are the Pollard-ρ algorithm and the number field sieve, respectively), we have to choose roughly a 354-bit q and a 7000-bit p. Such sizes would be too inefficient to be used in practice.

On the other hand, if we insist on efficiency and use 1024-bit p and 160-bit q, then what does the Pointcheval–Stern argument give us? With these bitlengths of p and q we have $T \approx 2^{80}$. This means that $t \approx 2^{31.5}$, which is a totally useless level of security. So if we do not want the Schnorr scheme to lose its advantage of short signatures and rapid computation, we probably have to put aside any thought of getting a "provable security" guarantee.

Unfortunately, this type of analysis 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.

Finally, returning to the question of reductions for Schnorr-type signatures, we note that Goh and Jarecki [29] recently proposed a signature scheme for which they gave a tight reduction (in the random oracle model) from the computational Diffie–Hellman problem (see Section 2). Generally speaking, this is not as good as a scheme whose security is closely tied to the discrete log problem, which is a more natural and possibly harder problem. However, Maurer and Wolf [44] have proved that the two problems Diffie–Hellman and discrete log are equivalent in certain groups. If the Goh–Jarecki signature scheme is implemented in such groups, then its security is more tightly bound to the hardness of the discrete logarithm problem than is the Schnorr signature scheme under the Pointcheval–Stern reduction.

6. Is the Random Oracle Assumption Bad for Practice?

In Sections 3–5 we saw reductionist security arguments for important cryptographic systems that used the "random oracle model," that is, treated hash functions as equivalent to random functions. Intuitively, this seems like a reasonable thing to do. After all, in practice a well-constructed hash function would never have any features that distinguish it from random functions that an attacker could exploit.

However, over the years many researchers have expressed doubts about the wisdom of relying on the random oracle model. For example, Canetti et al. [18] constructed examples of cryptographic schemes that are "provably secure" under the random oracle model but are insecure in any real-world implementation. Even though their examples were contrived and unlike any system that would ever be designed in practice, many felt that this work called into question the reliability of security results based on the random oracle assumption and showed a need to develop systems whose security is based on weaker assumptions about the hash function. Thus, the Cramer–Shoup encryption scheme [23] in Section 2 aroused great interest at the time because it was a practical system for which a reductionist security argument could be given under a weaker hash function assumption.

Recently, Bellare et al. [5], [4] obtained a striking result. They constructed an example of a type of cryptographic system that purportedly is practical and realistic and that has a natural and important security property under the random oracle model but not with any concrete hash function. The aim of [5] and [4] was to "bring concerns raised by previous work closer to practice" and thereby show that in real-world cryptography it might be wise to replace cryptosystems whose reductionist security depends on the random oracle assumption by those whose security argument uses a weaker hash function model. In this section we look at the construction in [5] and [4] and explain why we believe that the papers support a conclusion that is exactly the opposite of that of the authors.

The setting in [5] and [4] is a "hybrid" system; this means that an asymmetric (public-key) encryption scheme is used to establish a common key for a certain symmetric (private-key) encryption scheme, after which messages can be sent back and forth efficiently using the symmetric system.

Hybrid systems are important in real-world cryptography. In fact, most electronic commerce and other secure Internet communications use such a system. For example, an RSA-based protocol might establish a "session key"—perhaps a 128-bit random integer—for a buyer and merchant, after which a credit card number and other sensitive information are transmitted quickly and securely using a symmetric encryption method (such as RC4) with the session key.

It is important to note that in practice the symmetric and asymmetric systems must be constructed independently of one another. In Remark 2.1 of [4] it is emphasized that both the keys and the hash function that are used in the symmetric system must have no connection with any keys or hash functions used in the public-key system; otherwise, a symmetric and an asymmetric system might be insecure together even if they are each secure in isolation. This observation generalizes to a fundamental principle of sound cryptographic practice for a hybrid system: *None of the parameters and none of the ingredients in the construction of one of the two systems should incorporate elements of the other system.*

However, it turns out that the proof of the main results in [5] and [4] depends in an essential way on a violation of this principle. Namely, inside the private-key encryption algorithm is a step involving verification of a valid key and valid ciphertext for the public-key system (see Figure 3 of [4]). That is, the argument in [4] fails completely if the above principle of sound cryptographic practice is observed. This is clear from Remark 4.4 in [4], where the authors explain the central role played by the public-key verification steps in their private-key encryption.

Thus, one way to interpret their result is that it serves merely as a reminder of the importance of strictly observing the above principle of independence of the two parts of a hybrid system. However, we believe that there is a much more interesting and valuable conclusion to be drawn. The inability of the authors of [4] to obtain their results without using a construction that violates standard cryptographic practice could be interpreted as evidence in support of the random oracle model. Our reasoning here is analogous to what one does in evaluating the real-world intractability of a mathematical problem such as integer factorization or the elliptic curve discrete logarithm problem (ECDLP). If the top experts in algorithmic number theory at present can factor at most a 576-bit RSA modulus [54], then perhaps we can trust a 1024-bit modulus. If the best implementers of elliptic curve discrete logarithm algorithms have been able to attack at most a 109-bit ECDLP [19], then perhaps we can have confidence in a 163-bit group size. By the same token, if one of the world's leading specialists in provable security (and coauthor of the first systematic study of the random oracle model [6]) puts forth his best effort to undermine the validity for practical cryptography of the random oracle assumption, and if the flawed construction in [4] is the best he can do, then perhaps there is more reason than ever to have confidence in the random oracle model.

What about other papers that call the random oracle model into question? In all cases the constructions are at least as far removed from real-world cryptography as the one in [5] and [4]. We briefly discuss a recent example of this type of work that is concerned with signatures rather than encryption. In [35] and [36] Goldwasser and Tauman claim to have found a difficulty with Pointcheval and Stern's [49] use of the random oracle assumption to show security of signature schemes constructed by the method of Fiat and Shamir [26]. That is, suppose that we have an $(\alpha; \beta; \gamma)$ identification protocol. This means that Alice proves her identity to Bob by sending him a message α, then receiving from him a random sequence β, and finally responding with a sequence γ that Bob is convinced only Alice could have sent. In [26] an $(\alpha; \beta; \gamma)$ identification scheme is converted to an $(\alpha; H; \gamma)$ signature scheme (where H is a hash function) by replacing Bob's random β with the value $H(\alpha, m)$ (where m is the message). In [49] it is shown that if the identification protocol is secure in a strong sense, then the corresponding signature scheme is secure against chosen-message attack under the random oracle model.

In [35] and [36] Goldwasser and Tauman show that a certain modification of the $(\alpha; \beta; \gamma)$ protocol that has no effect on the identification procedure leads to an $(\alpha; H; \gamma)$ signature scheme that is still provably secure under the random oracle assumption but is completely insecure with any concrete hash function H. They modify the identification scheme by stipulating that Bob accept Alice's identity not only if she is able to supply a γ that satisfies him, but also if she is able to guess the value of β when she sends him α. Since only Bob would know this information at the α-stage of the protocol, this basically

means that she is allowed to prove either that she is Alice or that she is Bob, which one assumes she isn't.[16] So this modification of the identification procedure is innocuous.

However, the corresponding signature scheme is useless, because Bob is now told to accept the signature if Alice can give the hash value $H(\alpha, m)$, and this anyone can do, since H is computed by a publicly known algorithm. The main point in [35] and [36] is that the security proof in [49] still goes through, because in the random oracle model the value $H(\alpha, m)$ is unknown until the hash query is made during the β-stage of the procedure. According to [35] and [36], this shows that something is wrong with the random oracle model.

However, once again the random oracle assumption has led us astray only because a set-up has been chosen that would never arise in practice. Even if the original $(\alpha; \beta; \gamma)$ protocol had the peculiar feature that Bob is instructed to accept a correct guess of β in lieu of a correct γ, that feature would never be carried over to the signature scheme. So the difficulty in [35] and [36] would never arise in real-world cryptography. Like [5], [4], and [18], the work by Goldwasser and Tauman seems to be another case where leading experts dedicate considerable energy in an attempt to refute the validity of the random oracle model, but can only come up with a contrived construction that has no plausible connection with actual cryptographic practice. Our confidence in the random oracle assumption is unshaken.

7. Conclusion

Here is a summary of the main conclusions of the previous sections:

- (Section 3) There is no need for the PSS or Katz–Wang versions of RSA; one might as well use just the basic "hash and exponentiate" signature scheme (with a full-domain hash function).
- (Section 4) Currently the optimal RSA-type encryption scheme is the Boneh–Rabin "simplified OAEP."
- (Section 5) Even though the reductionist security results for the Schnorr signature scheme are quite weak from the standpoint of practical guarantees, it is nevertheless surprising that the opponents of DSA in 1992 found only very minor objections to DSA and failed to notice that the modifications of the Schnorr scheme used to get DSA had caused all of the reductionist security to disappear.
- (Section 6) Some recent work that claims to give evidence against the random oracle model actually is inadvertently providing evidence in support of that model.

Finally, we end with some informal comments.

8. An Art or a Science?

In his useful and wonderfully written survey [3], Bellare draws a sharp distinction between two phases in the development of a cryptographic system: the design and study

[16] The Alice of cryptographic fame is neither hermaphroditic nor transgendered.

of the underlying mathematical one-way function (what he calls the "atomic primitive") and the design and study of secure methods (called "protocols") of using such a primitive to achieve specific objectives. He argues that the former is an "art" because intuition and experience play a large role, and the choice between two primitives is ultimately a judgment call. In contrast, according to Bellare, the selection and analysis of protocols can be a "science"—it can almost be mechanized—if provable security techniques are used. He writes:

> . . . the design (or discovery) of good atomic primitives is more an art than a science. On the other hand, I'd like to claim that the design of protocols can be made a science.

In our opinion, this is a spurious distinction: the protocol stage is as much an art as the atomic-primitive stage. The history of the search for "provable" security is full of zigzags, misunderstandings, disagreements, reinterpretations, and subjective judgments. For example, all of our four assertions in the previous section are highly controversial, and can neither be proved nor disproved.

Later in the same article, Bellare makes a comment about terminology that we found helpful:

> . . . what is probably the central step is providing a model and definition, which does not involve proving anything. And one does not "prove a scheme secure": one provides a reduction of the security of the scheme to the security of some underlying atomic primitive [i.e., to the hardness of an underlying mathematical problem]. For that reason, I sometimes use the term "reductionist security" to refer to this genre of work.

We have taken his suggestion and used the term "reductionist security" instead of "provable security."

There are two unfortunate connotations of "proof" that come from mathematics and make the word inappropriate in discussions of the security of cryptographic systems. The first is the notion of 100% certainty. Most people not working in a given specialty regard a "theorem" that is "proved" as something that they should accept without question. The second connotation is of an intricate, highly technical sequence of steps. From a psychological and sociological point of view, a "proof of a theorem" is an intimidating notion: it is something that no one outside an elite of narrow specialists is likely to understand in detail or raise doubts about. That is, a "proof" is something that a non-specialist does not really expect to have to read and think about.

The word "argument," which we prefer here, has very different connotations. An "argument" is something that should be broadly accessible, and even a reasonably convincing argument is not assumed to be 100% definitive. In contrast to a "proof of a theorem," an "argument supporting a claim" suggests something that any well-educated person can try to understand and perhaps question.

Regrettably, many "provable security" papers seem to have been written to meet the goal of semantic security against comprehension by anyone outside the field. A syntactically scrambled informal argument (see, for example, [57]) is followed by a formalistic proof that is so turgid that other specialists do not even read it. As a result,

proof-checking has been a largely unmet security objective, leaving the papers vulnerable to attack. Indeed, Stern [58] has proposed adding a validation step to any security "proof":

> Also, the fact that proofs themselves need time to be validated through public discussion was somehow overlooked.

Unfortunately, this validation step will be hard to implement if the public finds the purported proof to be completely opaque.

Theoreticians who study the security of cryptographic systems should not try to emulate the practices of the most arcane branches of mathematics and science. Mathematicians who study p-adic differential equations, physicists who work on quantum chromodynamics, and chemists who investigate paramagnetic spin–orbit interactions do not seem bothered that their work is inaccessible to everyone outside a tiny circle of fellow specialists. This is to be expected, since their results and methods are intrinsically highly technical and out of reach to anyone who is not totally immersed in the narrow subfield. Moreover, only a negligible proportion of the world's people—somewhere between 2^{-25} and 2^{-30}—have any interest in what they are doing; the rest of us do not care one iota about any of it.

Cryptography is different. A lot of people in industry, government, and academia would truly like to understand to what extent they can have confidence in the systems used to protect, encrypt, and authenticate data.

The major theoretical advances—such as probabilistic encryption, the first good definition of secure digital signatures, the random oracle model, and the idea of public-key cryptography itself—are simple, natural, and easy to understand, or at least become so with the passage of time. In retrospect they look inevitable and perhaps even "obvious." At the time, of course, they were not at all obvious. The fact that these fundamental concepts seem natural to us now does not diminish our appreciation of their importance or our high esteem for the researchers who first developed these ideas.

This brings us to another way in which theoretical cryptography is more an art than a science. Its fruits, and even its inner workings, should be accessible to a broad public. One can say that something "looks easy" without meaning any disrespect. Top-notch ballet dancers make it look easy, as if anyone could do it; but the audience knows that their achievement is possible only through great talent and hard work. By the same token, researchers in "provable security" should strip away unnecessary formalism, jargon, and mathematical terminology from their arguments and strive to make their work "look easy." If they do so, their influence on real-world cryptography will undoubtedly become much greater than it is today.

Acknowledgments

We thank Steven Galbraith, Shafi Goldwasser, Ann Hibner Koblitz, Kenny Paterson, Berkant Ustaoglu, and the two anonymous referees for their valuable comments on earlier drafts of the paper.

References

[1] D. Agrawal, B. Archambeault, J. Rao and P. Rohatgi, The EM side-channel(s), *Cryptographic Hardware and Embedded Systems – CHES* 2002, LNCS 2523, Springer-Verlag, Berlin, 2002, pp. 29–45.

[2] M. Agrawal, N. Kayal and N. Saxena, PRIMES is in P, *Ann. of Math.*, **160** (2004), 781–793.

[3] M. Bellare, Practice-oriented provable-security, *Proc. First International Workshop on Information Security* (*ISW* '97), LNCS 1396, Springer-Verlag, Berlin, 1998, pp. 221–231.

[4] M. Bellare, A. Boldyreva and A. Palacio, An uninstantiable random-oracle-model scheme for a hybrid-encryption problem, Cryptology ePrint Archive, Report 2003/077, 2004.

[5] M. Bellare, A. Boldyreva and A. Palacio, An uninstantiable random-oracle-model scheme for a hybrid-encryption problem, *Advances in Cryptology – Eurocrypt* 2004, LNCS 3027, Springer-Verlag, Berlin, 2004, pp. 171–188.

[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, New York, 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, Berlin, 1994, pp. 92–111.

[8] M. Bellare and P. Rogaway, The exact security of digital signatures—how to sign with RSA and Rabin, *Advances in Cryptology – Eurocrypt* '96, LNCS 1070, Springer-Verlag, Berlin, 1996, pp. 399–416.

[9] D. Bernstein, Proving tight security for standard Rabin–Williams signatures, Preprint, 2003.

[10] D. Bleichenbacher, A chosen ciphertext attack against protocols based on the RSA encryption standard PKCS #1, *Advances in Cryptology – Crypto* '98, LNCS 1462, Springer-Verlag, Berlin, 1998, pp. 1–12.

[11] D. Boneh, The decision Diffie–Hellman problem, *Proc. Third Algorithmic Number Theory Symp.*, LNCS 1423, Springer-Verlag, Berlin, 1998, pp. 48–63.

[12] D. Boneh, Simplified OAEP for the RSA and Rabin functions, *Advances in Cryptology – Crypto* 2001, LNCS 2139, Springer-Verlag, Berlin, 2001, pp. 275–291.

[13] D. Boneh, R. DeMillo and R. Lipton, On the importance of checking cryptographic protocols for faults, *Advances in Cryptology – Eurocrypt* '97, LNCS 1233, Springer-Verlag, Berlin, 1997, pp. 37–51.

[14] D. Boneh and R. Lipton, Algorithms for black-box fields and their application to cryptography, *Advances in Cryptology – Crypto* '96, LNCS 1109, Springer-Verlag, Berlin, 1996, pp. 283–297.

[15] D. Boneh, B. Lynn and H. Shacham, Short signatures from the Weil pairing, *J. Cryptology*, **17** (2004), 297–319.

[16] D. Boneh and R. Venkatesan, Breaking RSA may not be equivalent to factoring, *Advances in Cryptology – Eurocrypt* '98, LNCS 1233, Springer-Verlag, Berlin, 1998, pp. 59–71.

[17] D. Branstad and M. Smid, Responses to comments on the NIST proposed digital signature standard, *Advances in Cryptology – Crypto* '92, LNCS 740, Springer-Verlag, Berlin, 1993, pp. 76–88.

[18] R. Canetti, O. Goldreich and S. Halevi, The random oracle model revisited, *Proc.* 30*th Annual Symp. Theory of Computing*, ACM, New York, 1998, pp. 209–218.

[19] Certicom Corp., Certicom announces elliptic curve cryptography challenge winner, Press Release, 27 April 2004.

[20] D. Coppersmith, Finding a small root of a univariate modular equation, *Advances in Cryptology – Eurocrypt* '96, LNCS 1070. Springer-Verlag, Berlin, 1996, pp. 155–165.

[21] J.-S. Coron, On the exact security of full domain hash, *Advances in Cryptology – Crypto* 2000, LNCS 1880, Springer-Verlag, Berlin, 2000, pp. 229–235.

[22] J.-S. Coron, Optimal security proofs for PSS and other signature schemes, *Advances in Cryptology – Eurocrypt* 2002, LNCS 2332, Springer-Verlag, Berlin, 2002, pp. 272–287.

[23] R. Cramer and V. Shoup, A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack, *Advances in Cryptology – Crypto* '98, LNCS 1462, Springer-Verlag, Berlin, 1998, pp. 13–25.

[24] W. Diffie and M. Hellman, New directions in cryptography, *IEEE Trans. Inform. Theory*, **IT-22** (1976), 644–654.

[25] T. ElGamal, A public key cryptosystem and a signature scheme based on discrete logarithms, *IEEE Trans. Inform. Theory*, **IT-31** (1985), 469–472.

[26] A. Fiat and A. Shamir, How to prove yourself: practical solutions to identification and signature problems, *Advances in Cryptology – Crypto* '86, LNCS 263, Springer-Verlag, Berlin, 1987, pp. 186–194.

[27] E. Fujisaki, T. Okamoto, D. Pointcheval and J. Stern, RSA-OAEP is secure under the RSA assumption, *Advances in Cryptology – Crypto* 2001, LNCS 2139, Springer-Verlag, Berlin, 2001, pp. 260–274.
[28] M. Garey and D. Johnson, *Computers and Intractability*: *A Guide to the Theory of NP-Completeness*, Freeman, San Franicsco, CA, 1979.
[29] E. Goh and S. Jarecki, A signature scheme as secure as the Diffie–Hellman problem, *Advances in Cryptology – Eurocrypt* 2003, LNCS 2656, Springer-Verlag, Berlin, 2003, pp. 401-415.
[30] O. Goldreich, *Foundations of Cryptography*, Vol. 2, Cambridge University Press, Cambridge, 2004.
[31] S. Goldwasser and S. Micali, Probabilistic encryption and how to play mental poker keeping secret all partial information, *Proc*. 14*th Annual Symp*. *Theory of Computing*, ACM, New York, 1982, pp. 365–377.
[32] S. Goldwasser and S. Micali, Probabilistic encryption, *J. Comput*. *System Sci*., **28** (1984), 270–299.
[33] S. Goldwasser, S. Micali and R. Rivest, A "paradoxical" solution to the signature problem, *Proc*. 25*th Annual Symp*. *Foundations of Comput*. *Science*, 1984, pp. 441–448.
[34] S. Goldwasser, S. Micali and R. Rivest, A digital signature scheme secure against adaptive chosen-message attacks, *SIAM J. Comput*., **17** (1988), 281–308.
[35] S. Goldwasser and Y. Tauman, On the (in)security of the Fiat–Shamir paradigm, *Proc*. 44*th Annual Symp*. *Foundations of Comput*. *Science*, 2003, pp. 102–113.
[36] S. Goldwasser and Y. Tauman, On the (in)security of the Fiat–Shamir paradigm, Cryptology ePrint Archive, Report 2003/034, 2003.
[37] A. Joux and K. Nguyen, Separating Decision Diffie–Hellman from Computational Diffie–Hellman in cryptographic groups, *J. Cryptology*, **16** (2003), 239–247.
[38] J. Katz and N. Wang, Efficiency improvements for signature schemes with tight security reductions, *Proc*. 10*th ACM Conf. Computer and Communications Security*, 2003, pp. 155–164.
[39] N. Koblitz, *A Course in Number Theory and Cryptography*, Springer-Verlag, Berlin, 1987.
[40] P. Kocher, Timing attacks on implementations of Diffie–Hellman, RSA, DSS, and other systems, *Advances in Cryptology – Crypto* '96, LNCS 1109, Springer-Verlag, Berlin, 1996, pp. 104–113.
[41] P. Kocher, J. Jaffe and B. Jun, Differential power analysis, *Advances in Cryptology – Crypto* '99, LNCS 1666, Springer-Verlag, Berlin, 1999, pp. 388–397.
[42] 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, Berlin, 2001, pp. 230–238.
[43] U. Maurer, Towards the equivalence of breaking the Diffie–Hellman protocol and computing discrete logarithms, *Advances in Cryptology – Crypto* '94, LNCS 839, Springer-Verlag, Berlin, 1994, pp. 271–281.
[44] U. Maurer and S. Wolf, The relationship between breaking the Diffie–Hellman protocol and computing discrete logarithms, *SIAM J. Comput*., **28**(5) (1999), 1689–1731.
[45] A. J. Menezes, *Elliptic Curve Public Key Cryptosystems*, Kluwer, Dordrecht, 1993.
[46] S. Micali, C. Rackoff and B. Sloan, The notion of security for probabilistic cryptosystems, *SIAM J. Comput*., **17** (1988), 412–426.
[47] A. Miyaji, M. Nakabayashi and S. Takano, New explicit conditions of elliptic curve traces for FR-reduction, *IEICE – Trans*. *Fund*. *Electron*., *Commun*. *Comput*. *Sci*., **E84-A**(5) (2001), 1234–1243.
[48] M. Naor and M. Yung, Public-key cryptosystems provably secure against chosen ciphertext attacks, *Proc*. 22*nd Annual Symp*. *Theory of Computing*, ACM, New York, 1990, pp. 427–437.
[49] D. Pointcheval and J. Stern, Security proofs for signature schemes, *Advances in Cryptology – Eurocrypt* '96, LNCS 1070, Springer-Verlag, Berlin, 1996, pp. 387–398.
[50] D. Pointcheval and J. Stern, Security arguments for digital signatures and blind signatures, *J. Cryptology*, **13** (2000), 361–396.
[51] M. Rabin, Digitalized signatures and public-key functions as intractable as factorization, Technical Report LCS/TR-212, MIT Lab. for Computer Science, 1979.
[52] C. Rackoff and D. Simon, Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack, *Advances in Cryptology – Crypto* '91, LNCS 576, Springer-Verlag, Berlin, 1992, pp. 433–444.
[53] R. Rivest, A. Shamir and L. Adleman, A method for obtaining digital signatures and public key cryptosystems, *Commun*. *ACM*, **21**(2) (1978), 120–126.
[54] RSA Security Inc., Mathematicians from around the world collaborate to solve latest RSA factoring challenge, Press Release, 27 April 2004.
[55] C. P. Schnorr, Efficient signature generation for smart cards, *J. Cryptology*, **4** (1991), 161–174.

[56] V. Shoup, Using hash functions as a hedge against chosen ciphertext attack, *Advances in Cryptology – Eurocrypt* 2000, LNCS 1807, Springer-Verlag, Berlin, 2000, pp. 275–288.
[57] V. Shoup, OAEP reconsidered, *Advances in Cryptology – Crypto* 2001, LNCS 2139, Springer-Verlag, Berlin, 2001, pp. 239–259.
[58] J. Stern, Why provable security matters, *Advances in Cryptology – Eurocrypt* 2003, LNCS 2656, Springer-Verlag, Berlin, 2003, pp. 449–461.
[59] J. Stern, D. Pointcheval, J. Malone-Lee and N. Smart, Flaws in applying proof methodologies to signature schemes, *Advances in Cryptology – Crypto* 2002, LNCS 2442, Springer-Verlag, Berlin, 2002, pp. 93–110.
[60] R. Taylor and A. Wiles, Ring-theoretic properties of certain Hecke algebras, *Ann. of Math.*, **141** (1995), 553–572.
[61] Y. Watanabe, J. Shikata and H. Imai, Equivalence between semantic security and indistinguishability against chosen ciphertext attacks, *Public Key Cryptography – PKC* 2003, LNCS 2567, Springer-Verlag, Berlin, 2003, pp. 71–84.
[62] A. Wiles, Modular elliptic curves and Fermat's Last Theorem, *Ann. of Math.*, **141** (1995), 443–551.
[63] H. Williams, A modification of the RSA public-key encryption procedure, *IEEE Trans. Inform. Theory*, **IT-26** (1980), 726–729.