OAEP Reconsidered*

Victor Shoup
IBM Zurich Research Lab, Säumerstr. 4, 8803 Rüschlikon, Switzerland
sho@zurich.ibm.com

November 23, 2000

Abstract

The OAEP encryption scheme was introduced by Bellare and Rogaway at Eurocrypt '94, and is widely believed to be secure against adaptive chosen ciphertext attack. The main justification for this belief is a proof of security in the random oracle model.

This paper shows conclusively that this justification is invalid. First, it observes that there appears to be a non-trivial gap in the proof. Second, it proves a theorem that essentially says that this gap cannot be filled using standard proof techniques of the type used in Bellare and Rogaway's paper, and elsewhere in the cryptographic literature.

It should be stressed that these results do not imply that RSA-OAEP in insecure. They simply undermine the justification that no attacks are possible in general.

In fact, we make the observation that RSA-OAEP with encryption exponent 3 actually is provably secure in the random oracle model, but the argument makes use of special properties of the RSA function. However, this should not necessarily be viewed as a good reason to use RSA-OAEP with encryption exponent 3.

The paper also presents a new scheme OAEP+ along with a complete proof of security in the random oracle model. OAEP+ is essentially just as efficient as OAEP.

1 Introduction

It is generally agreed that the "right" definition of security for a public key encryption scheme is *security against adaptive chosen ciphertext attack*, as

*This is a minor revision of the original November 16, 2000 version.

defined in [RS91]. This notion of security is equivalent to other useful notions, such as the notion of *non-malleability*, as defined in [DDN91, DDN00].

[DDN91] proposed a scheme that is provably secure in this sense, based on standard intractability assumptions. While this scheme is useful as a proof of concept, it is quite impractical. [RS91] also propose a scheme that is also provably secure; however, it too is also quite impractical, and moreover, it has special "public key infrastructure" requirements.

In 1993, Bellare and Rogaway proposed a method for converting any trapdoor one-way permutation into an encryption scheme [BR93]. They proved that this scheme is secure against adaptive chosen ciphertext attack in the *random oracle model*, provided the underlying trapdoor one-way permutation scheme is secure.

In the random oracle model, one analyzes the security of the scheme by *pretending* that a cryptographic hash function is really a *random oracle*. Now, a proof of security in the random oracle model does not necessarily imply *anything* about real security (see [CGH98]). Nevertheless, it seems that designing a scheme so that it is provably secure in the random oracle model is a good engineering principle, at least when all known schemes that are provably secure without the random oracle heuristic are too impractical. Subsequent to [BR93], many other papers have proposed and analyzed cryptographic schemes in the random oracle model.

The encryption scheme in [BR93] is very efficient from the point of view of computation time. However, it has a "message expansion rate" that is not as good as some other encryption schemes.

In 1994, Bellare and Rogaway proposed another method for converting any trapdoor one-way permutation into an encryption scheme [BR94]. This scheme goes by the name OAEP. The scheme when instantiated with the RSA trapdoor function [RSA78] goes by the name RSA-OAEP, and is the industry-wide standard for RSA encryption (PKCS#1 version 2, IEEE P1363). It is just as efficient computationally as the scheme in [BR93], but it has a better message expansion rate. With RSA-OAEP, one can encrypt messages whose bit-length is up to just a few hundred bits less than the number of bits in the RSA modulus; moreover, the resulting ciphertext is the same size as the RSA modulus itself.

Besides its efficiency in terms of both time and message expansion, and its compatibility with more traditional implementations of RSA encryption, perhaps one of the reasons that OAEP is so popular is the widespread *belief* that the scheme is provably secure in the random oracle model, provided the underlying one-way trapdoor permutation is secure.

In this paper we argue that this belief is unjustified. Specifically, we

argue that in fact, no *complete* proof of the general OAEP method has ever appeared in the literature. Moreover, we prove that no proof is attainable using standard "black box" reductions (even in the random oracle model). We then present a variation, OAEP+, and a complete proof of security in the random oracle model. OAEP+ is essentially just as efficient as OAEP.

There is one more twist to this story: we observe that RSA-OAEP with encryption exponent 3 actually *is* provably secure in the random oracle model; the proof, of course, is not a "black box" reduction, but exploits special algebraic properties of the RSA function. However, a note of caution is in order. It could be the case that RSA with exponent 3 is easier to invert than large exponent RSA, and so this should not necessarily be viewed as a good reason to use RSA-OAEP with encryption exponent 3. It seems better to recommend the use of OAEP+.

Note that although the precise specification of standards (PKCS#1 version 2, IEEE P1363) differ in a few minor points from the scheme described in [BR94], none of these minor changes affect the arguments we make here.

1.1 A missing proof of security

[BR94] contains a valid proof that OAEP satisfies a certain technical property which they call "plaintext awareness." Let us call this property *PA1*. However, it is claimed *without proof* that PA1 implies security against chosen ciphertext attack and non-malleability. Moreover, it is not even clear if the authors mean adaptive chosen ciphertext attack (as in [RS91]) or *indifferent* (a.k.a. *lunchtime*) chosen ciphertext attack (as in [NY90]).

Later, in [BDPR98], a new definition of "plaintext awareness" is given. Let us call this property *PA2*. It is claimed in [BDPR98] that OAEP is "plaintext aware." It is not clear if the authors mean to say that OAEP is PA1 or PA2; in any event, they certainly do not prove anything new about OAEP in [BDPR98]. Furthermore, [BDPR98] contains a valid proof that PA2 implies security against adaptive chosen ciphertext attack.

Notice that nowhere in this chain of reasoning is a proof that OAEP is secure against adaptive chosen ciphertext attack. What is missing is a proof that either OAEP is PA2, or that PA1 implies security against adaptive chosen ciphertext attack.

We should point out, however, that PA1 is trivially seen to imply security against *indifferent* chosen ciphertext attack, and thus OAEP is secure against indifferent chosen ciphertext attack. However, this is a strictly weaker and much less useful notion of security than security against adaptive chosen ciphertext attack.

1.2 Our contributions

In §4, we give a rather informal argument that there is a non-trivial obstruction to obtaining a complete proof of security for OAEP against adaptive chosen ciphertext attack (in the random oracle model).

In §5, we give more formal and compelling evidence for this. Specifically, we prove that if one-way trapdoor permutation schemes with an additional special property exist, then OAEP when instantiated with such a one-way trapdoor permutation scheme is in fact *insecure*. We do not know how to prove the existence of such special one-way trapdoor permutation schemes (assuming, say, that one-way trapdoor permutation schemes exist at all). However, we prove that there exists an oracle, relative to which such special one-way trapdoor permutation schemes exists. It follows that relative to an oracle, the OAEP construction is not secure.

Actually, our proofs imply something slightly stronger: relative to an oracle, OAEP is *malleable* against a *chosen plaintext* attack.

Of course, such relativized results do not necessarily imply anything about the ordinary, unrelativized security of OAEP. But they do imply that standard proof techniques, in which the adversary and the trapdoor function are treated as "black boxes," cannot possibly yield a proof of security, since they would relativize. Certainly, all of the arguments in [BR94] and [BDPR98] involve only "black box" reductions, and so they cannot possibly yield a proof of security.

In §6, we present a new scheme, called OAEP+. This is a variation of OAEP that is essentially just as efficient in all respects as OAEP, but for which we provide a complete, detailed proof of security against adaptive chosen ciphertext attack.

We conclude the paper in §7 on a rather ironic note. After considering other variations of OAEP, we sketch a proof that RSA-OAEP with encryption exponent 3 actually *is* secure in the random oracle model. This fact, however, makes essential use of Coppersmith's algorithm [Cop96] for solving low-degree modular equations. This proof of security does not generalize to large encryption exponents, and in particular, it does not cover the popular encryption exponent $2^{16} + 1$.

Part of the irony of this observation is that Coppersmith viewed his own result as a reason *not* to use exponent 3, while here, it ostensibly gives one reason why one perhaps *should* use exponent 3. But as we have already cautioned, this observation does not necessarily imply that using exponent-3 RSA-OAEP is a good idea.

Let us be clear about the implications of our results. They do not imply

a specific attack on PKCS #1 version 2. They only imply that the main *justification* for the *belief* that OAEP (and hence PKCS #1 version 2) is resistant against adaptive chosen ciphertext attack is *invalid*. Indeed, as we observe, some *particular* instantiations of OAEP are in fact provably secure.

Before moving ahead, we recall some definitions in §2, and the OAEP scheme itself in §3.

2 Preliminaries

2.1 Security against chosen ciphertext attack

We recall the definition of security against adaptive chosen ciphertext attack.

We begin by describing the attack scenario.

Stage 1 The key generation algorithm is run, generating the public key and private key for the cryptosystem. The adversary, of course, obtains the public key, but not the private key.

Stage 2 The adversary makes a series of arbitrary queries to a *decryption oracle*. Each query is a ciphertext y that is decrypted by the decryption oracle, making use of the private key of the cryptosystem. The resulting decryption is given to the adversary. The adversary is free to construct the ciphertexts in an arbitrary way—it is certainly *not* required to compute them using the encryption algorithm.

Stage 3 The adversary prepares two messages x_0, x_1, and gives these to an *encryption oracle*. The encryption oracle chooses $b \in \{0, 1\}$ at random, encrypts x_b, and gives the resulting "target" ciphertext y' to the adversary. The adversary is free to choose x_0 and x_1 in an arbitrary way, except that if message lengths are not fixed by the cryptosystem, then these two messages must nevertheless be of the same length.

Stage 4 The adversary continues to submit ciphertexts y to the decryption oracle, subject only to the restriction that $y \neq y'$.

Stage 5 The adversary outputs $b' \in \{0, 1\}$, representing its "guess" of b.

That completes the description of the attack scenario.

The adversary's *advantage* in this attack scenario is defined to be $|\Pr[b' = b] - 1/2|$.

A cryptosystem is defined to be *secure against adaptive chosen ciphertext attack* if for any efficient adversary, its advantage is negligible.

Of course, this is a complexity-theoretic definition, and the above description suppresses many details, e.g., there is an implicit security parameter which tends to infinity, and the terms "efficient" and "negligible" are technical terms, defined in the usual way. One can work in a *uniform* (i.e., Turing machines) or a *non-uniform* model (i.e., circuits) of computation. This distinction will not affect any results in this paper.

The definition of security we have presented here is from [RS91]. It is called *IND-CCA2* in [BDPR98]. It is known to be equivalent to other notions, such as non-malleability [DDN91, BDPR98, DDN00], which is called *NM-CCA2* in [BDPR98].

It is fairly well understood and accepted that this notion of security is the "right" one, in the sense that a general-purpose cryptosystem that is to be deployed in a wide range of applications should satisfy this property. Indeed, with this property, one can typically establish the security of larger systems that use such a cryptosystem as a component.

There are other, weaker notions of security against chosen ciphertext attack. For example, [NY90] define a notion that is sometimes called *security against indifferent chosen ciphertext attack*, or *security against lunchtime attack*. This definition of security is exactly the same as the one above, except that Stage 4 is omitted—that is, the adversary does not have access to the decryption oracle after it obtains the target ciphertext. While this notion of security may seem natural, it is actually not sufficient in many applications. This notion is called *IND-CCA1* in [BDPR98].

2.2 Trapdoor one-way permutations

We recall the notion of a trapdoor one-way permutation scheme. This consists of a probabilistic *generator* algorithm that outputs (descriptions of) two algorithms f and g, such that the function computed by f is a permutation on the set of k-bit strings, and the function computed by g is its inverse. Of course, this generator takes as input a security parameter; the parameter k, as well as the running times of f and g, should be bounded by a polynomial in this security parameter.

An attack against a trapdoor one-way permutation scheme proceeds as follows. First the generator is run, yielding f and g. The adversary is given f, but not g. Additionally, the adversary is given a random $y \in \{0,1\}^k$. The adversary then computes and outputs a string $w \in \{0,1\}^k$.

The adversary's *success probability* is defined to $\Pr[f(w) = y]$.

The scheme is is defined to be *secure* if for any efficient adversary, its success probability is negligible.

3 OAEP

We now describe the OAEP encryption scheme, as described in §6 of [BR94].

The general scheme makes use of a one-way trapdoor permutation. Let f be the permutation, acting on k-bit strings, and g its inverse. The scheme also makes use of two parameters k_0 and k_1, which should satisfy $k_0+k_1 < k$. It should also be the case that 2^{-k_0} and 2^{-k_1} are negligible quantities. The scheme encrypts messages $x \in \{0,1\}^n$, where $n = k - k_0 - k_1$.

The scheme also makes use of two functions, $G : \{0,1\}^{k_0} \to \{0,1\}^{n+k_1}$, and $H : \{0,1\}^{n+k_1} \to \{0,1\}^{k_0}$. These two functions will be modeled as random oracles in the security analysis.

We describe the key generation, encryption, and decryption algorithms of the scheme.

Key generation This simply runs the generator for the one-way trapdoor permutation scheme, obtaining f and g. The public key is f, and the private key is g.

Encryption Given a plaintext x, the encryption algorithm randomly chooses $r \in \{0,1\}^{k_0}$, and then computes

$$s \in \{0,1\}^{n+k_1},\ t \in \{0,1\}^{k_0},\ w \in \{0,1\}^{k},\ y \in \{0,1\}^{k}$$

as follows:

$$\begin{aligned} s &= G(r) \oplus (x \,\|\, 0^{k_1}), && (1)\\ t &= H(s) \oplus r, && (2)\\ w &= s \,\|\, t, && (3)\\ y &= f(w). && (4) \end{aligned}$$

The ciphertext is y.

Decryption Given a ciphertext y, the decryption algorithm computes

$$w \in \{0,1\}^{k},\ s \in \{0,1\}^{n+k_1},\ t \in \{0,1\}^{k_0},\ r \in \{0,1\}^{k_0},$$
$$z \in \{0,1\}^{n+k_1},\ x \in \{0,1\}^{n},\ c \in \{0,1\}^{k_1}$$

as follows:

$$\begin{aligned} w &= g(y), && (5)\\ s &= w[0 \ldots n + k_1 - 1], && (6)\\ t &= w[n + k_1 \ldots k], && (7)\\ r &= H(s) \oplus t, && (8)\\ z &= G(r) \oplus s, && (9)\\ x &= z[0 \ldots n - 1], && (10)\\ c &= z[n \ldots n + k_1 - 1]. && (11) \end{aligned}$$

If $c = 0^{k_1}$, then the algorithm outputs the cleartext x; otherwise, the algorithm *rejects* the ciphertext, and does not output a cleartext.

4 An informal argument that OAEP cannot be proven secure

We first recall the main ideas of the proof in [BR94] that OAEP is "plaintext aware" in the random oracle model, where G and H are modeled as random oracles.

The argument shows how a simulator that has access to a table of input/output values for the points at which *the adversary queried* G and H can simulate the decryption oracle without knowing the private key.

To make our arguments clearer, we introduce some notation. First, any ciphertext y implicitly defines values w, s, t, r, z, x, c. Let y' denote the target ciphertext, and let $w', s', t', r', z', x', c'$ be the corresponding implicitly defined values for y'. Note that $x' = x_b$ and $c' = 0^{k_1}$.

Let S_G the set of values r at which G was evaluated *by the adversary.* Also, let S_H be the set of values s at which H was evaluated *by the adversary.* Further, let $S'_G = S_G \cup \{r'\}$ and $S'_H = S_H \cup \{s'\}$, where r', s' are the values implicitly defined by y', as described above. We view these sets as growing incrementally as the adversary's attack proceeds—elements are added to these only when a random oracle is queried by the adversary or by the encryption oracle.

Suppose the simulator is given a ciphertext y to decrypt. One can show that if $r \notin S'_G$, then with overwhelming probability the actual decryption algorithm would reject y; this is because in this case, s and $G(r)$ are independent, and so the probability that $c = 0^{k_1}$ is 2^{-k_1}. Moreover, if $s \notin S'_H$, then with overwhelming probability, $r \notin S'_G$; this is because in this case, t

and $H(s)$ are independent, and so r is independent of the adversary's view. From this argument, it follows that the actual decryption algorithm would reject with overwhelming probability, unless $r \in S'_G$ and $s \in S'_H$.

If the decryption oracle simulator (a.k.a., plaintext extractor) has access to S'_G and S'_H, as well as the corresponding outputs of G and H, then it can effectively simulate the decryption without knowing the secret key, as follows. It simply enumerates all $r^* \in S'_G$ and $s^* \in S'_H$, and for each of these computes

$$t^* = H(s^*) \oplus r^*,\ w^* = s^* \,\|\, t^*,\ y^* = f(w^*).$$

If y^* is equal to y, then it computes the corresponding x^* and c^* values, via the equations (10) and (11); if $c^* = 0^{k_1}$, it outputs x^*, and otherwise rejects. If no y^* equals y, then it simply outputs reject.

Given the above arguments, it is easy to see that this simulated decryption oracle behaves exactly like the actual decryption oracle, except with negligible probability. Certainly, if some $y^* = y$, the simulator's response is correct, and if no $y^* = y$, then the above arguments imply that the real decryption oracle would have rejected y with overwhelming probability.

From this, one would like to conclude that the decryption oracle does not help the adversary. But this reasoning is invalid. The problem is, the above simulator should have access to S_G and S_H, along with the corresponding outputs of G and H, and not have access to $r', G(r'), s', H(s')$. The reason is that if the decryption simulator has access to $r', G(r'), s', H(s')$, then the proof that the adversary's advantage in guessing the bit b is closely related to the success probability of computing $f^{-1}(y')$ is doomed to failure: if the simulator needs to "know" r' and s', then it must also "know" w', and so one can not hope use the adversary to compute an unknown f-inverse of y'

On closer observation, it is clear that the decryption simulator does not need to know $s', G(s')$: if $s = s'$, then it must be the case that $t \neq t'$, which implies that $r \neq r'$, and so $c = 0^{k_1}$ with negligible probability. Thus, it is safe to reject all ciphertexts y such that $s = s'$.

If one could make an analogous argument that the decryption simulator does not need to know $r', G(r')$, we would be done. This is unfortunately not the case, as the following example illustrates.

4.1 An example

Suppose that we have an algorithm that actually can invert f. Now of course, in this case, we will not be able to construct a counter-example to the security of OAEP, but we will argue that the proof technique fails. In particular, we show how to build an adversary that uses the f-inverting

algorithm to break the cryptosystem, but it does so in such a way that no simulator given black box access to the adversary and its random oracle queries can use our adversary to compute $f^{-1}(y')$ for a given value of y'.

We now describe adversary. Upon obtaining the target ciphertext y', the adversary computes w' using the algorithm for inverting f, and then extracts the corresponding values s' and t'. The adversary then chooses an arbitrary, non-zero $\Delta \in \{0,1\}^n$, and computes:

$$\begin{aligned} s &= s' \oplus (\Delta \,\|\, 0^{k_1}), \\ t &= t' \oplus H(s') \oplus H(s), \\ w &= s \,\|\, t, \\ y &= f(w). \end{aligned}$$

It is easily verified that y is a valid encryption of $x = x' \oplus \Delta$, and clearly $y \neq y'$. So if the adversary submits y to the decryption oracle, he obtains x, from which he can then easily compute x'.

This adversary clearly breaks the cryptosystem—in fact, its advantage is 1/2. However, note in this attack, the adversary only queries the oracle H at the points s and s'. It never queries the oracle G at all. In fact $r = r'$, and the attack succeeds just where the gap in the proof was identified above.

What information has a simulator learned by interacting with the adversary as a black box? It has only learned s' and s (and hence Δ). So it has learned the first $n + k_1$ bits of the pre-image of y', but the last k_0 remain a complete mystery to the simulator, and in general, they will not be easily computable from the first $n + k_1$ bits. The simulator also has seen the value y submitted to the decryption oracle, but it does not seem likely that this can be used by the simulator to any useful effect.

5 Formal evidence that the OAEP construction is not sound

In this section, we present strong evidence that the OAEP construction is not sound. First, we show that if a special type of one-way trapdoor function f_0 exists, then in fact, we can construct another one-way trapdoor function f such that OAEP using f is *insecure*. Although we do not know how to explicitly construct such a special f_0, we can show that there is an oracle relative to which one exists. Thus, there is an oracle relative to which OAEP is insecure.

Definition 1 *We call a trapdoor one-way permutation scheme* XOR-malleable *if the following property holds. There exists an efficient algorithm U, such that for infinitely many values of the security parameter, $U(f_0, f_0(t), \delta) = f_0(t + \delta)$ with nonnegligible probability. Here, the probability is taken over the random bits of the one-way permutation generator, and random bit strings t and δ in the domain $\{0,1\}^{k_0}$ of f_0.*

Theorem 1 *If there exists an XOR-malleable one-way trapdoor permutation scheme, then there exists a one-way trapdoor permutation scheme such that when OAEP is instantiated with this scheme, the resulting encryption scheme is insecure (in the random oracle model).*

We now prove this theorem, which is based on the example presented in §4.1.

Let f_0 be the given XOR-malleable trapdoor one-way permutation on k_0-bit strings. Let U be the algorithm that computes $f_0(t \oplus \delta)$ from $(f_0, f_0(t), \delta)$. Choose $n > 0$, $k_1 > 0$, and set $k = n + k_0 + k_1$. Let f be the permutation on k-bit strings defined as follows: for $s \in \{0,1\}^{n+k_1}, t \in \{0,1\}^{k_0}$, $f(s \,\|\, t) = s \,\|\, f_0(t)$.

It is clear that f is a one-way trapdoor permutation.

Now consider the OAEP scheme that uses this f as its trapdoor one-way permutation, and uses the parameters k, n, k_0, k_1 for the padding scheme.

Recall our notational conventions: any ciphertext y implicitly defines values w, s, t, r, z, x, c, and the target ciphertext y' implicitly defines $w', s', t', r', z', x', c'$.

We now describe adversary. Upon obtaining the target ciphertext y', the adversary decomposes y' as $y' = s' \,\|\, f_0(t')$. The adversary then chooses an arbitrary, non-zero $\Delta \in \{0,1\}^n$, and computes:

$$\begin{aligned} s &= s' \oplus (\Delta \,\|\, 0^{k_1}), \\ v &= U(f_0, f_0(t'), H(s') \oplus H(s)), \\ y &= s \,\|\, v. \end{aligned}$$

It is easily verified that y is a valid encryption of $x = x' \oplus \Delta$, provided $v = f_0(t' \oplus H(s') \oplus H(s))$, which by our assumption of XOR-malleability occurs with non-negligible probability. So if the adversary submits y to the decryption oracle, he obtains x, from which he can then easily compute x'.

This adversary clearly breaks the cryptosystem. That completes the proof of the theorem.

Of course, one might ask if it is at all reasonable to believe that XOR-malleable trapdoor one-way permutations exist at all. First of all, note that the standard RSA function is a one-way trapdoor permutation that is not XOR-malleable, but is still malleable in a very similar way: given $\alpha = (a^e \bmod N)$ and $(b \bmod N)$, we can compute $((ab)^e \bmod N)$ as $(\alpha \cdot (b^e \bmod N))$. Thus, we can view the RSA function itself as a kind of malleable trapdoor one-way permutation, but where XOR is replaced by multiplication mod N. In fact, one could modify the OAEP scheme so that $t, H(s)$ and r are numbers mod N, and instead of the relation $t = H(s) \oplus r$, we would use the relation $t = H(s) \cdot r \bmod N$. It would seem that if there were a proof of security for OAEP, then it should go through for this variant of OAEP as well. But yet, this variant of OAEP is clearly insecure, even though the underlying trapdoor permutation is presumably secure.

Beyond this, we show the following.

Theorem 2 *There exists an oracle, relative to which XOR-malleable trapdoor one-way permutations exist.*

Theorems 1 and 2 imply the following.

Corollary 1 *There exists an oracle, relative to which the OAEP construction is insecure.*

We now prove Theorem 2.

We first begin by describing a *distribution* of oracles.

Let W be chosen at random from the set of all functions on $\{0,1\}^*$ such that for any $n \geq 0$, W restricted to $\{0,1\}^n$ acts as a permutation on $\{0,1\}^n$.

Let F be a family of permutations, such that for every positive integer k_0, and for every $pk \in \{0,1\}^{k_0}$, F_{pk} is a random permutation on $\{0,1\}^{k_0}$.

Our oracle $\mathcal{O}$ responds to four different types of queries:

$\mathcal{O}_1$: Given $sk \in \{0,1\}^*$, return $pk = W(sk)$.

$\mathcal{O}_2$: Given $pk, \delta \in \{0,1\}^*$ with $|pk| = |\delta|$, return $F_{pk}(\delta)$.

$\mathcal{O}_3$: Given $sk, v \in \{0,1\}^*$ with $|sk| = |v|$, return $F^{-1}_{W(sk)}(v)$.

$\mathcal{O}_4$: Given $pk, v, \delta \in \{0,1\}^*$ with $|pk| = |v| = |\delta|$, return $F_{pk}(F^{-1}_{pk}(v) \oplus \delta)$.

The idea here is that the function W can be used to generate public key/secret key pairs for a *random* trapdoor permutation. If one chooses

secret key sk' at random, then $pk' = W(sk')$ is the corresponding public key. This can be accomplished using the $\mathcal{O}_1$ query.

The $\mathcal{O}_2$ query can be used to compute the permutation in the forward direction using pk'. Using $\mathcal{O}_3$ with the trapdoor sk', one can compute the inverse permutation.

Query $\mathcal{O}_4$ is what makes our trapdoor permutation XOR-malleable.

To make a rigorous and precise proof, we state and prove the following very simple, but useful lemma, which we will also use for some other proofs in the paper.

Lemma 1 *Let E, E', F, and F' be events defined on a probability space such that $\Pr[E \wedge \neg F] = \Pr[E' \wedge \neg F']$ and $\epsilon = \Pr[F] = \Pr[F']$. Then we have*

$$\left|\Pr[E] - \Pr[E']\right| \leq \epsilon.$$

Proof. If $\epsilon = 0$ or $\epsilon = 1$, the lemma is trivially true, so assume $0 < \epsilon < 1$. We have

$$\Pr[E] = \Pr[E|\neg F](1-\epsilon) + \Pr[E|F]\epsilon$$

and

$$\Pr[E'] = \Pr[E'|\neg F'](1-\epsilon) + \Pr[E'|F']\epsilon.$$

Subtracting these two equations, noting that $\Pr[E|\neg F] = \Pr[E'|\neg F']$, and taking absolute values, we have

$$\left|\Pr[E] - \Pr[E']\right| = \epsilon \left|\Pr[E|F] - \Pr[E'|F']\right| \leq \epsilon.$$

□

Theorem 2 will now follow from the following lemma.

Lemma 2 *Any adversary that makes at most m oracle queries, succeeds in inverting a permutation on k_0 bits with probability $O(m^2/2^{k_0})$. Here, the probability is taken over the random choice of the oracle, the random choice of the secret key, the random choice of the element to be inverted, and the random choices made by the adversary.*

We can assume that whenever the adversary makes an $\mathcal{O}_3$ query with a given value of sk, he has previously made an $\mathcal{O}_1$ query with the same value. Any adversary can be modified to conform to this convention, increasing m by a constant factor.

Let $\mathbf{G}_0$ be the original attack game. Let (sk', pk') be secret key/public key of the generated trapdoor permutation. Let $f_0 = F_{sk'}$ denote this permutation, and assume that it is a permutation on $\{0,1\}^{k_0}$. Let $v' \in \{0,1\}^{k_0}$ be the string whose f_0-inverse the adversary is trying to compute, and let t' be this inverse. Let S_0 denote the event that the adversary succeeds.

We consider a modified attack game, $\mathbf{G}_1$, defined as follows. Game $\mathbf{G}_1$ is exactly the same as $\mathbf{G}_0$, except that in game $\mathbf{G}_1$, if the adversary ever inputs sk' to the $\mathcal{O}_1$ oracle, which can be detected by testing if the output equals pk', it politely halts. Conceptually, $\mathbf{G}_0$ and $\mathbf{G}_1$ are games that operate on the same probability space, but the rules of the game are different. Let S_1 be the event that the adversary succeeds in $\mathbf{G}_1$, and let F_0 be the event that in game $\mathbf{G}_0$ (or equivalently, $\mathbf{G}_1$), the adversary inputs sk' to the $\mathcal{O}_1$ oracle. Then Lemma 1 applies with (S_0, S_1, F_0, F_0), and moreover, it is clear that $\Pr[F_0] = O(m/k_0)$. Therefore,

$$|\Pr[S_0] - \Pr[S_1]| = O(m/k_0). \tag{12}$$

By construction, the only information the adversary learns about f_0 in game $\mathbf{G}_1$ is through its initial input v', and calls to the oracles $\mathcal{O}_2$ and $\mathcal{O}_4$.

We now define a game $\mathbf{G}_1'$ that is completely equivalent to game $\mathbf{G}_1$, but formulated in a slightly different way. In this game, we process $\mathcal{O}_1$ and $\mathcal{O}_3$ queries just as in game $\mathbf{G}_1$. Also, we process $\mathcal{O}_2$ and $\mathcal{O}_4$ queries with $pk \neq pk'$ just as in game $\mathbf{G}_1$. However, we process $\mathcal{O}_2$ and $\mathcal{O}_4$ queries with $pk = pk'$ differently.

At the outset of the game, we generate a vector $(v_1, \ldots, v_{2^{k_0}})$ that is a random permutation of $\{0,1\}^{k_0}$. We also generate a sequence $(s_1, \ldots, s_m)$, where each s_i is uniformly and independently drawn from $\{0,1\}^{k_0}$.

Now, we shall also define sequences $(t_0, t_1, t_2, \ldots)$ and $(D_1, D_2, \ldots)$ incrementally as the game proceeds. Each t_i is a bit string of length k_0. Each D_i is a pair (j, δ), where j is an integer and δ is a bit string of length k_0. We will maintain two counters, a and b, and it will always be the case that D_i is defined for $1 \leq i \leq a$, and that t_j is defined for $0 \leq j \leq b$.

Conceptually, the permutation f_0 is implicitly defined by $f_0(t_j \oplus \delta) = v_i$, where $D_i = (j, \delta)$, $1 \leq i \leq a$, and $0 \leq j \leq b$.

At the outset of the game, we set $a = 1$, $b = 1$, $t_0 = 0^{k_0}$, $t_1 = s_1$, and $D_1 = (1, 0^{k_0})$. We also set $v' = v_1$ and $t' = t_1$. We give v' to the adversary—this is the element whose inverse the adversary is supposed to compute. At the end of the game, the adversary succeeds if its output is equal to t'.

Now consider an $\mathcal{O}_2$ query with input (pk', δ). We first test if there is an i with $1 \leq i \leq a$, such that $D_i = (0, \delta)$. If so, we let the oracle output

the corresponding v_i. Otherwise, we test if there exists an i with $1 \leq i \leq a$, $D_i = (j, \tilde{\delta})$, and $1 \leq j \leq b$, such that

$$\delta = t_j \oplus \tilde{\delta}, \tag{13}$$

If so, we let the oracle output the corresponding v_i. Otherwise, we increment a, and set $D_a = (0, \delta)$, and output v_a.

Now consider an $\mathcal{O}_4$ query with input (pk', v, δ).
Case 1. $v = v_i$ for some $1 \leq i \leq a$:

- If $D_i = (0, \tilde{\delta})$ for some $\tilde{\delta}$, then we process this $\mathcal{O}_4$ query just like an $\mathcal{O}_2$ query with input $(pk', \delta \oplus \tilde{\delta})$.
- Otherwise, $D_i = (j, \tilde{\delta})$ for some $1 \leq j \leq b$ and some $\tilde{\delta}$. If there exists an i', with $1 \leq i' \leq a$ such that $D_{i'} = (j, \delta \oplus \tilde{\delta})$, then we output $v_{i'}$.
- Otherwise, we test there exists an i', with $1 \leq i' \leq a$, $D_{i'} = (j', \tilde{\delta}')$, $0 \leq j' \leq b$, and $j' \neq j$, such that

  $$t_j \oplus \tilde{\delta} \oplus \delta = t_{j'} \oplus \tilde{\delta}'. \tag{14}$$

  (Note that we allow $j' = 0$.) If so, we output $v_{i'}$.
- Otherwise, we increment a, set $D_a = (j, \delta \oplus \tilde{\delta})$, and output v_a.

Case 2. $v \neq v_i$ for all $1 \leq i \leq a$:

- Let

  $$T = \{t_j \oplus \tilde{\delta} : D_i = (j, \tilde{\delta}),\ 1 \leq i \leq a\}.$$

  We increment b, and then we define t_b as follows. First, we test if

  $$s_b \in T. \tag{15}$$

  If so, we choose t_b at random from $\{0,1\}^{k_0} \backslash T$; otherwise, we set $t_b = s_b$.
- Next, we increment a. Let a' be such that $v_{a'} = v$. By construction, we have $a' \geq a$. We now swap v_a and v'_a. Next, we define $D_a = (b, 0^{k_0})$, and then perform the actions in case 1.

Let S'_1 be the event that the adversary succeeds in game $\mathbf{G}'_1$. It is straightforward to verify that

$$\Pr[S_1] = \Pr[S'_1]. \tag{16}$$

Now we define a game $\mathbf{G}_2$ that is just like $\mathbf{G}_1'$, except that we simply behave *as if* the tests (13), (14), and (15) *always fail*. Conceptually, we view $\mathbf{G}_1'$ and $\mathbf{G}_2$ as operating on the same probability space; in particular, the vectors $(v_1, \ldots, v_{2^{k_0}})$ and $(s_1, \ldots, s_m)$ are the same in both games. Note that in game $\mathbf{G}_2$ it no longer makes sense to speak of an implicitly defined permutation f_0; however, we can still define the event S_2 that the adversary outputs t'. Let F_2 be the event that in game $\mathbf{G}_2$, one of these tests (13), (14), and (15) passes (even though this is ignored in game $\mathbf{G}_2$). Notice that in game $\mathbf{G}_2$, the values v_i seen by the adversary, and hence the inputs to the oracle, are independent of the values s_j. It follows F_2 is equivalent to the event that one of $O(m^2)$ equations holds, where each equation is of the form $s_j = \delta$, where δ and s_j are independent, or of the form $s_j \oplus s_{j'} = \delta$, where $j \neq j'$, and δ, s_j, and $s_{j'}$ are independent. Each equation is satisfied with probability $1/2^{k_0}$, and hence $\Pr[F_2] = O(m^2/2^{k_0})$. One also sees that $\Pr[S_2 \wedge \neg F_2] = \Pr[S_1' \wedge \neg F_2]$, since both games proceed *identically* up until the first point where F_2 occurs. So we apply Lemma 1 with (S_1', S_2, F_2, F_2), and we obtain

$$|\Pr[S_1'] - \Pr[S_2]| = O(m^2/2^{k_0}). \tag{17}$$

Finally, observe that

$$\Pr[S_2] = O(1/2^{k_0}), \tag{18}$$

since in this game, the value t' is independent of the adversary's view.

So finally, Lemma 2 follows from (12), (16), (17), and (18).

There are a few more details required to finish the proof of Theorem 2. The reason we are not done is that we want to show that there exists a fixed oracle relative to which the implied permutation is one way. These details are straightforward, but slightly tedious.

Let $\{M_i\}_{i \geq 1}$ be a complete enumeration of of probabilistic, polynomial time Turing machines. Fix numbers $0 < \beta < \alpha < 1$. Lemma 2 implies that for each $i \geq 1$, there exists an integer $k(i)$ such that whenever $k_0 \geq k(i)$, machine M_i wins the inversion game on k_0-bit permutations with probability at most $2^{-\alpha k_0}$. Here, the probability space includes the random choice of oracle $\mathcal{O}$. Let c, d be positive numbers whose values will be determined below, and for $i \geq 1$ define $k'(i) = \max\{k(i), \lceil c \log i + d \rceil\}$.

For positive integers i and k_0, let us say that oracle $\mathcal{O}$ is (i, k_0)-*good* if machine M_i wins the inversion game on k_0-bit permutations with probability at most $2^{-\beta k_0}$, where the probability is conditioned on the choice of $\mathcal{O}$; otherwise, we say that $\mathcal{O}$ is (i, k_0)-*bad*. By Markov's inequality, for any

$i \geq 1$ and any $k_0 \geq k'(i)$, the probability that a random $\mathcal{O}$ is (i, k_0)-bad is at most $2^{-\gamma k_0}$, where $\gamma = \alpha - \beta$. Let us say that an oracle $\mathcal{O}$ is *good* if it is (i, k_0)-good for all $i \geq 1$ and $k_0 \geq k'(i)$; otherwise, we say that $\mathcal{O}$ is *bad*. Let p be the probability that a random oracle $\mathcal{O}$ is bad. Then we have:

$$p \leq \sum_{i \geq 1} \sum_{k_0 \geq k'(i)} 2^{-\gamma k_0} \leq (1/(1 - 2^{-\gamma})) \sum_{i \geq 1} 2^{-\gamma(c \log i + d)}.$$

From this, one sees that $p < 1$ for appropriate choices of c and d. Thus, there exists a good oracle $\mathcal{O}$. Such a good oracle then satisfies the conditions of Theorem 2; namely, for all $i \geq 1$, for all $k_0 \geq k'(i)$, the probability that M_i wins the inversion game on k_0-bit permutations is at most $2^{-\beta k_0}$.

That completes the proof of Theorem 2.

Remark. The proof above is quite similar to proofs of lower bounds for the discrete logarithm problem presented in [Sho97] (in particular, Theorem 2 in that paper). There are a few technical differences, and the proof we have presented here is much more complete. We should also point out that Lemma 2 is fairly tight, in the sense that the well-known baby step/giant step attack for the discrete logarithm problem (c.f., §3.6.2 of [MvOV97]) can be easily adapted to inverting XOR-malleable permutations, provided the algorithm U is highly reliable.

6 OAEP+

We now describe the OAEP+ encryption scheme, which is just a slight modification of the OAEP scheme.

The general scheme makes use of a one-way trapdoor permutation. Let f be the permutation, acting on k-bit strings, and g its inverse. The scheme also makes use of two parameters k_0 and k_1, which should satisfy $k_0 + k_1 < k$. It should also be the case that 2^{-k_0} and 2^{-k_1} are negligible quantities. The scheme encrypts messages $x \in \{0,1\}^n$, where $n = k - k_0 - k_1$.

The scheme also makes use of three functions:

$$\begin{aligned} G &: \{0,1\}^{k_0} \to \{0,1\}^n, \\ H' &: \{0,1\}^{n+k_0} \to \{0,1\}^{k_1}, \\ H &: \{0,1\}^{n+k_1} \to \{0,1\}^{k_0}. \end{aligned}$$

These three functions will be modeled as independent random oracles in the security analysis.

We describe the key generation, encryption, and decryption algorithms of the scheme.

Key generation This simply runs the generator for the one-way trapdoor permutation scheme, obtaining f and g. The public key is f, and the private key is g.

Encryption Given a plaintext x, the encryption algorithm randomly chooses $r \in \{0,1\}^{k_0}$, and then computes

$$s \in \{0,1\}^{n+k_1},\ t \in \{0,1\}^{k_0},\ w \in \{0,1\}^{k},\ y \in \{0,1\}^{k}$$

as follows:

$$\begin{aligned} s &= (G(r) \oplus x) \,\|\, H'(r \,\|\, x), && (19)\\ t &= H(s) \oplus r, && (20)\\ w &= s \,\|\, t, && (21)\\ y &= f(w). && (22) \end{aligned}$$

The ciphertext is y.

Decryption Given a ciphertext y, the decryption algorithm computes

$$\begin{gathered} w \in \{0,1\}^{k},\ s \in \{0,1\}^{n+k_1},\ t \in \{0,1\}^{k_0},\\ r \in \{0,1\}^{k_0},\ x \in \{0,1\}^{n},\ c \in \{0,1\}^{k_1} \end{gathered}$$

as follows:

$$\begin{aligned} w &= g(y), && (23)\\ s &= w[0 \ldots n+k_1-1], && (24)\\ t &= w[n+k_1 \ldots k], && (25)\\ r &= H(s) \oplus t, && (26)\\ x &= G(r) \oplus s[0 \ldots n-1], && (27)\\ c &= s[n \ldots n+k_1-1]. && (28) \end{aligned}$$

If $c = H'(r \,\|\, x)$, then the algorithm outputs the cleartext x; otherwise, the algorithm *rejects* the ciphertext, and does not output a cleartext.

Theorem 3 *If the underlying trapdoor one-way permutation scheme is secure, then OAEP+ is secure against adaptive chosen ciphertext attack.*

We start with some notations and conventions.

Let $\mathbf{G}_0$ be the original attack game. Let b and b' be as defined in §2.1, and let S_0 be the event that $b = b'$.

Let q_G, q_H, and $q_{H'}$ bound the number of queries made by the adversary to oracles G, H, and H' respectively, and let q_D bound the number of decryption oracle queries.

We assume without loss of generality that whenever the adversary makes a query of the form $H'(r \,\|\, x)$, for any $r \in \{0,1\}^{k_0}, x \in \{0,1\}^n$, the adversary has previously made the query $G(r)$.

Also, let T_0 bound the total running of the adversary's attack in game $\mathbf{G}_0$—this running time includes that of the adversary's own algorithm, as well as the that of the other system components (key generation, encryption oracle, and decryption oracle). Let T_f denote the time needed to evaluate f.

Finally, let $InvAdv(T)$ be the maximal advantage that any adversary that runs in time T has of inverting the given one-way trapdoor permutation scheme.

We shall show that

$$\begin{aligned} |\Pr[S_0] - 1/2| \quad &\leq \quad InvAdv(O(T_0 + q_G q_H T_f)) + \\ &\quad (q_{H'} + 2q_D)/2^{k_1} + (q_D + 1)q_G/2^{k_0}. \end{aligned} \tag{29}$$

We shall define a sequence $\mathbf{G}_1, \mathbf{G}_2, \ldots$ of modified attack games. In each of these games, there are well-defined quantities b and b'. For any $i \geq 1$, we let S_i be the event that $b = b'$ in game $\mathbf{G}_i$. Also, for $i \geq 1$, let T_i bound the total running of the adversary's attack in game $\mathbf{G}_i$.

Any ciphertext y implicitly defines values w, s, t, r, x, c. Let y' denote the target ciphertext, and let w', s', t', r', x', c' be the corresponding implicitly defined values for y'. Note that $x' = x_b$ and $c' = H(r' \,\|\, x')$.

We define sets S_G and S_H, as in §4, as follows. Let S_G the set of values r at which G was evaluated *by the adversary*. Also, let S_H be the set of values s at which H was evaluated *by the adversary*. We view these sets as growing incrementally as the adversary's attack proceeds—elements are added to these only when a random oracle is queried by the adversary.

Game $\mathbf{G}_1$. Now we modify game $\mathbf{G}_0$ to define a new game $\mathbf{G}_1$. We view $\mathbf{G}_0$ and $\mathbf{G}_1$ as being "driven" from the same underlying probability space—just the rules of how functions on this probability space are computed are different.

We modify the decryption oracle as follows. Given a ciphertext y, the new decryption oracle computes w, s, t, r, x, c as usual. If the old decryption oracle rejects, so does the new one. But the new decryption oracle also rejects if $r \notin S_G$. More precisely, if the new decryption oracle computes r via equation (26), and finds that $r \notin S_G$, then it rejects right away, without ever

evaluating $G(r)$. Thus, in game $\mathbf{G}_1$, the decryption oracle never evaluates G at a point at which the adversary has not already evaluated G.

Let F_1 be the event that a ciphertext is rejected in $\mathbf{G}_1$, but not rejected in $\mathbf{G}_0$. Unless either the adversary or the encryption oracle invoked $H'(r \,\|\, x)$, then this value is independent of all the other values available (directly or indirectly) to the adversary, and hence in game $\mathbf{G}_0$ we will reject with probability $1 - 1/2^{k_1}$. Since $y \neq y'$, either $x \neq x'$ or $r \neq r'$. If $r = r'$, then we must have $x \neq x'$, and so although the encryption oracle has made the query $H'(r' \,\|\, x')$, this will not "help" the adversary. So in any case, the adversary must have explicitly made the query $H'(r \,\|\, x)$ himself if it is to be made at all, and if this occurs, then by our convention, the adversary has already queried $G(r)$.

From the above, it follows that $\Pr[F_1] \leq q_D/2^{k_1}$. Moreover, it is clear by construction that $\Pr[S_0 \wedge \neg F_1] = \Pr[S_1 \wedge \neg F_1]$, since the two games proceed identically until the event F_1 occurs. So applying Lemma 1 with (S_0, S_1, F_1, F_1), we have

$$|\Pr[S_0] - \Pr[S_1]| \leq q_D/2^{k_1}. \tag{30}$$

Game $\mathbf{G}_2$. Now we modify game $\mathbf{G}_1$ to obtain a new game $\mathbf{G}_2$. In this new game, we modify the decryption oracle yet again. Given a ciphertext y, the new decryption oracle computes w, s, t, r, x, c as usual. If the old decryption oracle rejects, so does the new one. But the new rejection oracle also rejects if $s \notin S_H$. More precisely, if the new decryption oracle computes s via equation (24), and finds that $s \notin S_H$, then it rejects right away, without ever evaluating $H(s)$. Thus, in game $\mathbf{G}_2$, the decryption oracle never evaluates G or H at points other than those at which the adversary did.

Let F_2 be the event that a ciphertext is rejected in $\mathbf{G}_2$, but not rejected in $\mathbf{G}_1$. Consider a ciphertext $y \neq y'$.

We consider two cases.

Case 1. $s \notin S_H$, $s = s'$: Now, $s = s'$ and $y \neq y'$ implies $t \neq t'$. Moreover, $s = s'$ and $t \neq t'$ implies that $r \neq r'$. If this ciphertext is rejected in game $\mathbf{G}_2$ but not in $\mathbf{G}_1$, it must be the case that $H'(r' \,\|\, x') = H'(r \,\|\, x)$. The probability that such a collision can be found over the course of the attack is $(q_{H'} + q_D)/2^{k_1}$. Note that r' is fixed by the encryption oracle, and so "birthday attacks" are not possible.

Case 2. $s \notin S_H$, $s \neq s'$: In this case, the oracle H was never evaluated at s by either the adversary, the encryption oracle, or the decryption oracle. Since $t = H(s) \oplus r$, the value r is independent of all values accessible by the adversary, directly or indirectly. It follows that the probability that $r \in S_G$

is at most $q_G/2^{k_0}$. Over the course of the entire attack, these probabilities sum to $q_D q_G/2^{k_0}$.

It follows that $\Pr[F_2] \leq (q_{H'} + q_D)/2^{k_1} + q_D q_G/2^{k_0}$. Moreover, it is clear by construction that $\Pr[S_1 \wedge \neg F_2] = \Pr[S_2 \wedge \neg F_2]$. So applying Lemma 1 with (S_1, S_2, F_2, F_2), we have

$$|\Pr[S_1] - \Pr[S_2]| \leq (q_{H'} + q_D)/2^{k_1} + q_D q_G/2^{k_0}. \tag{31}$$

Game $\mathbf{G}_3$. Now we modify game $\mathbf{G}_2$ to obtain an equivalent game $\mathbf{G}_3$. We modify the decryption oracle so that it does not make use of the trapdoor for f at all. The new decryption oracle simply enumerates all $r^* \in S_G$ and $s^* \in S_H$, and for each of these computes

$$t^* = H(s^*) \oplus r^*,\ w^* = s^* \,\|\, t^*,\ y^* = f(w^*).$$

If y^* is equal to y, then it computes the corresponding x^* and c^* values, via the equations (27) and (28); if $c^* = H'(r^* \,\|\, x^*)$, it outputs x^*, and otherwise rejects. If no y^* equals y, then it simply outputs reject.

It is clear that

$$\Pr[S_3] = \Pr[S_2]. \tag{32}$$

Moreover, by using appropriate data structures,

$$T_3 = O(T_0 + q_G q_H T_f). \tag{33}$$

Game $\mathbf{G}_4$. In game $\mathbf{G}_4$, we modify the *encryption oracle* in game $\mathbf{G}_3$. Instead of computing y' using the encryption algorithm, we simply choose it at random. To avoid confusion, we denote this value in game $\mathbf{G}_4$ as y''.

Now, y'' determines values $w'', s'', t'', r'', x'', c''$ as usual. Let F_3 be the event that the adversary in game $\mathbf{G}_3$ evaluates $G(r')$. Let F_4 be the event that the adversary in game $\mathbf{G}_4$ evaluates $G(r'')$. Now, by inspection, one sees that $\Pr[F_3] = \Pr[F_4]$, and that $\Pr[S_3 \wedge \neg F_3] = \Pr[S_4 \wedge \neg F_4]$. So Lemma 1 applies with (S_3, S_4, F_3, F_4), and we obtain

$$|\Pr[S_3] - \Pr[S_4]| \leq \Pr[F_4].$$

So it remains to bound $\Pr[F_4]$. Let F_4' be the event that the adversary queries $H(s'')$ in game $\mathbf{G}_4$. We have

$$\begin{aligned} \Pr[F_4] &= \Pr[F_4 \wedge F_4'] + \Pr[F_4 | \neg F_4'] \Pr[\neg F_4'] \\ &\leq \mathit{InvAdv}(O(T_0 + q_G q_H T_f)) + q_G/2^{2^{k_0}}. \end{aligned}$$

This follows from the fact that

$$\Pr[F_4 \wedge F_4'] \leq InvAdv(O(T_0 + q_G q_H T_f)),$$

which follows from the observation that if the adversary evaluates G at r'' and H at s'', then we can easily convert the attack into an algorithm that computes $f^{-1}(y'')$ for random y''. The inverting algorithm takes y'' as input, and then runs the adversary against game $\mathbf{G}_4$. Of course, the inverting algorithm does not need random oracles—it just simulates the random oracles' responses in the natural way whenever the adversary makes a query. When the adversary terminates, the inverting algorithm enumerates all $r^* \in S_G$ and $s^* \in S_H$, and for each of these computes

$$t^* = H(s^*) \oplus r^*,\ w^* = s^* \,\|\, t^*,\ y^* = f(w^*).$$

If y^* is equal to y'', then it outputs w^* and terminates.

It is also clear that

$$\Pr[F_4 | \neg F_4'] \leq q_G / 2^{2^{k_0}},$$

since if the adversary never queries H at s'', then r'' is independent of all values accessible to the adversary.

So we conclude that

$$|\Pr[S_3] - \Pr[S_4]| \leq InvAdv(O(T_0 + q_G q_H T_f)) + q_G / 2^{2^{k_0}}. \tag{34}$$

Further, we observe that

$$Pr[S_4] = 1/2. \tag{35}$$

This follows from the fact that in game $\mathbf{G}_4$, all values accessible to the adversary are independent of the hidden bit b, so the probability that the adversary outputs b is $1/2$.

Equations (30), (31), (32), (34), (35) together imply (29).

That completes the proof of Theorem 3.

7 Further Observations

7.1 Other variations of OAEP

Instead of modifying OAEP as we did, one could also modify OAEP so that instead of adding the data-independent redundancy 0^{k_1} in (1), one added the

data-dependent redundancy $H''(x)$, where H'' is a hash function mapping n-bit strings to k_1-bit strings. This variant of OAEP—call it OAEP′—would also be secure against adaptive chosen ciphertext attack, but its security would be quantitatively inferior to that of OAEP+. Indeed, OAEP′ can be attacked using the technique in the proof of Theorem 1, combined with an "birthday attack" that finds collisions in H''.

7.2 But RSA-OAEP with exponent 3 is provably secure

Consider RSA-OAEP. Let N be the modulus and e the encryption exponent. Then this scheme actually *is* secure in the random oracle model, provided $k_0 \leq \log_2 N/e$. This condition is satisfied by typical implementations of RSA-OAEP with $e = 3$.

We sketch very briefly why this is so.

We first remind the reader of the attempted proof of security of OAEP in §4, and we adopt all the notation specified there.

Suppose an adversary submits a ciphertext y to the decryption oracle. We observed in §4 that if the adversary never explicitly queried $H(s)$, then with overwhelming probability, the actual decryption oracle would reject. The only problem was, we could not always say the same thing about $G(r)$ (specifically, when $r = r'$).

For a bit string v, let $I(v)$ denote the unique integer such that v is a binary representation of $I(v)$.

If a simulated decryption oracle knows s (it will be one of the adversary's H-queries), then $X = I(t)$ is a solution to the equation

$$(X + 2^{k_0} I(s))^e \equiv y \pmod{N}.$$

To find $I(t)$, we can apply Coppersmith's algorithm [Cop96]. This algorithm works provided $I(t) < N^{1/e}$, which is guaranteed by our assumption that $k_0 \leq \log_2 N/e$.

More precisely, for all $s^* \in S_H$, the simulated decryption oracle tries to find a corresponding solution t^* using Coppersmith's algorithm. If all of these attempts fail, then the simulator rejects y. Otherwise, knowing s and t, it decrypts y in the usual way.

We can also apply Coppersmith's algorithm in the step of the proof where we use the adversary to help us to extract a challenge instance of the RSA problem.

Not only does this prove security, but we get a potentially more efficient reduction—the implied inverting algorithm has a running time roughly equal

to that of the adversary, plus $O(q_D q_S T_C)$, where T_C is the running time of Coppersmith's algorithm.

References

[BDPR98] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among notions of security for public-key encryption schemes. In *Advances in Cryptology–Crypto '98*, pages 26–45, 1998.

[BR93] M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *First ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[BR94] M. Bellare and P. Rogaway. Optimal asymmetric encryption. In *Advances in Cryptology—Eurocrypt '94*, pages 92–111, 1994.

[CGH98] R. Canetti, O. Goldreich, and S. Halevi. The random oracle model, revisted. In *30th Annual ACM Symposium on Theory of Computing*, 1998.

[Cop96] D. Coppersmith. Finding a small root of a univariate modular equation. In *Advances in Cryptology–Eurocrypt '96*, pages 155–165, 1996.

[DDN91] D. Dolev, C. Dwork, and M. Naor. Non-malleable cryptography. In *23rd Annual ACM Symposium on Theory of Computing*, pages 542–552, 1991.

[DDN00] D. Dolev, C. Dwork, and M. Naor. Non-malleable cryptography. *SIAM J. Comput.*, 30(2):391–437, 2000.

[MvOV97] A. Menesez, P. van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.

[NY90] M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *22nd Annual ACM Symposium on Theory of Computing*, pages 427–437, 1990.

[RS91] C. Rackoff and D. Simon. Noninteractive zero-knowledge proof of knowledge and chosen ciphertext attack. In *Advances in Cryptology–Crypto '91*, pages 433–444, 1991.

[RSA78] R. L. Rivest, A. Shamir, and L. M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, pages 120–126, 1978.

[Sho97] V. Shoup. Lower bounds for discrete logarithms and related problems. In *Advances in Cryptology–Eurocrypt '97*, 1997.