Securely Combining Public-Key Cryptosystems

Stuart Haber Benny Pinkas

STAR Lab, Intertrust Tech.
821 Alexander Road
Princeton, NJ 08540
{stuart,bpinkas}@intertrust.com

Abstract

It is a maxim of sound computer-security practice that a cryptographic key should have only a single use. For example, an RSA key pair should be used only for public-key encryption or only for digital signatures, and not for both.

In this paper we show that in many cases, the simultaneous use of related keys for two cryptosystems, e.g. for a public-key encryption system and for a public-key signature system, does not compromise their security. We demonstrate this for a variety of public-key encryption schemes that are secure against chosen-ciphertext attacks, and for a variety of digital signature schemes that are secure against forgery under chosen-message attacks. The precise form of the statement of security that we are able to prove depends on the particular cryptographic schemes in question and on the cryptographic assumptions needed for their proofs of security; but in every case, our proof of security does not require any additional cryptographic assumptions.

Among the cryptosystems that we analyze in this manner are the public-key encryption schemes of Cramer and Shoup, Naor and Yung, and Dolev, Dwork, and Naor, which are all defined in the standard model, while in the random-oracle model we analyze plaintext-aware encryption schemes (as defined by Bellare and Rogaway) and in particular the OAEP+ cryptosystem. Among public-key signature schemes, we analyze those of Cramer and Shoup and of Gennaro, Halevi, and Rabin in the standard model, while in the random-oracle model we analyze the RSA PSS scheme as well as variants of the El Gamal and Schnorr schemes. (See references within.)

1 Introduction

It is conventional wisdom that keys used by different cryptosystems must be independent. This principle was only encouraged by the recognition that the textbook versions of the early public-key cryptosystems based on number-theory problems had the property that we now call malleability, related to their random self-reducibility (as first pointed out for RSA by [9]). From the very beginning, researchers advocated adding redundancy to messages (as Rabin did in one of the founding papers of our field [19]), but a sophisticated understanding of methods to do this, whose security we can properly analyze, is much newer.

Most proofs of security of a cryptographic scheme assume that it uses randomly chosen keys that are not used by any other application; these proofs say nothing about the combined use of two schemes that are secure when used in isolation. In other words, although two cryptosystems $\mathcal{C}$ and $\mathcal{C}'$ might have been proven to be secure by themselves, the availability of cryptosystem $\mathcal{C}$ might reduce the security of cryptosystem $\mathcal{C}'$ if their respective keys are related.

Encryption schemes and signature schemes are commonly used in combination, most often with each of them using independent keys. We study the use of encryption and signature schemes whose keys are not independent of each other. As we demonstrate below, it turns out that, in many cases, a signature scheme *can* be securely used in combination with an encryption scheme with which it shares a key.

The need to do this occurs in practice. For example, a large system may be carefully designed so that all cryptographic operations between parties are performed using public keys for a digital signature algorithm (and any encryption is performed using authenticated session keys generated, say, by using a Diffie-Helman key exchange). If a new function is demanded of the system that entails the use of public-key encryption, it would be desirable to add this capability to the system without substantially increasing the attendant overhead of managing new keys and certificates. The relevant engineering considerations can range from the hardware constraints of small devices to the size of key-management databases (e.g., of revocation lists). Perhaps the most important motivation in practice may be the desire to minimize the number of parts of the existing system where changes must be made in order to add the new encryption functionality, since the trouble entailed by each additional change may be very high.

Our motivation in addressing the security of cryptosystems with related keys is theoretical as well as practical. It is worthwhile to investigate the validity of the assertions of engineering folklore, such as the advice always to use independent keys in different cryptosystems. More generally, one shouldn't just study systems in isolation but rather in the context of their actual use, as part of and in conjunction with other systems.

We emphasize that the designer of a system must be very careful in using two cryptosystems with related keys. The preferred choice should always be to have each cryptosystem use independent keys. Related keys should only be used when there is a *proof*, rather than just a heuristic argument, that the combination is secure. Furthermore, cryptographic security is usually not the only relevant consideration. There are many situations in which the wider context of their use dictates different requirements for public-key encryption algorithms and digital-signature algorithms in the same system. Even when it makes sense to use the same algorithmic primitive—RSA, say—both for encryption and for signatures, the keys may have different lifetimes in the two cases, they may have different storage and protection requirements, and they may be subject to different revocation policies and different access rights. For example, if the digital signatures should have the property of non-repudiation, then every signing key should be discarded after its lifetime has ended; but decryption keys are often archived for the purposes of later data recovery. In this study we address only the question of cryptographic security.

To stress the danger of using related keys, we note that in the realm of symmetric-key cryptography researchers have developed cryptanalytic attacks (starting with Biham's result in [4]) against the use of several instances of the same encryption scheme with related keys.

While we have been able to prove that, for a number of different algorithm choices, public-key cryptosystems with related keys may be used securely in combination, we have not obtained a general characterization of when this is possible. This is the subject of on-going study.

1.1 Our approach

The usual method of proving the security of a cryptosystem involves the description of an adversary with precisely specified powers, which is given a precise challenge. For example, in the case of an encryption scheme, one sort of adversary is allowed to query a decryption oracle with ciphertexts of its choice, and is then given the challenge of distinguishing between encryptions of two different messages. A typical proof of security shows that the existence of a successful adversary with a certain bound on its computational resources (run-time, number of queries, and so on) contradicts a "cryptographic assumption" about the infeasibility of a specific computational problem.

We are studying the combined use of a pair of public-key cryptosystems $\mathcal{C}$ and $\mathcal{C}'$. In general, each of these may either be a public-key signature scheme or a public-key encryption scheme. Now suppose that each of $\mathcal{C}$ and $\mathcal{C}'$ has a proof of security—with a definite class of adversaries, under a definite cryptographic assumption—when it is used in isolation. We would like to reason about the combined use of the two systems where their respective keys are not independent.

Our general approach is to consider an adversary $\mathcal{A}$ for cryptosystem $\mathcal{C}$ (the *attacked* cryptosystem), which we endow with the ability to query an oracle for cryptosystem $\mathcal{C}'$ (the *oracle* cryptosystem). If cryptosystem $\mathcal{C}'$ is an encryption scheme we give the adversary access to a decryption oracle, and if it is a signature scheme we give it access to a signature-generation oracle. The adversary $\mathcal{A}$ can ask this oracle any legitimate query, including queries based on its interaction with cryptosystem $\mathcal{C}$.

The desired result is that in this scenario $\mathcal{A}$ has no greater probability of success in attacking system $\mathcal{C}$ than it would if it did not have access to the oracle for system $\mathcal{C}'$. We prove this property by constructing a *simulator* that does not have the private keys of system $\mathcal{C}$, and is able nonetheless to answer the adversary's queries to the $\mathcal{C}'$-oracle in a manner that is indistinguishable from that of an oracle that does have full knowledge of the private keys of system $\mathcal{C}'$.

Given such a simulator, the hypothesized security proof for system $\mathcal{C}$ therefore holds as well in the situation where users combine system $\mathcal{C}$ with system $\mathcal{C}'$. If this were not the case, an adversary that could break the combined system could be converted into an adversary that breaks $\mathcal{C}$ alone.

The two schemes $\mathcal{C}$ and $\mathcal{C}'$ may be completely independent, using independently chosen key pairs, but we are particularly interested in the case where they are not independent: for example, when $\mathcal{C}$ is a signature scheme and $\mathcal{C}'$ is an encryption scheme, both of them based on the discrete log or the Diffie-Hellman problem, and a typical combined use of them involves the choice of a pair of the form (x, g^x) where the exponent x is part of the user's private key and the group element g^x is part of the public key of *both* schemes. The precise relation between the keys in $\mathcal{C}$ and the keys in $\mathcal{C}'$ depends, of course, on the particular schemes $\mathcal{C}$ and $\mathcal{C}'$.

If $\mathcal{C}$ and $\mathcal{C}'$ are indeed completely independent then it is easy to see that the combined use of the two schemes together is secure. If an adversary $\mathcal{A}$ can attack $\mathcal{C}$ when it is given oracle access to system $\mathcal{C}'$, then $\mathcal{A}$ has exactly the same probability of success in attacking $\mathcal{C}$ when it interacts instead with a simulator that chooses a $\mathcal{C}'$-instance at random, independently of the $\mathcal{C}$-instance under attack. The simulator is able to answer all queries from $\mathcal{A}$ because it has chosen the private key(s) of the particular $\mathcal{C}'$-instance.

The situation is more complex when the two systems have related keys: *a priori* there is no reason for complete knowledge of the keys of cryptosystem $\mathcal{C}'$ to be available to a simulator, since these keys are not independent of the keys of cryptosystem $\mathcal{C}$, which must be kept secret from it. It is therefore unclear whether one could simulate the answers of a system-$\mathcal{C}'$ oracle to legitimate queries from the adversary.

It is perhaps surprising that in many cases we *can* simulate the answers of $\mathcal{C}'$. In some cases this is possible even though the private key of $\mathcal{C}'$ is identical to that of $\mathcal{C}$, and therefore the simulator has no information about it. In other cases the simulator has partial knowledge of the private key of $\mathcal{C}'$, but no information about certain parts of this key.

If we can prove a similar result about $\mathcal{C}'$ and $\mathcal{C}$ with the roles reversed—with $\mathcal{C}'$ as the attacked cryptosystem, this time, and $\mathcal{C}$ as the oracle cryptosystem—then one may use $\mathcal{C}$ and $\mathcal{C}'$ in combination without compromising the security of either system due to the presence of the other one.

1.2 Results

Most of this work explores scenarios where the cryptosystems are either encryption schemes or signature schemes.

In the case where the oracle cryptosystem is an encryption scheme, we provide results for the following schemes:

- In the standard model: the Cramer-Shoup [7] and the Dolev-Dwork-Naor [10] encryption schemes, which are secure against adaptive chosen-ciphertext attacks; and the Naor-Yung scheme [17], which is secure against non-adaptive chosen-ciphertext attacks.

- In the random-oracle model: a proof for the OAEP+ [23] encryption scheme, which is an example of a plaintext-aware cryptosystem ([2]), and a sketch of a proof for the use of plaintext-aware cryptosystems in general.

In the case where the oracle cryptosystem is a signature scheme, we provide results for the following schemes:

- In the standard model: the Cramer-Shoup [8] and the Gennaro-Halevi-Rabin [14] signature schemes, whose security is based on the strong RSA assumption (the scheme of [14] also relying on an assumption about the properties of a hash function).
- In the random-oracle model: El Gamal signatures, if they use a hash function H which is modeled as a random oracle, as suggested in [18]; RSA based signatures according to the PSS construction of [3]; and Schnorr's signature scheme [21].

Remark 1: For each oracle cryptosystem we must define the relation that holds between its private keys and those of the attacked cryptosystem. In all the systems we explore, we make the natural assumption that some part of the private key of the oracle system is part of the private key of the attacked cryptosystem, and is therefore completely unknown to the simulator. If the private key contains additional information, we assume it to have the same distribution as it has when the oracle system is used in isolation. In many cases—namely, all the results we provide for cryptosystems whose security is proved in the random-oracle model—our results hold even if the simulator has no information whatsoever about the private key of the oracle cryptosystem.

Remark 2: The *attacked* cryptosystem is not required to have any specific type of security (e.g. chosen-ciphertext security, existential forgery, etc.). We only need to provide a proof that depends on the properties of the *oracle* system, and this proof ensures that the security of the attacked system—whatever its actual strength—is not degraded due to its use in conjunction with the oracle system.

Remark 3: Our main motivation is a scenario in which one system is an encryption scheme and the other system is a signature scheme. We refer to this scenario throughout the paper. There is, however, no obligation to limit the type of the attacked system. In particular, it could be of the same type as the oracle system; e.g., both may be encryption schemes. An interesting example is the case of the Naor-Yung cryptosystem, which is only known to be secure against non-adaptive attacks. We describe in Section 3.3 a setting with two instances of this scheme having related keys, which is secure even though the adversary can query one instance after observing the challenge for the other instance.

2 Notation and Definitions

2.1 Basic Definitions

We first provide definitions for public key encryption and signature schemes, and then define their combined use.

Definition 1 (PUBLIC-KEY ENCRYPTION SCHEME) *A* public-key encryption scheme *is a triple of polynomial-time algorithms* $\mathsf{Enc} = (K, E, D)$ *as follows:*

- *The* key-generation algorithm K *is a probabilistic algorithm taking a security parameter k as input (in unary), and producing a pair (e, d) of corresponding public encryption and private decryption keys.*
- *The* encryption algorithm E *is a probabilistic algorithm taking as input a public encryption key e and a message $m \in \{0,1\}^*$, and producing as output a ciphertext $y \in \{0,1\}^*$; we write $E_e(m) = y$.*
- *The* decryption algorithm D *is a deterministic algorithm taking as input a private decryption key d and a ciphertext string $y \in \{0,1\}^*$, and producing as output either a message $m \in \{0,1\}^*$ or a special* reject *symbol.*

We require that if $K(k)$ outputs a key-pair (e, d) and $E_e(m)$ outputs y, both with positive probability, then $D_d(y) = m$.

Definition 2 (PUBLIC-KEY SIGNATURE SCHEME) *A* public key signature scheme *is a triple of polynomial-time algorithms* $\mathsf{Sig} = (K, S, V)$ *as follows:*

- *The* key-generation algorithm K *is a probabilistic algorithm taking a security parameter* k *as input (in unary), and producing as output a pair* (s, v) *of corresponding private signature and public verification keys.*

- *The* signature algorithm S *is a probabilistic algorithm taking as input a private signature key* s *and a message* $m \in \{0,1\}^*$ *and producing as output a signature* $\sigma \in \{0,1\}^*$*; we write* $S_s(m) = \sigma$*.*

- *The* verification algorithm V *is a deterministic algorithm taking as input a public verification key* v *and a message-signature pair* (m, σ) *and producing as output either* accept *or* reject*.*

We require that if $K(k)$ *outputs a key-pair* (v, s) *and* $S_s(m)$ *outputs* σ*, both with positive probability, then* $V_v(m, \sigma) = \mathsf{accept}$*; and for any other pair* (m', σ')*,* $V_v(m', \sigma') = \mathsf{reject}$*.*

Definition 3 (COMBINED PUBLIC-KEY SCHEME) *Given an encryption scheme* $\mathsf{Enc} = (K_1, E, D)$ *and a signature scheme* $\mathsf{Sig} = (K_2, S, V)$*, the* combined public-key scheme $\Sigma = (\mathsf{Enc}, \mathsf{Sig})$ *is a suite of algorithms* (K, E, D, S, V)*.*

- *The key-generation algorithm* K *is a probabilistic algorithm that, given a security parameter* k*, produces two pairs of keys* $[(e, d), (v, s)]$*, one for* Enc *and one for* Sig*.*

- *Encrypting and decrypting are performed with* E *and* D*, and signing and verifying are performed with* S *and* V*, exactly as in the respective stand-alone schemes.*

The only difference between the combined scheme and the original schemes is the key generation algorithm. Typically, the probability space $K(k)$ chosen by K with security parameter k will depend on $K_1(k)$ and $K_2(k)$. The distribution of (e, d) in $K(k)$ is the same as its distribution in $K_1(k)$, and the same holds for the distribution of (s, v) in $K(k)$ and $K_2(k)$. However the distributions of (e, d) and (s, v) in $K(k)$ might not be independent. Several examples may be found in the later sections of this paper.

2.2 Security Definitions

We first describe the setting in which an encryption scheme Enc is the attacked cryptosystem and a signature scheme Sig is the oracle cryptosystem; this is the setting for our results in §4 below.

An *encryption-scheme adversary* is a probabilistic algorithm $\mathcal{A}$ that is meant to attack a given encryption scheme $\mathsf{Enc} = (K, E, D)$, along with a specification of its *attack scenario*, including the experiment that measures its success. Depending on the scenario, $\mathcal{A}$ may or may not have access to a decryption oracle during certain parts of its run; for example, if $\mathcal{A}$ is limited to chosen-plaintext attacks, then it cannot access a decryption oracle at all. The attack scenario may include an experiment along the following lines: After the choice of a key pair $K(k) = (e, d)$, $\mathcal{A}$ is given e and then runs in two stages, the first of which ends with its output of a pair of messages; one of these is then chosen at random and encrypted with $E_e(\cdot)$; the resulting ciphertext is given to $\mathcal{A}$'s second stage, whose challenge is to guess which of the two messages was encrypted. One can adapt this general definition to obtain the common adversarial models of chosen-plaintext attack, chosen-ciphertext attack in the pre-processing mode (or non-adaptive CCA, as defined in [17]), and chosen-ciphertext attack in the post-processing mode (or adaptive CCA, as defined in [20]).

If the encryption scheme Enc is used as part of a combined scheme (Enc, Sig) as in Definition 3 above, we may augment the encryption-scheme adversary by allowing it to make use of Sig in its attack. This may be formalized with the following *augmented encryption-scheme attack scenario*:

Definition 4 (AUGMENTED ENCRYPTION SCHEME ATTACK) *An adversary operates against* Enc *when it is used as part of the combined scheme* $\Sigma = (\mathsf{Enc}, \mathsf{Sig}) = (K, E, D, S, V)$*. The following experiment measures the adversary's success.*

Keys are chosen for the encryption and signature scheme using the key generation algorithm $K(k) = [(e,d),(s,v)]$. The adversary is given e and v.

The attack experiment for the adversary attacking Enc *is run, with the following addition: Besides any access that this attack scenario provides to a $D_d(\cdot)$-oracle, the adversary is also allowed to make queries of its choice to an oracle for $S_s(\cdot)$. That is, the adversary is given legitimate signatures on messages of its choice. These queries can be made at any time, even after the challenge is given to the adversary.*

Definition 5 (SECURITY OF AN ENCRYPTION SCHEME IN A COMBINED PUBLIC-KEY SCHEME) *The combined scheme $\Sigma = (\mathsf{Enc}, \mathsf{Sig})$ does not compromise the security of* Enc *if the following holds:* for any augmented encryption scheme adversary $\mathcal{A}$ for the combined scheme, there exists an adversary $\mathcal{A}'$ for Enc alone with success probability at most negligibly worse than the success probability of $\mathcal{A}$ *(where the probabilities are taken over the key space and over the random choices of the adversary and of the oracles it queries).*

In all of the examples for this setting that we consider in §4 below, $\mathcal{A}'$ operates by:

1. taking the public key e that $\mathcal{A}'$ is given by $K_1(k)$;
2. choosing a Sig-key (s,v) so that $[(e,d),(s,v)]$ has the correct distribution $K(k)$ for the augmented encryption-scheme attack;
3. running the given adversary $\mathcal{A}$, while being able to simulate the answers that $\mathcal{A}$ would receive for its queries to the $S_s(\cdot)$-oracle.

We have to show that the simulator's answers are indistinguishable from those that $\mathcal{A}$ would receive in an actual run of its attack experiment. The challenge in showing this is that $\mathcal{A}'$ does not have complete knowledge of the private signature key s, which is related to the private decryption key d, which in turn is unknown to $\mathcal{A}'$.

Next we describe the setting in which the signature scheme Sig is the attacked cryptosystem and the encryption scheme Enc is the oracle cryptosystem; this is the setting for our results in §3 below.

A *signature-scheme adversary* is a probabilistic algorithm $\mathcal{A}$ that is meant to attack a given signature scheme $\mathsf{Sig} = (K, S, V)$, along with a specification of its *attack scenario*, including the experiment that measures its success. Depending on the scenario, $\mathcal{A}$ may or may not have access to a signing oracle during certain parts of its run. In parallel with our definitions above for the encryption schemes, we next describe an *augmented signature-scheme attack scenario.*

Definition 6 (AUGMENTED SIGNATURE SCHEME ATTACK) *An adversary operates against* Sig *when it is used as part of the combined scheme $\Sigma = (\mathsf{Enc}, \mathsf{Sig}) = (K, E, D, S, V)$. The following experiment measures the adversary's success.*

Keys are chosen for the combined scheme using the key generation algorithm $K(k) = [(e,d),(s,v)]$. The adversary is given e and v.

The attack experiment for the adversary attacking Sig *is run, with the following addition: Besides any access that our attack scenario provides to an $S_s(\cdot)$-oracle, the adversary is allowed to make queries to an oracle for $D_d(\cdot)$; that is, the adversary can decrypt ciphertext strings of its choice.*

Definition 7 (SECURITY OF A SIGNATURE SCHEME IN A COMBINED PUBLIC-KEY SCHEME) *The combined scheme $\Sigma = (\mathsf{Enc}, \mathsf{Sig})$ does not compromise the security of* Sig *if the following holds:* for any augmented signature scheme adversary $\mathcal{A}$ for the combined scheme, there exists an adversary $\mathcal{A}'$ for Sig alone with success probability at most negligibly worse than the success probability of $\mathcal{A}$.

In all of the examples for this setting that we consider in §3 below, we construct $\mathcal{A}'$ by:

1. taking the public key v that $\mathcal{A}'$ is given by $K_2(k)$;
2. choosing an Enc-key (e,d) so that $[(e,d),(s,v)]$ has the correct distribution $K(k)$;

3. running the given adversary $\mathcal{A}$, while being able to simulate the answers that $\mathcal{A}$ would receive for its queries to the $D_d(\cdot)$-oracle.

As in the parallel setting for encryption schemes, we have to show that the simulator's answers are identical to those that $\mathcal{A}$ would receive in an actual run of its attack experiment.

We can similarly combine two different encryption schemes or two different signature schemes, making the appropriate changes in the definitions for the two settings described above. The main issue is not the type of the two cryptosystems that are used together, but rather whether either one of them affects the security of the other.

3 Security in the Presence of Encryption Schemes

This section describes several public-key encryption schemes that can be used in conjunction with a different cryptosystem, typically a signature scheme, while using related keys and without affecting the security of the other cryptosystem. In other words, the signature scheme is the attacked cryptosystem and the encryption scheme is the oracle cryptosystem.

For each specific encryption scheme Enc, in order to demonstrate that Enc can be safely combined with a specific signature scheme Sig, we must first define the relation between the private key of Enc and the private key of Sig; that is, we must specify how the key generation algorithm of the combined scheme operates. In all the schemes we discuss the following natural relation holds between the keys: One part of the private key of the encryption scheme, denoted d_1, is also part of the private key of the attacked scheme, and is therefore completely defined (and unknown to the adversary in our proofs of security). The rest of the private key of the encryption scheme, denoted d_2, is distributed to ensure that the combined distribution of $d = (d_1, d_2)$ has the same distribution it has when it is generated by the key-generation algorithm of the encryption scheme alone, subject to the constraint that d_1 has a particular value.

The second step we must take is to show that the conditions required by Definition 7 above are fulfilled: for any adversary $\mathcal{A}$ operating against the combined scheme there is an adversary $\mathcal{A}'$ operating against the signature scheme alone (the attacked cryptosystem) with success probability close to that of $\mathcal{A}$. Therefore, $\mathcal{A}$ gains nothing by having access to the decryption oracle and decrypting ciphertexts of its choice. This type of result is shown by providing a simulator S that has exactly the same knowledge that $\mathcal{A}'$ has about the keys of the signature scheme Sig. In particular, S does not know the secret key, and therefore does not have full knowledge of the decryption key of the decryption oracle $D_d(\cdot)$ of Enc. We show that if $\mathcal{A}$ interacts with S it obtains the same information that it obtains interacting with Enc. Therefore the interaction with Enc does not increase $\mathcal{A}$'s success in breaking Sig to be greater than the success probability of $\mathcal{A}'$.

3.1 Results

In all the encryption schemes that we study in this section, the decryption process includes a first stage in which the validity of the ciphertext is checked, and a second stage in which the ciphertext is decrypted. The simulators we design are able to perform the first stage in exactly the same way as in the original decryption process, while the second stage must be changed in order to compensate for the fact that parts of the private key are unknown to the simulator.

In the standard model we give combined-scheme security results for several schemes:

We treat the Cramer-Shoup scheme, which provides chosen ciphertext security against adaptive attacks without any random-oracle assumption, based on the Decision Diffie-Hellman problem. We also give a sketch of a proof for the Dolev-Dwork-Naor scheme, which is a generic and less efficient construction that provides the same level of security. (A brief description of our result concerning this system appears in §3.3 discussing the Naor-Yung scheme, since the proof techniques are similar).

The Naor-Yung encryption scheme, which is secure against non-adaptive (or "lunch-time") chosen-ciphertext attacks. This scheme depends on the use of an encryption scheme that is secure against chosen-plaintext attack, and on the security of a scheme for non-interactive zero-knowledge proofs.

In the random-oracle model we present results for the OAEP+ encryption scheme of Shoup [23]. This scheme satisfies the property of "plaintext-awareness" defined by [2] in order to describe encryption schemes for which the only correctly formatted ciphertexts—i.e., the only ciphertexts that will be decrypted—are those which were computed by a party that is "aware" of the plaintext. We also sketch a general proof for plaintext-aware schemes, which turns out to be rather straightforward.

3.2 The Cramer-Shoup system

The Cramer-Shoup encryption scheme [7] is quite practical and provides chosen-ciphertext security against adaptive attacks. This is achieved in the standard security model, based on the hardness of the Decision Diffie-Hellman problem in a cyclic group G of large prime order q.

The scheme operates in the following way. We describe the slight variation given in [22]. (We follow the author's notation, which gives different meaning to the symbols e, d than we gave in §2.)

Key Generation: A group G (along with its size, q), and random elements $g_1, g_2 \in G$ and $x_1, x_2, y_1, y_2, z \in (\mathbf{Z}/q\mathbf{Z})$ are chosen. The public key is

$$(g_1,\ g_2,\ c = g_1^{x_1} g_2^{x_2},\ d = g_1^{y_1} g_2^{y_2},\ h = g_1^z),$$

together with a universal one-way hash function $H : \{0,1\}^* \to (\mathbf{Z}/q\mathbf{Z})$. The private key is (x_1, x_2, y_1, y_2, z).

Encryption: The input is a plaintext message $m \in G$. The encryption algorithm chooses a random element $r \in (\mathbf{Z}/q\mathbf{Z})$ and computes

$$u_1 = g_1^r,\ u_2 = g_2^r,\ e = h^r m,\ \alpha = H(u_1, u_2, e),\ v = c^r d^{r\alpha}.$$

The ciphertext is (u_1, u_2, e, v).

Decryption: Given a ciphertext (u_1, u_2, e, v) the decryption algorithm first computes α and examines the equation $u_1^{x_1+y_1\alpha} u_2^{x_2+y_2\alpha} = v$. If it does not hold then the ciphertext is rejected; otherwise the output is $m = e(u_1^z)^{-1}$.

Relation between keys: We consider now what sort of relation makes sense between the keys of a Cramer-Shoup instance and the keys of a signature scheme which is used in combination with it. The most natural scenario is to use the Cramer-Shoup instance together with a signature scheme that also uses the group G, having (g_1, h) as its public key and z as its private key. This applies to a number of signature schemes that are defined over cyclic groups, such as El Gamal, DSA, or Schnorr's signature scheme. Let Sig denote any of these schemes, using the same group G as our Cramer-Shoup instance, with (g_1, h) as the public verification key. The other keys of the Cramer-Shoup instance are distributed exactly as in the definition of the key generation of the scheme, independently of the signature system.

Simulating decryption: Given an adversary $\mathcal{A}$ that attacks a signature scheme Sig when used together with the Cramer-Shoup scheme as described above, we construct an adversary $\mathcal{A}'$ attacking Sig alone. This adversary $\mathcal{A}'$ runs $\mathcal{A}$ as a black box and must therefore simulate the operation of a decryption oracle for the Cramer-Shoup instance.

The adversary $\mathcal{A}'$ operates in the following manner:

1. $\mathcal{A}'$ is given the public key $v = (g_1, h)$ of Sig.

2. $\mathcal{A}'$ defines keys for the Cramer-Shoup instance. It chooses a random exponent x and defines $g_2 = h^{x^{-1}}$, where x^{-1} satisfies $xx^{-1} \equiv 1 \bmod q$ (and therefore $h = g_2^x$). It then chooses random elements $x_1, x_2, y_1, y_2 \in G$ and sets $c = g_1^{x_1} g_2^{x_2}, d = g_1^{y_1} g_2^{y_2}$ as in the original scheme. The distribution of the keys is exactly that of the keys in the original scheme, subject to the constraint that (g_1, h) are fixed as part of the public key of the attacked signature scheme.

3. $\mathcal{A}'$ runs the given adversary $\mathcal{A}$ against the combined use of the two schemes, and provides a decryption oracle for the Cramer-Shoup instance. When $\mathcal{A}$ asks to decrypt a ciphertext (u_1, u_2, e, v) the decryption oracle first verifies that $u_1^{x_1+y_1\alpha} u_2^{x_2+y_2\alpha} = v$, using its knowledge of x_1, x_2, y_1, y_2. If this equation holds it computes the plaintext $m = e(u_2^x)^{-1}$. (That is, $\mathcal{A}'$ uses its knowledge of $\log_{g_2} h$ for performing the decryption, instead of using $\log_{g_1} h$ as in the original scheme).

It can easily be verified that the view of $\mathcal{A}$ is identical to its view in an actual run against a combined instance of Sig and the Cramer-Shoup scheme, and its probability of success is therefore unchanged.

3.3 The Naor-Yung System

The Naor-Yung encryption scheme [17] was the first to provide chosen-ciphertext security (although only against "lunch-time" attacks, giving chosen-ciphertext security in the pre-processing mode, and not against adaptive attacks). Security is proven in the standard model based on a cryptographic assumption and without modeling any function as a random oracle.

On a high level, the Naor-Yung encryption scheme operates in the following way: The scheme uses two independently chosen instances (E_1, E_2) of a chosen-plaintext secure encryption scheme, and a noninteractive zero knowledge (NIZK) proof system [5]. To encrypt a message, one encrypts it twice, once using each of the two encryption keys, and provides a NIZK proof for the statement that the two ciphertexts are encryptions of the same plaintext. In the decryption phase the owner of the private key first verifies the NIZK proof, and if it is valid it decrypts one of the ciphertexts using one of its private keys. It is important for us that the verification of the NIZK proof does not require knowledge of the private key, and therefore can be performed by any party.

Relation between keys: To set the Naor-Yung scheme in our context we assume that the keys of one of the two encryption instances, say E_1, are related, or possibly equal, to the keys of the signature scheme Sig and are therefore unknown to our adversary. The keys of E_2 are distributed according to their original distribution, independently of Sig. The key of the Naor-Yung scheme also contains a random string R which is used for the NIZK proofs.

Simulating the decryption oracle: The construction of a simulator S that provides decryptions of legitimate ciphertexts without knowing the private keys of E_1 is straightforward. The simulator chooses a key pair for E_2, and sets the public key of its encryption scheme to contain the public keys of E_1 and E_2 plus a random string R. When S receives a ciphertext it first checks the NIZK proof to verify that the two encryptions under the two public keys are of the same plaintext. If this is correct, it uses the private key of E_2 to decrypt the second encryption. This is the correct decryption, since the soundness of the NIZK proof ensures that the two plaintexts are equal. The view of the adversary is identical to its view in interacting with the combined cryptosystem, up to a negligible error that depends on the error probability of the NIZK proof.

Combined use of two Naor-Yung schemes: Assume that the adversary has access to two instances of the Naor-Yung scheme, NY and NY', which share one of their keys. That is, cryptosystem NY has keys (E_1, E_2, R) and NY' has keys (E_1', E_2', R') with $E_1 = E_1'$, while the other keys are independent of E_1 and of each other. Suppose also that the adversary can present only "lunch-time" chosen-ciphertext queries to NY, but can present adaptive queries to NY' even *after* it receives a challenge for NY. It is somewhat surprising that this combined scheme does not affect the security of NY, even though the Naor-Yung scheme is not known to be secure against adaptive chosen-ciphertext attacks.

To see why this holds, assume that there is an adversary $\mathcal{A}$ that breaks the combined scheme. We show how to use it to construct an adversary $\mathcal{A}'$ that breaks NY with a lunch-time attack. $\mathcal{A}'$ first chooses a random E_2' and R', and sends the public keys of NY and NY' to $\mathcal{A}$. Before receiving the challenge ciphertext, it can forward $\mathcal{A}$'s queries about NY to NY's decryption oracle. After receiving the challenge, $\mathcal{A}$ can only ask queries about NY', and $\mathcal{A}'$ can answer these since it knows the decryption key for E_2'.

A sketch of a proof for the Dolev-Dwork-Naor scheme: The Dolev-Dwork-Naor scheme [10], which preceded the Cramer-Shoup scheme, provides the same security against adaptive chosen-ciphertext attacks, using a generic construction which is considerably less efficient. Without going into detail, the scheme of [10] uses k pairs of encryption keys, all chosen according to a scheme which is chosen-plaintext secure. When a party generates an encrypted message it must choose a signature key, and a hash of the corresponding verification key chooses one encryption key out of every pair. Every bit of the message is encrypted with each of the k chosen keys, and in addition the ciphertext is signed and contains a NIZK proof of consistency among the encryptions. The proof that this scheme can be securely used in combination with a signature scheme is similar to the proof given for the Naor-Yung scheme. Namely, most of the keys of the system can be part of the private key of the attacked scheme and therefore unknown, since it is sufficient to be able to decrypt only one of the k encryptions of the message. This is achieved, for example, when one of the k pairs of encryption schemes is independent of the key of the attacked cryptosystem. Therefore, in every choice of keys that is made in the encryption process, one of the two keys in this pair is used to encrypt the message, and the result can be decrypted by the simulator. Alternatively, it is sufficient that a large sample of the keys in different pairs are independent of the key of the attacked cryptosystem, so that for every encrypted message there is a high probability that one of these keys is used for encryption, and the resulting ciphertext can be decrypted by the simulator.

Remark: Our results show that using an encryption scheme whose keys are related to those of a signature scheme does not reduce the security of the latter. It seems, therefore, that if encryption scheme E_1 provides weaker security than encryption scheme E_2, it should be more difficult to prove the result with respect to scheme E_1 than with respect to scheme E_2. Therefore, the result regarding the Naor-Yung scheme (which does not provide security against adaptive post-processing attacks) seems stronger than the results regarding the Cramer-Shoup and the Dolev-Dwork-Naor schemes.

3.4 Plaintext-Awareness and OAEP+

3.4.1 Plaintext-awareness

Plaintext-awareness was introduced by Bellare and Rogaway in [2], and a modified definition was given in [1]. We follow Shoup [23] and call these definitions PA1 and PA2, respectively. This notion of security applies only in the random-oracle model.

Intuitively, an encryption scheme provides plaintext awareness if an adversary that sends ciphertext queries to a decryption oracle can only receive decryptions of ciphertexts for which it is already "aware" of the plaintexts that are encrypted by these ciphertexts.

This property is captured by the requirement that there be a *plaintext extractor* which, given a ciphertext along with a transcript of the interaction of the adversary with the hash functions (or any other functions in the scheme) that are modeled as random oracles, can output the plaintext that is encrypted by the given ciphertext. Bellare et al. [1] have shown PA2 to be strictly stronger than adaptive chosen-ciphertext security in the random-oracle world.

The OAEP scheme [2] was proven to have the PA1 property, and its instantiation with the RSA cryptosystem, RSA-OAEP, is part of two industry standards, PKCS #1, version 2 and IEEE P1363. Recently Shoup [23] has shown that one cannot prove that the PA1 definition provides adaptive chosen-ciphertext security if the proof only uses black box reductions. (PA1 still provides chosen-ciphertext security against non-adaptive attacks.) In addition, Shoup has suggested a new scheme, OAEP+, which satisfies the PA2 definition (and therefore provides chosen-ciphertext security against adaptive attacks, as proven in [1]). Very recently, Fujisaki et al. [13] have shown that RSA-OAEP (namely, OAEP using the RSA function as the trapdoor permutation) *is* secure, and Boneh [6] has given a simpler variant of OAEP for the RSA and Rabin functions. We will discuss these cryptosystems in the full version of this paper.

Our aim is to prove that the decryption oracle D of a plaintext-aware encryption scheme does not help any adversary to break a signature scheme Sig whose keys may be related to the private keys of the encryption scheme. Intuitively this should be true, since the plaintext-awareness property ensures that by examining a ciphertext and the adversary's interaction with the random oracle, the plaintext extractor can provide the plaintext that is encrypted by the given ciphertext. The plaintext extractor can therefore provide the adversary with the same answers as D, even though (like the adversary) it does not know the private keys of D. We first treat the OAEP+ scheme, and then give a sketch of a proof for general plaintext-aware cryptosystems.

3.4.2 OAEP+

In this section, we prove the security of a combined public-key scheme using OAEP+ [23]. An instance of the scheme uses a trapdoor permutation f, and three functions G, H, H', modeled as random oracles. The key generation algorithm chooses a random trapdoor permutation f, whose inverse, f^{-1}, is the private key. Given a plaintext x, the encryption is performed as follows: (1) A random string r is chosen. (2) Define $s = (G(r) \oplus x)||H'(r||x)$, and $t = H(s) \oplus r$. (3) Set the ciphertext to $y = f(s||t)$. To decrypt a ciphertext y, the decryptor uses the private key to compute $x = f^{-1}(y)$, which it parses as $x = s_L||s_R||t$, and then computes $r = H(s) \oplus t$, $x = G(r) \oplus s_L$, and $c = s_R$. If $c = H'(r||x)$, then x is output as the cleartext; otherwise the ciphertext is rejected.

Relation between keys: Next we prove that it is secure to use OAEP+ in combination with a signature scheme having f^{-1} as (part of) its private key. This relation completely defines the keys of OAEP+.

Simulating decryptions: The simulator interacts with an adversary $\mathcal{A}$ and observes its behavior. The simulator observes the lists of queries that $\mathcal{A}$ sends to the three functions that are modeled as random oracles, and the answers that $\mathcal{A}$ receives. (Our proof does not require the simulator to change the oracle responses to these queries.) Let $S_G, S_H, S_{H'}$ denote these lists. The simulator is required to decrypt the given ciphertext y. In order to do so it follows the operation of the decryption oracle defined in Game G3 in [23].

In more detail, the simulator examines all the pairs of $(r^*, x^*) \in S_{H'}$. For each pair it defines $s^* = (G(r^*) \oplus x^*)||H'(r^*||x^*)$. If $s^* \in S_H$ it computes $t^* = H(s^*) \oplus r^*$ and $y^* = f(s^*||t^*)$. If $y^* = y$, it stops and outputs x^* as the decryption. It is proven in [23] that the answers of this oracle are negligibly close to the answers of a decryption oracle that knows the decryption key f^{-1} of the scheme. This is exactly the property we require of our simulator.

3.4.3 Plaintext-Awareness in general

We give a sketch of a proof that a plaintext-aware encryption scheme with private key d can be used safely in combination with a different scheme that uses d as part of its private key. We describe this for the plaintext-awareness definition given in [1], definition PA2.

Definition The definition of plaintext-awareness given in [1] concerns an adversary B that is given a public key, an oracle for a hash function H that is used by the encryption algorithm and is modeled as a random oracle, and an encryption oracle that provides valid encryptions generated with H as the hash function (but does not provide the corresponding plaintexts).

Let $\Pi = (K, E, D)$ be an encryption scheme. Let B be an adversary, and let KE be a "knowledge extractor" algorithm whose running time is polynomial in the length of its inputs. We describe an experiment of running B and then KE. Let k be a positive integer.

1. Fix H as a random function from the domain of all functions with input and output lengths as required by the encryption algorithm.
2. Run the key generation algorithm to get $(e, d) = K(1^k)$.
3. Run the adversary B and record the following data: (1) The queries that it makes to H, and the corresponding answers. (2) The answers (ciphertexts) given by the encryption oracle. (3)The ciphertext y generated by B. The ciphertext y will be a challenge for KE to decrypt. It must hold that y is not among the answers given by the encryption oracle.

Define the success probability in this experiment to be the probability that KE, given as input the key e and the recorded data, outputs the same answer as a decryption oracle (which knows the private key) for the query y. The encryption scheme Π is *plaintext-aware* if it provides chosen-plaintext security in the sense of indistinguishability; and there is a knowledge-extractor KE that, for any feasible adversary B, has success probability negligibly close to 1.

Using a plaintext-aware encryption scheme as the oracle cryptosystem Consider a plaintext-aware encryption cryptosystem, Enc, which is used in combination with a signature scheme Sig. As in Definition 3, the combined key-generation algorithm generates the keys $((e, d), (s, v))$ so that (e, d) and (s, v) might be related, but each pair alone is distributed according to the distribution of the corresponding cryptosystem. (We capture this relation by making the most difficult assumption for our proof, namely that d is part of s).

We prove that the use of the plaintext-aware encryption cryptosystem Enc does not degrade the security of the signature scheme Sig. We should therefore demonstrate that for any adversary $\mathcal{A}$ that operates against the combined cryptosystem and tries to break Sig, there is an adversary $\mathcal{A}'$ that runs against Sig alone and has a success probability which is at most negligibly worse than the success probability of $\mathcal{A}$. The adversary $\mathcal{A}'$ will run $\mathcal{A}$ as a black box, and should therefore answer the queries that $\mathcal{A}$ generates to a decryption oracle of Enc.

The setting, including the probability distribution of the key space, exactly fits the definition of plaintext-awareness. $\mathcal{A}'$ can therefore run the knowledge extractor KE which provides answers that are negligibly close to those of a decryption oracle for Enc. This ensures that the view that $\mathcal{A}$ sees when it is run as a black box by $\mathcal{A}'$ is negligibly close to its view when it runs against the combined scheme.

4 Security in the Presence of Signature Schemes

This section describes several public-key signature schemes that can be used in conjunction with a different cryptosystem, typically an encryption scheme, that uses related keys. We prove that the use of the signature scheme does not affect the security of the other cryptosystem. In other words, the encryption scheme is the attacked cryptosystem and a signature scheme is the oracle cryptosystem.

For each of these signature schemes, we specify a combined scheme (as in Definition 3) by describing an encryption scheme (or a family of encryption schemes) with related keys; and we prove, following Definition 5, that the combined scheme does not compromise the security of the encryption scheme being used.

For each scheme we describe the relation that can hold between the keys of the two schemes without affecting the security of the encryption scheme. As in §3, the proof method is to show that an adversary for the encryption scheme gains nothing by having access to the signature-generation oracle and asking it to sign arbitrary messages of its choice. This is demonstrated by constructing a simulator that has exactly the same information as the adversary, and is able to give answers that are indistinguishable from those of a signature-generation oracle with knowledge of the private key of the signature scheme.

4.1 Results

In the standard model we provide results for the Cramer-Shoup and the Gennaro-Halevi-Rabin signature schemes. In the random-oracle world we provide results for the El Gamal signature scheme, as modified by Pointcheval and Stern [18] to obtain security against adaptive chosen-message attacks, for the RSA based PSS scheme of Bellare and Rogaway [3], and for the Schnorr signature scheme [21]. The proof for the Schnorr signature scheme will only be provided in the full version of this paper, since it is similar to the other proofs we describe in this section.

An interesting property of our constructions of signature simulators is that none of the simulators has any knowledge about any part of the private key of the signature scheme whose signatures it simulates (in contrast to the simulators for the encryption schemes in the standard model in §3, where the simulators had partial knowledge of the private keys). In the standard model the signature simulators are able to do their job since the public keys are generated in a special way that gives the simulators additional knowledge, while preserving the original distribution of the keys. In the random oracle model the simulators operate by setting the output of the random oracle to appropriate values.

4.2 The Cramer-Shoup signature scheme

Cramer and Shoup proposed a signature scheme and proved it to be secure against adaptive chosen-message attacks in the standard model, under the strong RSA assumption [8]. The scheme is efficient and does not require the signer to maintain any state (unlike, e.g., the signature scheme of Dwork and Naor [11], which is not stateless).

The strong RSA assumption is that given an RSA modulus n and a random $z \in (\mathbf{Z}/n\mathbf{Z})^*$, it is infeasible to find $r > 1$ and $y \in (\mathbf{Z}/n\mathbf{Z})^*$ such that $y^r = z \bmod n$. The signature scheme also uses a hash function H which is collision-intractable. The basic scheme operates as follows:

Key generation: An RSA modulus $n = pq$ is chosen (with p, q prime). The public key is (n, h, x, e'), where $h, x \in_R QR_n$ (the set of quadratic residues mod n) and e' is a random prime of appropriate length (as defined in [8]). The private key is (p, q).

Signature generation: To sign a message m, a random prime $e \neq e'$ and a random $y' \in QR_n$ are chosen. Compute an x' satisfying $(y')^{e'} = x'h^{H(m)} \bmod n$, and solve the equation $y^e = xh^{H(x')} \bmod n$ for y. The signature is (e, y, y').

Signature verification: Given (e, y, y'), first check that e is odd, of the right length, and different from e'. Then compute $x' = (y')^{e'}h^{-H(m)} \bmod n$, and check whether $x = y^e h^{-H(x')} \bmod n$.

Relation between keys: We show that the Cramer-Shoup signature scheme with a public key n can be used in combination with a cryptosystem whose private key includes the factorization of n. In the proof, the simulator must forge signatures without knowing the factorization of n. Note that the only step in which this knowledge is used is the extraction of the eth root of $xh^{H(x')}$ in order to compute

y, where e is chosen by the signature generation oracle. The simulator can therefore define the other elements of the public key so as to enable it to compute these roots. This can be accomplished in the following way. Suppose that the simulator expects to be required to compute at most ℓ signatures. In order to compute x and h for the public key it chooses in advance ℓ random values $e_1, \dots, e_\ell$, and two random values $x_0, h_0 \in QR_n$. It defines $x = x_0^{e_1 \cdots e_\ell} \bmod n$ and $h = h_0^{e_1 \cdots e_\ell} \bmod n$. It also chooses e' as a random prime of the length required by the original scheme. The public key is (n, h, x, e'). The key has the same distribution as in the original scheme, subject to the constraint that n is given.

Operation of the simulator: When the simulator is required to compute the ith signature, it chooses $e = e_i$. It then computes the signature in the usual way, except that y is computed as

$$(xh^{H(x')})^{1/e} = x_0^{e_1 \cdots e_{i-1} e_{i+1} \cdots e_\ell} (h_0^{e_1 \cdots e_{i-1} e_{i+1} \cdots e_\ell})^{H(x')} \bmod n.$$

This computation does not require knowledge of the factorization of n. The view of the adversary is identical to its view in interacting with the combined scheme.

4.3 The Gennaro-Halevi-Rabin Scheme

Gennaro, Halevi and Rabin proposed an efficient and stateless signature scheme secure against adaptive chosen-message attacks, based on the strong RSA assumption in the standard model [14], and on the use of a "chameleon" hash function. Although the result is similar to the Cramer-Shoup scheme, the construction is very different, and it is worthwhile to examine our approach in relation to it. The scheme operates in the following way. (We describe here the variant that does not require the random-oracle assumption):

Key generation: The public key is composed of a random RSA modulus n, a random value $s \in (\mathbf{Z}/n\mathbf{Z})^*$, and a hash function H. The private key is the factorization of n, namely primes p, q such that $pq = n$. It is also assumed that p, q are "safe", so that finding an odd integer that is not co-prime with $\phi(n)$ is as hard as factoring n.

Signature generation and verification: To sign a message m the signature algorithm first chooses a random R and hashes it together with m to get an exponent $e = H(R, m)$. It then uses p, q to compute $\sigma = s^{1/e} \bmod n$. The signature is (σ, R). To verify the signature the verifier computes $e = H(R, m)$ and verifies that $\sigma^e = s \bmod n$.

In order for the scheme to be secure in the standard model the hash function must satisfy several properties. For our purpose, it is only important to know that chameleon hash functions [15] can be used by the scheme. In particular, we use the fact that these functions have a trapdoor. Without knowledge of this trapdoor the function H is collision-intractable. If the trapdoor is known then, given m_1, R_1, m_2 it is easy to find an R_2 such that $H(R_1, m_1) = H(R_2, m_2)$.

Relation between keys: As in §4.2 the public key n is also the public key of the attacked cryptosystem, and therefore the simulator must forge signatures without knowing the private key, which is the factorization of n. We require that the public key s be random, as in the distribution of the keys of the original system, but we provide the simulator with additional useful information about s. We do this with a technique that is similar to the method used to prove the security of the scheme in [14]. The key s is chosen in the following way: Suppose that the simulator expects to be required to compute at most ℓ signatures. In order to compute s for the public key it chooses H, chooses in advance 2ℓ values $R'_1, m'_1, \dots, R'_\ell, m'_\ell$, and sets $e_i = H(R'_i, m'_i)$ for $1 \le i \le \ell$. It then chooses a random $r \in (\mathbf{Z}/n\mathbf{Z})^*$ and sets the public key to contain $s = r^{e_1 \cdots e_\ell}$ and the function H. The distributions of the public key generated by the simulator, and the public key in the original scheme, are different only in case $e_1 \cdots e_\ell$ is not co-prime to $\phi(n)$, but this event happens with negligible probability.

Signature simulation: When it is required to sign a message m_i the simulator uses the trapdoor of H to find a value R_i such that $H(R_i, m_i) = H(R'_i, m'_i) = e_i$. It computes $\sigma = r^{e_1 \cdots e_{i-1}, e_{i+1} \cdots e_\ell} \bmod n$ and publishes (σ, R_i) as the signature. The view of the adversary differs from its view in interacting with the combined scheme only if the simulator cannot find a collision in H, which happens with negligible probability.

4.4 PSS Signatures

In [3], Bellare and Rogaway described an RSA-based signature scheme for which the ability to forge signatures is tightly related to the ability to invert the RSA function, in the random-oracle model. The scheme operates as follows.

Key generation: An ordinary RSA key (n, e, d) is generated, with $n = pq$, p and q prime, e relatively prime to $\phi(n)$, and $ed \equiv 1 \bmod \phi(n)$. The modulus n is of length $k > k_0 + k_1$. The public key is (n, e) and the private key is d. The scheme also makes use of three hash functions, h, g_1, and g_2, modeled as random oracles in reasoning about the scheme's security.

Signature generation: To sign a message $m \in \{0,1\}^*$, choose $r \in_R \{0,1\}^{k_0}$; compute $w = h(m||r)$ and $r^* = g_1(w) \oplus r$; let $y = 0||w||r^*||g_2(w)$; and return $y^d \bmod n$ as the signature.

Signature verification: To verify $x \in (\mathbf{Z}/n\mathbf{Z})^*$ as a valid signature for a message m with respect to the public key (n, e), compute $y = x^e \bmod n$ and parse y as $b||w||r^*||\gamma$; compute $r = r^* \oplus g_1(w)$; and verify the conditions $b = 0$, $g_2(w) = \gamma$, and $h(m||r) = w$. The signature is valid iff all three of the conditions are satisfied.

Relation between keys: We show that an instance of this signature scheme with RSA modulus n may be combined with any other cryptosystem that uses n as part of its public key. As in §§4.2 and 4.3, our simulator must be able to forge signatures without knowing the factorization of n.

Signature simulation: The simulator is given a message m, for which it must compute a signature without being able to compute dth roots mod n. It does this by choosing a random point $x \in (\mathbf{Z}/n\mathbf{Z})^*$, and then choosing random-oracle responses appropriately so that $y = x^e \bmod n$ is a valid signature for m. The simulator begins by making sure that the first bit of y is 0; if not, then it chooses a new x at random, and repeats. The simulator parses y as $y = 0||w||r^*||\gamma$; chooses $r \in_R \{0,1\}^{k_0}$; and then makes the oracle-value assignments $h(m||r) = w$, $g_1(w) = r \oplus r^*$, and $g_2(w) = \gamma$. (As usual, and in order to make sure that the functions are consistent, the simulator must keep a list of previous input-output pairs for which it has provided answers for h, g_1 and g_2, and if any of these inputs occurs again it must choose a new random x and repeat the process again.)

4.5 El Gamal Signatures

Pointcheval and Stern [18] provide a version of the El Gamal signature scheme [12] which is secure against adaptive chosen-message attacks in the random-oracle model. The scheme is defined in the following way.

Key generation; The public key is (p, g, h), where p is prime, g is an element with $(\mathbf{Z}/p\mathbf{Z})^* = \langle g \rangle$, and $h \in (\mathbf{Z}/p\mathbf{Z})^*$. The private key is x such that $h = g^x \bmod p$. The system also uses a hash function $H : M \times (\mathbf{Z}/p\mathbf{Z})^* \to (\mathbf{Z}/(p-1)\mathbf{Z})$, where the message space is M.

Signature generation and verification: To sign a message $m \in M$, pick a random $k \in (\mathbf{Z}/p\mathbf{Z})^*$, compute $r = g^k \bmod p$, $\quad s = k^{-1}(H(m, r) - xr) \bmod (p-1)$, and output (r, s). (The only difference from the El Gamal signature scheme is that $g^{H(m,r)}$ replaces g^m.) To verify (r, s) as a signature for m with respect to (p, g, h), check whether $h^r r^s = g^{H(m,r)} \bmod p$.

Relation between keys: We consider an additional cryptosystem whose private key contains the discrete log x. Therefore, the only freedom that the simulator has is in defining the hash function H (modeled as a random oracle).

A signature simulator: The simulator is asked by the adversary $\mathcal{A}$ to compute signatures, and should do so without knowing $x = \log_g(h)$. The simulator takes advantage of the fact that since the hash function H is modeled as a random oracle, $\mathcal{A}$ must compute each value of H independently, and accepts any randomly chosen value as the output of H for any specific query. The simulator can therefore observe the queries that $\mathcal{A}$ makes to H, and can manipulate the output of H while being assured that this will not be detected by $\mathcal{A}$. The only remaining problem is that the signature requires $g^{H(m,r)}$ to have a specific value, and the simulator does not know how to extract discrete logs in order to set the output of H accordingly.

To overcome this problem and forge a signature for a message m with respect to the public key (p, g, h), the simulator chooses random $u, v \in (\mathbf{Z}/(p-1)\mathbf{Z})$ with $(v, p-1) = 1$, and computes $r = g^u h^v \bmod p$ and $s = -rv^{-1} \bmod (p-1)$. It also sets $H(m, r) = us \bmod (p-1)$. It outputs (r, s) as the signature. This output is a valid signature for m since $h^r r^s = h^r g^{us} h^{vs} = h^r g^{us} h^{v(-rv^{-1})} \bmod p$. (This "forgery" technique, adapted here for our simulation, is well known; see e.g. [16], §11.5.2, Note 11.66 (iii).)[1]

5 Acknowledgements

We wish to thank Xavier Serret-Avila and Marius Schilder for stimulating discussions that motivated this work.

References

[1] M. Bellare, A. Desai, D. Pointcheval and P. Rogaway, *Relations Among Notions of Security for Public-Key Encryption Schemes*, Adv. in Cryptology – Proc. of Crypto '98, LNCS 1462, pp. 26-45.

[2] M. Bellare and P. Rogaway, *Optimal Asymmetric Encryption* Adv. in Cryptology – Proc. of Eurocrypt '94, Springer-Verlag LNCS 950, pp. 92-111.

[3] M. Bellare and P. Rogaway, *The Exact Security of Digital Signatures: How to Sign with RSA and Rabin*, Adv. in Cryptology – Proc. of Eurocrypt '96, Springer-Verlag LNCS 1070, pp. 399-416.

[4] E. Biham, *New Types of Cryptanalytic Attacks Using Related Keys*, J. of Cryptology 7(4): 229-246 (1994).

[5] M. Blum, P. Feldman and S. Micali, *Non-Interactive Zero-Knowledge and Its Applications*, STOC 1988, 103-112.

[6] D. Boneh, *Simplified OAEP for the RSA and Rabin functions*, Adv. in Cryptology – Proc. of Crypto 2001.

[7] R. Cramer and V. Shoup, *A Practical Public Key Cryptosystem Provably Secure Against Adaptive Chosen Ciphertext Attack*, Adv. in Cryptology – Proc. of Crypto '98, Springer-Verlag LNCS 1462, 13-25.

[8] R. Cramer and V. Shoup, *Signature Schemes Based on the Strong RSA Assumption*, ACM Conference on Computer and Communications Security 1999, 46-51.

[9] G. Davida, *Chosen Signature Cryptanalysis of the RSA (MIT) Public Key Cryptosystem*, TR-CS-82-2, Dept. of EECS, Univ. of Wisconsin, Milwaukee, 1982.

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

[11] C. Dwork and M. Naor, *An Efficient Existentially Unforgeable Signature Scheme and Its Applications*, Journal of Cryptology 11(3), pp. 187-208 (1998).

[12] T. El Gamal, *A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms*, Adv. in Cryptology – Proc. of Crypto '84, Springer-Verlag LNCS 196, pp. 10-18.

[13] E. Fujisaki, T. Okamoto, D. Pointcheval and J. Stern, *RSA-OAEP is Secure under the RSA Assumption*, Adv. in Cryptology – Proc. of Crypto '2001.

[14] R. Gennaro, S. Halevi and Tal Rabin, *Secure Hash-and-Sign Signatures Without the Random Oracle*, Adv. in Cryptology – Proc. of Eurocrypt '99, Springer-Verlag LNCS 1592, pp. 123-139.

[1] The adversary may query the simulator for $H(m, r)$ before it asks the simulator to sign m. This happens with negligible probability, since r is defined by the simulator as a function of u, v, which are chosen at random. In order to take care of this event, the simulator keeps a list L of all the queries to H that were asked by the adversary. If the simulator is asked to sign m, and it happens to choose u, v that define an r such that $(m, r) \in L$, then it chooses new values (u, v).

[15] H. Krawczyk and T. Rabin, *Chameleon hash functions*, Theory of Cryptography Library: Record 98-10, 1998.

[16] A. Menezes, P. van Oorschot and S. Vanstone, **Handbook of Applied Cryptography**, CRC Press, October 1996.

[17] M. Naor and M. Yung, *Public-key Cryptosystems Provably Secure against Chosen Ciphertext Attacks*, STOC 1990, pp. 427-437.

[18] D. Pointcheval and J. Stern, *Security Proofs for Signature Schemes*, Adv. in Cryptology – Proc. of EUROCRYPT 1996, LNCS 1070, pp. 387-398.

[19] M. Rabin, *Digitalized Signatures and Public-Key Functions as Intractable as Factorization*, MIT/LCS/TR-212, 1979.

[20] C. Rackoff and D. Simon, *Noninteractive zero-knowledge proof of knowledge and chosen ciphertext attack*, Adv. in Cryptology – Proc. of Crypto '91, pp. 433-444.

[21] C.-P. Schnorr, *Efficient Signature Generation by Smart Cards*, J. of Crypt. 4(3), 161-174 (1991).

[22] V. Shoup, *Using hash functions as a hedge against chosen ciphertext attacks*, Adv. in Cryptology – Proc. of Eurocrypt '2000, LNCS 1807, pp. 275-288.

[23] V. Shoup, *OAEP Reconsidered*, Adv. in Cryptology – Proc. of Crypto 2001. A more complete version is available as: Cryptology ePrint Archive: Report 2000/060 (February 6, 2001).