Completely Non-Malleable Schemes

Marc Fischlin*

Institute for Theoretical Computer Science, ETH Zürich, Switzerland

marc.fischlin@inf.ethz.ch
http://www.fischlin.de/

Abstract An encryption scheme is non-malleable if the adversary cannot transform a ciphertext into one of a related message under the given public key. Although providing a very strong security property, some application scenarios like the recently proposed key-substitution attacks yet show the limitations of this notion. In such settings the adversary may have the power to transform the ciphertext *and* the given public key, possibly without knowing the corresponding secret key of her own public key. In this paper we therefore introduce the notion of completely non-malleable cryptographic schemes withstanding such attacks. We show that classical schemes like the well-known Cramer-Shoup DDH encryption scheme become indeed insecure against this stronger kind of attack, implying that the notion is a strict extension of chosen-ciphertext security. We also prove that, unless one puts further restrictions on the adversary's success goals, completely non-malleable schemes are hard to construct (as in the case of encryption) or even impossible (as in the case of signatures). Identifying the appropriate restrictions we then show how to modify well-known constructions like RSA-OAEP and Fiat-Shamir signatures yielding practical solutions for the problem in the random oracle model.

1 Introduction

According to the seminal paper by Dolev et al. [DDN00] an encryption scheme is called non-malleable if giving a ciphertext to an adversary does not significantly help this adversary to produce a ciphertext of a related message under the same public key. Analogous requirements can be formulated for other cryptographic primitives like signatures or commitments. While this definition of non-malleability is already quite strong and suffices in most settings it yet leaves open if there are cases where refined notions are needed and, if so, whether they can be achieved at all.

* This work was supported by the Emmy Noether Programme Fi 940/1-1 of the German Research Foundation (DFG). Part of this work done while visiting University of California, San Diego, USA.

An extended abstract appears in ICALP 2005
Lecture Notes in Computer Science, © Springer-Verlag

Motivation. A possible stronger definition of non-malleability, introduced here as complete non-malleability, basically allows the adversary to transform the public key as well. That is, in case of encryption the adversary may output a ciphertext of a related message *under an adversarial chosen public key*. For this, the adversary does not even need to know the matching secret key to the chosen public key.

Our initial interest in completely non-malleable schemes stems from the area of (regular) non-malleable commitments. Previous constructions of such non-malleable commitments usually require a common reference string (e.g., [CIO98,FF00,CKOS01,FF02]), or are rather theoretical in terms of efficiency [DDN00,Bar02]. Coming up with an efficient non-malleable commitment protocol in the plain model is still an open problem.

Early in cryptography it has been observed that efficient commitment schemes can be derived from encryption schemes. To commit, the sender creates a key pair and sends a ciphertext of the message together with the public key. To decommit, the sender transmits the message with the random bits used to create the ciphertext, or simply sends the secret key (if appropriate). Now, if the encryption scheme was *completely* non-malleable then the resulting commitment scheme in this basic construction would be non-malleable in the ordinary sense. And the derived commitment scheme would be non-interactive and would not rely on public parameters either.

In addition to the application to commitment schemes, it turns out that, recently, the problem of complete non-malleability also appeared in similar flavors in related areas like signatures or hash functions:

- Blake-Wilson and Menezes [BWM99] show how to deploy unknown key-share attacks to show weaknesses in the station-to-station key agreement protocol. In their case, the adversary is given a signature s for message m under some public verification key vk and her task is to find a different key pair (sk^*, vk^*) such that s is also a valid signature for m under vk^* (called key-substitution attack). The problem of unknown-key attacks has later been addressed more general in [MS04], showing that standard signature schemes like Schnorr signatures are in fact secure against this kind of attack. Here we show that the schemes are *insecure* with respect to the stronger security requirement of complete non-malleability where the adversary does not need to know the secret key. The schemes are therefore in principle subject to other attacks in the key-agreement setting.
- Kaliski [Kal02] discusses attacks on hash-and-sign signature schemes in which the adversary is allowed to replace the original hash function (description). Given a signature s for the hash value $\textsc{hash}(m)$ of a message the adversary tries to find a hash function $\textsc{hash}^*$ and message m^* such that $\textsc{hash}^*(m^*) = \textsc{hash}(m)$. In this case, s would also be a valid signature for $\textsc{hash}^*(m^*)$.
- In strong unforgeability attacks on signature schemes the adversary is allowed to query a signature oracle and is considered to be successful if she later forges a signature for a new message, but also if she creates a new signature for a previously signed message. This security notion is for ex-

ample important for blind signatures used as coins in e-cash systems: It should be infeasible to create another coin from a given coin, even for the same message. Such attacks can be easily cast in the framework of complete non-malleability.

Our Results. In this work we discuss the issue of complete non-malleability for public-key encryption and signatures. We first show that most of the well-known encryption and signature schemes fall prey to complete non-malleability attacks. Specifically, we propose attacks against the Cramer-Shoup DDH encryption scheme, RSA-OAEP and signatures of the Fiat-Shamir type like Schnorr signatures. This shows that the security notion of complete non-malleability is not covered by chosen-ciphertext security and by unforgeability against chosen-message attacks, respectively.

Then we give a formal framework for complete non-malleability of public-key encryption and signatures. There are two major differences to the basic definition of non-malleability. First, the adversary's goal in the definition of [DDN00] for encryption is to relate the original secret message m to a chosen message m^* via a relation $\mathsf{R}(m, m^*)$. Here we extend the relation to include the given public key pk. For message-only relations it remains for example unclear if it is easy to modify a ciphertext of some message m into a ciphertext of the related message $m^* = m + e$ under the same public key, where e is for example an RSA-exponent in the public key. We show that such attacks are feasible, at least for general schemes. Namely, we present a scheme which is non-malleable for relations over messages, but for which the adversary can easily produce a ciphertext c^* of a message m^* under pk such that a specific relation $\mathsf{R}(pk, m, m^*)$ is satisfied. We stress that the adversary does not even take advantage of the possibility to select her own public key for this attack.

Our separating example for relations $\mathsf{R}(m, m^*)$ over messages shows that (regular) non-malleable commitments constructed by means of encryption schemes in the common reference string model (as in [CKOS01]) may not provide adequate security for the classical Internet auction example. In the auction case the users' bids are encrypted with a public key published in the reference string. Now, an adversarial bidder may be able to transform such a sealed bid of an honest user into one which is related via this public key, and may thus overbid this user easily with a reasonably small amount (e.g., by $m^* = m + e$).

The second, and more significant extension of the [DDN00] framework for encryption is that the adversary now has the power to tamper the public key. Consequently, the relations now also range over the given public key pk, the adversarial chosen public key pk^* and, for sake of generality, also over adversary's ciphertext. Similarly, for signatures we let the relation include the given verification key vk, the adversarial key vk^*, message m^* and signature s^*.

Concerning constructions of completely non-malleable schemes, the bad news is that schemes *for general relations* are hard to derive or even impossible. We show that there are relations where complete non-malleability cannot be proven via black-box proofs for both encryption and signatures. Even worse, for more

complex relations we prove that completely non-malleable signature schemes do not exist at all.

On the positive side, we can show that practical schemes like RSA-OAEP and Fiat-Shamir signatures can be made completely non-malleable *in the random oracle model* (while the basic versions do not achieve this goal, not even in the random oracle model). Security holds for a broad class of relations which, roughly, excludes only such relations for which we are able to show our unconditional impossibility results. Also, our solutions are essentially as efficient as the original schemes, thus giving us complete non-malleability almost for free.

However, we remark that the completely non-malleable versions of the schemes above are proven secure in the random oracle model only. A closer look reveals why this model provides a useful countermeasure: Random oracles are by nature highly non-malleable constructs, because outputs of related inputs are completely uncorrelated and because all users in the system use the *same* hash function oracle as a common anchor. The advantage of giving security of these schemes in terms of complete non-malleability, even in the random oracle model, is that security now follows for a vast number of attacks, including key-substitution and strong-unforgeability attacks. That is, any attacks where the adversary's goal can be cast through such relations, provably fail; extra security proofs become obsolete. An interesting open question is whether there are secure schemes in the plain model for interesting relations or not.

Organization. To provide some intuition about the power of complete non-malleability attack we start with the attack on the Cramer-Shoup encryption scheme (and other schemes) in Section 2. We define completely non-malleable schemes formally in Section 3. We then given an informal overview over the results in Section 4. In Section 5 we discuss the separation with respect to relations over messages and those including the public key. Section 6 covers all negative results and the concluding Section 7 presents the positive examples in the random oracle model.

2 Attacks on Well-Known Schemes

We first provide some intuition and present attacks on schemes which are known to be secure against ordinary attacks. We show that the Cramer-Shoup encryption scheme, the RSA-OAEP encryption scheme and the Schnorr signature scheme all suffer from weaknesses with respect to complete non-malleability. We remark that all these schemes still satisfy their designated security goals like chosen-ciphertext security or unforgeability. Our attacks merely show that they do not withstand the stronger kind of attack.

For the examples in this section we do not give formal security definitions of complete non-malleability. For the moment, it suffices to keep in mind that the adversary's goal is to modify a given public key and a cryptogram (e.g., a ciphertext or a signature) into a related one.

2.1 Cramer-Shoup Encryption Scheme

The Cramer-Shoup encryption scheme [CS98] is semantically secure against adaptive chosen-ciphertext attacks under the decisional Diffie-Hellman assumption. It is thus also non-malleable (in the classical sense) with respect to such attacks.

Key Generation: The public key is given by the description of a group $\mathcal{G}_q$ of prime order q for which the decisional Diffie-Hellman problem is believed to be intractable, two random generators g_1, g_2 of this group as well as c, d and h where

$$c = g_1^{x_1} g_2^{x_2}, \qquad d = g_1^{y_1} g_2^{y_2}, \qquad h = g_1^{z_1} g_2^{z_2}$$

for random values $x_1, x_2, y_1, y_2, z_1, z_2 \leftarrow \mathbb{Z}_q$. The public key also contains a collision-intractable hash function H. The secret key is $(x_1, x_2, y_1, y_2, z_1, z_2)$.

Encryption: To encrypt a message $m \in \mathcal{G}_q$ pick a random $r \leftarrow \mathbb{Z}_q$ and compute

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

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

Decryption: To decrypt a ciphertext (u_1, u_2, e, v) compute $\alpha = H(u_1, u_2, e)$ and verify that $v = u_1^{x_1+\alpha y_1} u_2^{x_2+\alpha y_2}$. If so, then output $m = e/u_1^{z_1} u_2^{z_2}$.

The attack showing that the scheme fails to provide complete non-malleability now proceeds as follows. Given a public key $(\mathcal{G}, g_1, g_2, H, c, d, h)$ and a ciphertext (u_1, u_2, e, v) first recompute $\alpha = H(u_1, u_2, e)$. With overwhelming probability $\alpha \neq 0 \mod q$ and we can invert α in $\mathbb{Z}_q^*$; else, if a random ciphertext maps to 0 with noticeable probability, collisions for H could be found easily. Next compute

$$u_1^* = u_1^2, \quad u_2^* = u_2^2, \quad e^* = e^2, \quad \alpha^* = H(u_1^*, u_2^*, e^*), \quad v^* = v^{2\alpha^*/\alpha}$$

and finally prepare the public key as

$$c^* = c^{\alpha^*/\alpha}, \qquad d^* = d, \qquad h^* = h.$$

A simple calculation shows that

$$v^* = v^{2\alpha^*/\alpha} = c^{2r\alpha^*/\alpha} d^{2r\alpha\alpha^*/\alpha} = (c^*)^{2r} (d^*)^{2r\alpha^*}$$

Hence, the tuple (u_1^*, u_2^*, e^*, v^*) is a valid ciphertext of $m^* = m^2 \in \mathcal{G}$ under randomness $r^* = 2r \bmod q$ and public key $(\mathcal{G}, g_1, g_2, H, c^*, d^*, h^*)$.[1]

[1] At first glance it seems that replacing h by $h^* = h^a$ (or similar substitutions), and leaving the other ciphertext components untouched, would work as well. But then the adversary would encrypt a message $m^* = e/(h^*)^r = mh^{r(1-a)}$. This, however, would be a random message (over the choice of r) and would be thus unlikely to be related to m in a reasonable way.

The attack shows that the encryption scheme cannot be used as a non-malleable commitment scheme, as explained in the introduction. With this attack the adversary would be able to open her commitment correctly with $(m^2, 2r)$ after seeing the decommitment (m, r) of the original sender. Analogously, if the adversary is given the original secret key $(x_1, x_2, y_1, y_2, z_1, z_2)$ she can modify it to $(x_1\alpha^*/\alpha, x_2\alpha^*/\alpha, y_1, y_2, z_1, z_2)$.

We also point out that the attack does not depend on the primitives $\mathcal{G}_q$ and H, nor the generators g_1, g_2. Even if this part of the public key was provided by a trusted party as common parameters or if H was a random oracle, then the attack would still work. Neither relies the attack on any decryption requests passed to the original key holder.

2.2 RSA-OAEP

The RSA-OAEP encryption scheme has been proposed by Bellare and Rogaway in [BR95]. The scheme is widely used in practice in the RSA-PKCS version [RSA02]. It is known to be plaintext aware under the RSA assumption in the random oracle model, and also chosen-ciphertext secure in the random oracle model [FOPS01]. We assume that two random oracles G and H are available to all parties:

Key Generation: Generate an RSA modulus N and an RSA exponent e. The public key is given by (N, e) while the secret key is given by $d = e^{-1} \bmod \varphi(N)$.

Encryption: To encrypt a message m pick a random string r and compute

$$y = \big(m||0^k \oplus G(r)\big) \;||\; \big(r \oplus H(m||0^k \oplus G(r))\big) \in \mathbb{Z}_N^*.$$

Return $c = y^e \bmod N$.

Decryption: To decrypt $c \in \mathbb{Z}_N^*$ compute $y = c^d \bmod N$ and parse y as $(m||0^k \oplus G(r)) \;||\; (r \oplus H(m||0^k \oplus G(r)))$. Compute the hash value of the left part under H and xor it to the right part to retrieve r. Then compute the exclusive-or of the left part and $G(r)$. If the least significant k bits are all 0 then output m.

For the attack copy the public key but replace the exponent e by $3e$. Later, after seeing the ciphertext c, substitute c by $c^3 \bmod N$. Clearly, this is a valid encryption of the same message under a different public key.

We remark that, if one stipulates the RSA exponent to be of a special form like $e = 2^{16} + 1$ and has the verifier check this, then our attack fails of course. However, such a check is currently not intended, neither in the RSA-OAEP [BR95] nor in RSA-PKCS [RSA02].

2.3 Schnorr Signatures

The attack presented here against the Schnorr signatures scheme [Sch91] works, mutatis mutandis, also for other Fiat-Shamir like signature schemes. The original Schnorr scheme for random oracle H goes as follows:

Key Generation: Let $\mathcal{G}_q$ be again a group of prime order q. Assume that the discrete logarithm problem in this group is intractable, and let g be a generator of this group. The user holds a secret key $x \leftarrow \mathbb{Z}_q$ and the public key is given by $(\mathcal{G}_q, g, X)$ for $X = g^{-x}$.

Signing: To sign a message m the user picks $r \leftarrow \mathbb{Z}_q$, computes $R = g^r$, $c = H(R, m)$ and $y = r + cx \bmod q$. The user outputs (c, y).

Verification: To verify a signature (c, y) for message m the verifier checks that $c = H(X^c g^y, m)$.

For the attack we simply modify a given public key X to $X^* = Xg^{-x'}$ for some $x' \in \mathbb{Z}_q^*$. Then, a given signature (c, y) for some message m under public key X can be transformed into a valid signature $(c^*, y^*) = (c, y + cx' \bmod q)$ for the same message m under public key X^*. If the original user later reveals his secret key x then the adversary can claim $x^* = x + x' \bmod q$ as her secret key.

3 Definitions

In this section we define completely non-malleable public-key encryption and signature schemes. Our approach follows the line of Dolev at el. [DDN00] and also investigates the non-malleability question of an encryption or signature scheme merely with respect to itself. Achieving non-malleability between different schemes is in general impossible, even in the basic case.

3.1 Encryption

Discussion. An obvious problem with defining completely non-malleable encryption schemes lies in the adversary's possibility to choose her own public key and the uniqueness of ciphertexts. With a fake, yet valid-looking public key the adversary might be able to produce ciphertexts which can be decrypted ambiguously. We consider this to be a characteristic of the encryption scheme, and not an issue of complete non-malleability. Specifically, we allow the adversary to produce such phony keys if the scheme supports it, i.e., if one cannot distinguish good keys from fake ones. We note that, for the application to non-malleable commitments as explained in the introduction, verifying the validity of keys is for example necessary.

Relations. As mentioned in the introduction, regular non-malleability says that it is hard to transform a given ciphertext of message m into one of a related message m^* under the same key. There, related messages are designated according to an efficiently computable (probabilistic) algorithm R which basically takes the messages m and m^* as input.[2] But here we are interested in more general attacks where, as in the examples of non-malleable commitments or key-substitution attacks on signatures, finding a related public key pk^* or ciphertext c^* to the given key pk may be considered a success. Hence, we let the relations in general also depend on pk and pk^*, c^*.

To capture both the original definition of relations over messages only and the more general approach including public keys, we look at classes $\mathcal{R}$ of relations and define complete non-malleability with respect to such classes. The class for the basic definition then spans over relations $\mathsf{R}(pk, m, pk^*, m^*, c^*) = \mathsf{R}_0(m, m^*) \wedge pk = pk^*$, for example.

Message Distributions. We assume that the distribution of the user's message is determined according to some efficiently computable probabilistic algorithm M from some class $\mathcal{M}$. The message distribution M may depend on the given public key. Yet, we sometimes focus on distributions which are independent of the key, i.e., we consider the class $\mathcal{M}_{\text{pk-ind}}$ of distributions M such that $\mathsf{M}(pk) \equiv \mathsf{M}(pk')$ for any keys pk, pk'. In this case the distribution M may still depend on the security parameter, though.

Dolev at al. [DDN00] let the adversary and the simulator determine the message distribution after seeing the public key and having queried the decryption oracle in a preprocessing phase. This can be subsumed in our model by letting these two algorithms output some parameter μ before the ciphertext is created. Unless stated differently all our results, positive and negative ones, remain valid in the setting where the adversary and simulator select such values; yet, we usually do not include them here for sake of simplicity.

Attack Model. In the first stage of the actual attack the adversary $\mathcal{A}$ is given a public key pk and access to a decryption oracle $\text{DEC}(sk, \cdot)$, where $(sk, pk) \leftarrow \text{KGEN}(1^k)$ have been produced by the key generator. The adversary also gets a description of the relation R and the message distribution M. A message m is sampled according to the distribution $\mathsf{M}(pk) \in \mathcal{M}$ and encrypted under pk to ciphertext $c \leftarrow \text{ENC}(pk, m; r)$. The adversary starts the attack on the ciphertext c, the decryption oracle and some information about the message m in form of the value $h \leftarrow \mathsf{hist}(m)$ of an efficiently computable probabilistic function hist. This function can be formally regarded of part of the distribution M. The adversary finally outputs a public key pk^*, possibly for a different yet polynomially related security parameter, and a ciphertext c^*.

[2] The definition in [DDN00] lets the relations include another string chosen by the adversary, mainly to deal with the case of symmetric encryption schemes. All our positive and negative results for public-key encryption and signatures remain valid for this extension.

Let $\pi_{\text{enc}}(\mathcal{A}, \mathsf{M}, \mathsf{R})$ be the probability that $(pk, c) \neq (pk^*, c^*)$ and that there exists some m^*, r^* such that $c^* = \text{ENC}(pk^*, m^*; r^*)$ and $\mathsf{R}(pk, m, pk^*, m^*, c^*)$ for the relation R. We call this a related-ciphertext attack. Here, as usual for non-malleability definitions, R may implicitly depend on the encryption scheme itself and some security parameter. However, we do not demand that $m \neq m^*$; it suffices to produce a different key/ciphertext pair.

As explained in the introduction, the usage of the encryption scheme as a commitment may result in different attacks and success goals, e.g., the adversary may be obliged to actually open her ciphertext after seeing the opening of the original ciphertext. Therefore, let $\pi_{\text{open}}(\mathcal{A}, \mathsf{M}, \mathsf{R})$ denote the probability that $\mathcal{A}$ after the first stage, on input α^* and values m, r, also returns m^*, r^* such that $c^* = \text{ENC}(pk^*, m^*; r^*)$ and $\mathsf{R}(pk, m, pk^*, m^*, c^*) = 1$. This is called a related-opening attack. Write $\pi_{\text{sk-open}}(\mathcal{A}, \mathsf{M}, \mathsf{R})$ for the probability that $\mathcal{A}$ for input α^* and the secret key sk returns sk^* such that $\text{DEC}(sk^*, c^*) = m^*$ and $\mathsf{R}(pk, m, pk^*, m^*, c^*) = 1$ in a so-called related-key-opening attack. The three cases are described informally in the middle column in Figure 1.

	$\mathcal{A}$ gets pk, c, oracle $\text{DEC}(sk, \cdot)$ and ...	$\mathcal{S}$ gets pk [and possibly oracle $\text{DEC}(sk, \cdot)$] and...
$\pi_{\text{enc}}^{(\prime)}$	$\mathcal{A}$ outputs pk^*, c^*	$\mathcal{S}$ outputs pk', c', m', r'
$\pi_{\text{open}}^{(\prime)}$	$\mathcal{A}$ outputs pk^*, c^*, then m^*, r^* after m, r	$\mathcal{S}$ outputs pk', c', m', r'
$\pi_{\text{sk-open}}^{(\prime)}$	$\mathcal{A}$ outputs pk^*, c^*, then sk^* after sk	$\mathcal{S}$ outputs pk', c', m', r', sk'

Figure1. Overview of Attack and Simulation Modes for Encryption

Simulation Model. To capture the idea of the user's ciphertext not helping to produce a ciphertext of a related message we define a simulator $\mathcal{S}$ which is supposed to be as successful as the adversary but without seeing the ciphertext. $\mathcal{S}$ gets as input a public key pk and descriptions of the relation and the message distribution, but does not get access to a decryption oracle. Then, a message m is sampled according to $\mathsf{M}(pk)$ and algorithm $\mathcal{S}$ receives $h \leftarrow \mathsf{hist}(m)$ as input.

Depending on the adversary's attack mode, the simulator's task becomes increasingly challenging such that a successful simulator for a security level automatically constitutes a simulator for a lower level. Precisely, the simulator is supposed to output a key pk', a ciphertext c', a message m' and randomness r' (if the adversary runs a related-ciphertext or a related-opening attack),[3] and a key pk', a ciphertext c', a message m', a random string r' and a secret key sk' (if the adversary runs a related-key-opening attack). Again, see Figure 1 for an overview.

[3] For some of our negative results we use a milder requirement and let the simulator only output pk', c'. This even strengthens these hardness results.

Concerning the auxiliary power of the simulator there are two possibilities. One version is to give the simulator, like the adversary, additional access to the decryption oracle. We call this an *assisted* simulator. This reflects the approach that the simulator should have comparable power as the adversary. The other possibility is to deny the simulator access to DEC. We call such simulators *stand-alone* simulators. This approach follows the definition of [DDN00].

Although the definition with assisted simulators appears to be more intuitive at first, it is not clear that giving the simulator access to DEC captures the "right flavor" of complete non-malleability. The additional power may for example allow to prove schemes to be secure which are completely malleable in a natural sense. While this question has somewhat been settled for chosen-ciphertext security, where this additional power is acceptable, our separation of complete non-malleability from chosen-ciphertext security means that these arguments cannot be transfered without precautions. Instead, a conservative approach for designing schemes is therefore to rely on stand-alone simulators, as it suffices for our solutions in the random oracle model for example. We note that our impossibility results hold for both cases, although in a slightly weaker sense for assisted simulators.

Let both $\pi'_{\text{enc}}(\mathcal{S}, \mathsf{M}, \mathsf{R})$ and $\pi'_{\text{open}}(\mathcal{S}, \mathsf{M}, \mathsf{R})$ denote the probability that $c' = \text{ENC}(pk', m'; r')$ and that $\mathsf{R}(pk, m, pk', m', c') = 1$ in the first and second simulation experiment, respectively. Similarly, $\pi'_{\text{sk-open}}(\mathcal{S}, \mathsf{M}, \mathsf{R})$ stands for the probability that $c' = \text{ENC}(pk', m'; r')$, $m' = \text{DEC}(sk', c')$ and $\mathsf{R}(pk, m, pk', m', c') = 1$ in the third simulation experiment.

Definition 1. *A public-key encryption scheme is completely non-malleable (for stand-alone or assisted simulator) with respect to kind* $\in \{enc, open, sk\text{-}open\}$*, distribution class* $\mathcal{M}$ *and relation class* $\mathcal{R}$*, if for any adversary* $\mathcal{A}$ *there exists a (stand-alone or assisted) simulator* $\mathcal{S}$ *such that for any distribution* $\mathsf{M} \in \mathcal{M}$ *and any relation* $\mathsf{R} \in \mathcal{R}$ *the absolute difference* $|\pi_{kind}(\mathcal{A}, \mathsf{M}, \mathsf{R}) - \pi'_{kind}(\mathcal{S}, \mathsf{M}, \mathsf{R})|$ *is negligible.*

In the sequel, when speaking of completely non-malleable encryption schemes we refer to related-ciphertext attacks and $\pi_{\text{enc}}(\mathcal{A}, \mathsf{M}, \mathsf{R})$, $\pi'_{\text{enc}}(\mathcal{S}, \mathsf{M}, \mathsf{R})$. The definitions for completely non-malleable encryption (and signatures in the next section) can be extended in a straightforward way to the random oracle model.

3.2 Signatures

The attack scenario for completely non-malleable signature schemes resembles the setting of adaptive chosen-message attacks known from regular signature schemes.

Discussion. Defining the attack model for completely non-malleable signature schemed as outlined above, it seems that the adversary can always generate a new signature under a new public key, i.e., the adversary can naturally generate a new key pair and sign some message with the self-generated secret key. As

explained, this attack can be confined as in the example of unknown-key attacks [BWM99] where the adversary is supposed to find a matching key pair for a given message and a given signature. Here we do not restrict the adversary's goal in such a way. First, we do not want to give up generality and exclude certain application scenarios, e.g., signatures encrypted together with the message under a malleable encryption scheme, where the message is not known but the signature may still be transformable by permeating the malleable ciphertext. Second, if the adversary can trivially output a signature, i.e., without relying on the original signature, then this does not violate the idea of (complete) non-malleability and we should therefore be able to prove this formally as well.

Attack and Simulation Model. At the outset of the complete non-malleability attack the adversary $\mathcal{A}$ gets as input the description of the relation R and a verification key vk, generated together with the secret signing key sk by $\text{KGEN}(1^k)$. The adversary is then allowed to query a signature oracle $\text{SIG}(sk, \cdot)$ about messages of her choice. For definitional reasons we let the signature oracle prepend the verification key vk and the message m to each signature reply s for such a query. The adversary finally outputs some verification key vk^*, a message m^* and some signature s^*. Define $\pi_{\text{sig}}(\mathcal{A}, \mathsf{R})$ as the probability that s^* is a valid signature for m^* under vk^*, i.e., $\text{VF}(vk^*, m^*, s^*) = 1$, that (vk^*, m^*, s^*) is different from any previously given answer (vk, m, s) of the signature oracle, and that $\mathsf{R}(vk, vk^*, m^*, s^*)$ holds for relation R from the class $\mathcal{R}$.

The simulator only gets vk and the relation as input and is supposed to output a triple (vk', m', s') without having oracle access to $\text{SIG}(sk, \cdot)$. Let $\pi'_{\text{sig}}(\mathcal{S}, \mathsf{R})$ be the probability that s' is a valid signature for m' under vk' and that $\mathsf{R}(vk, vk', m', s')$ is satisfied. The attack and simulation model is outlined in Figure 2.

	$\mathcal{A}$ gets vk, oracle $\text{SIG}(sk, \cdot)$ and ...	$\mathcal{S}$ gets vk and ...
$\pi^{(\prime)}_{\text{sig}}$	$\mathcal{A}$ outputs vk^*, m^*, s^*	$\mathcal{S}$ outputs vk', m', s'

Figure2. Overview of Attack and Simulation Mode for Signatures

Similar to the encryption case one could also distinguish between stand-alone simulators (as defined here) and assisted simulators (which additionally get access to the signature oracle). In the latter case one would have to unorthodoxly extend the model to allow the adversary to ask for a "challenge signature" which the simulator is denied. We do not follow this approach here as our negative results would hold for this case as well, and our constructions in the random oracle already work for stand-alone simulators.

Security Definition. The idea is now to say that for any adversary there is a simulator such that the success probabilities differ only insignificantly. But with this definition a signature scheme could be completely non-malleable and yet be insecure in the sense of unforgeability, e.g., if it is easy to derive the secret key

from the verification key. Therefore, we also throw in the mild assumption that the signature scheme must be unforgeable under key-only attacks, i.e., it must be infeasible on input vk (but no signature oracle) to find some message together with a valid signature under vk.

Definition 2. *A signature scheme is completely non-malleable for relation class $\mathcal{R}$ if it is existentially unforgeable under key-only attacks and if for any adversary $\mathcal{A}$ there exists a simulator $\mathcal{S}$ such that for any relation $\mathsf{R} \in \mathcal{R}$ the absolute difference $|\pi_{sig}(\mathcal{A}, \mathsf{R}) - \pi'_{sig}(\mathcal{S}, \mathsf{R})|$ is negligible.*

Implications. We briefly discuss some consequences of the definition, showing that the definition is powerful to reflect the notions of strong unforgeability (i.e., where the adversary is also considered victorious if she finds a new signature under the original verification key to a message previously signed by the signature oracle) or key-substitution attacks (where the adversary tries to find another key vk^* to a valid triple vk, m, s), both under adaptive chosen-message attacks. For this, let $\mathsf{R}_{\text{str-unf}}(vk, vk^*, m^*, s^*)$ be the relation such that $\mathsf{R}_{\text{str-unf}}(vk, vk^*, m^*, s^*) = 1$ iff $vk = vk^*$; let $\mathsf{R}_{\text{key-sub}}$ be the relation such that $\mathsf{R}_{\text{key-sub}}(vk, vk^*, m^*, s^*) = 1$ iff $\text{VF}(vk, m^*, s^*) = 1$. Note that a successful attack also requires $\text{VF}(vk^*, m^*, s^*) = 1$ by definition.

Proposition 1. *Let $(\text{KGEN}, \text{SIG}, \text{VF})$ be a signature scheme which is completely non-malleable with respect to $\mathcal{R} \ni \mathsf{R}_{str\text{-}unf}$. Then the scheme is strongly unforgeable under adaptive chosen-message attacks.*

Proof. Assume towards contradiction that there is an adversary $\mathcal{A}$ that executes a chosen-message attack and refutes the strong unforgeability requirement. On input vk and oracle access $\text{SIG}(sk, \cdot)$ adversary $\mathcal{A}$ returns with noticeable probability a forgery (m^*, s^*) under key vk such that $\text{SIG}(sk, \cdot)$ has never answered m^* with s^*. Then $\mathcal{A}$ in particular mounts a complete non-malleability attack for relation $\mathsf{R}_{\text{str-unf}}$ in $\mathcal{R}$ and thus succeeds with noticeable probability $\pi_{\text{sig}}(\mathcal{A}, \mathsf{R}_{\text{str-unf}})$. Because the scheme is completely non-malleable there is a simulator $\mathcal{S}$ that outputs a valid signature tuple (vk', m', s') where $vk' = vk$ with noticeable probability, too. But $\mathcal{S}$ is only given vk and does not get access to a signature oracle. Hence, $\mathcal{S}$ forges with noticeable probability a signature in a key-only attack, refuting the presumption about the security of the signature scheme against this kind of attack. □

Proposition 2. *Let $(\text{KGEN}, \text{SIG}, \text{VF})$ be a signature scheme which is completely non-malleable with respect to $\mathcal{R} \ni \mathsf{R}_{key\text{-}sub}$. Then the scheme is secure against key-substitution attacks.*

Proof. The proof is almost identical to the previous one. Only this time consider the relation $\mathsf{R}_{\text{key-sub}}$. Again, if there was an adversary running a key-substitution attack with noticeable success probability $\pi_{\text{sig}}(\mathcal{A}, \mathsf{R}_{\text{key-sub}})$, then there would be a simulator $\mathcal{S}$ returning (vk', m', s') such that $\text{VF}(vk, m', s') = 1$ with noticeable probability, too. But $\mathcal{S}$ would not have access to a signature oracle and would therefore forge a signature s' for message m' under vk in a key-only attack. □

3.3 Extensions to the Random Oracle Model

The definitions for completely non-malleable encryption and signature can be extended to the random oracle model. In this case, a random function H is chosen at the outset and all parties, including encryption and decryption algorithm, or signer and verifier, adversary, simulator etc. have oracle access to this function. Even the relation and the message distribution may contain oracle queries. We note the simulator $\mathcal{S}$ must also succeed with respect to the *given* oracle H, i.e., the simulator does not have the possibility to choose oracle values. Such oracles are called non-programmable [Nie02].

4 Summary of Results

In this section we summarize our (positive and negative) results. For better comprehensibility the results are stated in an informal way. The formal results and technical details can be found in Sections 5, 6 and 7.

Regular Non-Malleability and Relations over Messages Only. We show that extending the relations in the definition of [DDN00] for regular non-malleability, i.e., where the adversary does not tamper the public key, to include the given public key pk (in addition to the messages m, m^*) can be fatal to security:

Theorem 1 (informal). *There is an encryption scheme which is non-malleable with respect to $\mathcal{R}_{msg} = \{R(m, m^*)\}$ but which is malleable with respect to some relation $\mathsf{R}_{pk}(pk, m, m^*)$.*

Hardness of Constructions for General Relations. Here we discuss our negative results for constructions of completely non-malleable schemes where, in contrast to the previous case, the adversary is allowed to output another key pk^*. We show that there are relations for which completely non-malleable schemes are hard to construct. Although we prove this result for a specific set of "bad" relations, we note that the implication carries over to any class where such relations can be "somehow embedded" in relations of the class.

Theorem 2 (informal). *Public-key encryption schemes which are completely non-malleable according to black-box stand-alone simulators and general relations, do not exist.*

Note that the previous theorem assumes that the simulator is stand-alone. For assisted simulators, which are granted access to DEC, we can show the same result for relations which are efficiently computable relative to an oracle. We note that the black-box simulator does not have access to this oracle directly, but only through the relation. This corresponds to the case that the simulator can efficiently compute the relation (via black-box access) but is denied the description of the relation.

Theorem 3 (informal). *Public-key encryption schemes which are completely non-malleable according to black-box assisted simulators and general relations (relative to an oracle), do not exist.*

The results about encryption easily transfers to signatures:

Proposition 3 (informal). *Signature schemes which are completely non-malleable according to black-box simulations for general relations, do not exist.*

Yet, for signatures we can show that completely non-malleable systems for general relations are impossible at all, even when allowing non-black-box constructions or if the simulator depends on the relation.

Theorem 4 (informal). *There do not exist completely non-malleable signature schemes with respect to general relations.*

Constructions in the Random Oracle Model. On the positive side, solutions in the random oracle for completely non-malleable schemes exist. And while OAEP encryption [BR95] and Fiat-Shamir signatures [FS86] provably do not have this property, slight variations of these schemes work. The basic idea to simply include the public encryption or signature key, respectively, to each hash function evaluation. We append the term "with public-key hashing" to such modified schemes:

Proposition 4 (informal). *RSA-OAEP with public-key hashing is completely non-malleable with respect to stand-alone simulators and any relations, in the random oracle model.*

A similar result holds for Fiat-Shamir signatures:

Proposition 5 (informal). *Fiat-Shamir signatures with public-key hashing are completely non-malleable with respect to general relations (except for essentially those relations, for which the unconditional impossibility results of Theorem 4 holds), in the random oracle model.*

In both cases the proofs rely on the original results [BR95,FOPS01,PS00] about the security against regular chosen-ciphertext attacks and chosen-message attacks.

5 Regular Non-Malleability and Relations Over Messages Only

In this section we present an encryption scheme which is malleable in an intuitive sense, but which allows a non-malleability proof if the relations $\mathsf{R}_{\mathrm{msg}}(pk, m, m^*) = \mathsf{R}(m, m^*)$ in the class $\mathcal{R}_{\mathrm{msg}}$ are only based on the messages and not on the given public key. The example here follows a related construction in [DDN00] to separate semantic security and non-malleability in the case of chosen-ciphertext preprocessing attacks where the adversary can only access the decryption oracle before receiving the challenge ciphertext. Indeed, our scheme is only secure with respect to such attacks.

Although our counterexample holds for preprocessing attacks we can still show that even in the setting where decryption queries are allowed after receiving

the challenge, the problem remains hard. That is, in Section 6.5 we briefly discuss that our impossibility result about black-box construction of completely non-malleable schemes (Section 6) carries over to regular non-malleable schemes and to relations $\mathsf{R}(pk, m, m^*)$, i.e., if the adversary is not allowed to maul the public key but if the relations depend on the given public key. Again, if the relation solely depend on the messages then we know black-box constructions to achieve basic non-malleability in this scenario.

To prove our separation result we start with the non-malleable encryption scheme of Dolev et al. [DDN00]. Most of the details of the encryption scheme are irrelevant to our discussion here. We merely remark that this scheme has a special simulator $\mathcal{S}_0$ which extracts the message m_0^* of the adversary $\mathcal{A}_0$ by running $\mathcal{A}_0$ on a prepared public key pk_0' (different from the given public key pk_0) and a fake ciphertext c_0' of a random message. With the chosen key pk_0' the simulator is able to decrypt ciphertexts c_0^* different from c_0', although the latter may fail with some negligible probability.

Modify the given encryption scheme as follows. We presume that only messages of n bits are encrypted. The key generation algorithm now also appends a random string $r \leftarrow \{0,1\}^n$ to the original keys pk_0, sk_0 and outputs $pk = pk_0||r$ and $sk = sk_0||r$. To encrypt a message $m \in \{0,1\}^n$ pick a random bit b and encrypt $m_0 = m$ if $b = 0$ and $m_0 = m \oplus r$ if $b = 1$ under the original scheme and public key pk_0. Given the ciphertext c_0 append the bit b in clear and return $c = c_0||b$. To decrypt a ciphertext $c = c_0||b$ decrypt c_0 under the original scheme with sk_0 to get m_0 and return $m = m_0$ if $b = 0$ and $m = m_0 \oplus r$ if $b = 1$.

For the following, we call a scheme *strongly malleable with respect to* R *in a key-only attack* if there is a probabilistic polynomial-time adversary $\mathcal{A}$ such that $\mathcal{A}$ on input any public key pk and any ciphertext c of some message m always outputs a ciphertext c^* of a message m^* such that $\mathsf{R}(pk, m, m^*)$ is satisfied. The specific relation for which we prove strong non-malleability is $\mathsf{R}_{\mathrm{pk}}(pk, m, m^*) = 1$ if $m^* = m \oplus r$ and $pk = pk_0||r$ for some pk_0 and $r \in \{0,1\}^n$.

We allow arbitrary message distributions as long as they are independent of the given public key (as explained in Section 3.1). Recall that such distributions are denoted by $\mathcal{M}_{\text{pk-ind}}$.

Theorem 5. *Assume trapdoor permutations exist. Then there is an encryption scheme which is non-malleable with respect to $\mathcal{R}_{msg}$ and $\mathcal{M}_{pk\text{-}ind}$ in chosen-ciphertext preprocessing attacks, but which is strongly malleable with respect to R_{pk} and $\mathcal{M}_{pk\text{-}ind}$ in key-only attacks.*

Proof. If trapdoor permutations exist then the constructed encryption scheme above exist. We first show non-malleability with respect to relations over messages.

For any adversary $\mathcal{A}$ consider the following simulator $\mathcal{S}$ which uses the simulator $\mathcal{S}_0$ of the original scheme as a subroutine. $\mathcal{S}$ gets as input a key $pk = pk_0||r$. It runs $\mathcal{S}_0$ to prepare a key pk_0', and starts a simulation of the adversary on $pk' = pk_0'||r$. Any decryption query $c = c_0||b$ in the preprocessing phase in answered by chopping of the bit b and handing c_0 to $\mathcal{S}_0$. This simulator returns a

message m_0 and we forward $m = m_0$ for $b = 0$ and $m = m_0 \oplus r$ if $b = 1$ to the adversary. We neglect the small error involved in this decryption runs.

Then, for distribution $\mathsf{M} \in \mathcal{M}_{\text{pk-ind}}$ a message m is sampled and $\mathsf{hist}(m)$ is forwarded to simulator who hands this information to the adversary. $\mathcal{S}$ also runs $\mathcal{S}_0$ to get a ciphertext c'_0 of some random message. $\mathcal{S}$ picks a bit b' and hands $c' = c'_0||b'$ to the adversary. Except with negligible probability this challenge ciphertext c' is different from each of the previous decryption queries.

The adversary finally outputs $c^* = c^*_0||b^*$. If $c^*_0 = c'_0$ (and therefore $b^* \neq b'$) then the simulator simply returns a random message $m' \in \{0,1\}^n$. If $c^*_0 \neq c'_0$ then the simulator invokes $\mathcal{S}_0$ to get the message m_0. This fails only with negligible probability. The simulator then outputs $m' = m_0$ if $b^* = 0$ and $m' = m_0 \oplus r$ if $b^* = 1$.

Under the condition that $c^*_0 = c'_0$, i.e., the adversary has flipped the appended bit to encrypt message $m^* = m \oplus r$, the simulator $\mathcal{S}$ outputs a random message m'. In this case, $\text{Prob}_{m'}[\mathsf{R}(m, m')] = \text{Prob}_r[\mathsf{R}(m, m \oplus r)]$. On the other hand, given that $c^*_0 \neq c'_0$ the derived message m' equals the adversary's message m^* encrypted in c^*, and thus $\text{Prob}[\mathsf{R}(m, m')] = \text{Prob}[\mathsf{R}(m, m^*)]$. In both cases the simulator therefore outputs a related message with almost the same probability that the adversary succeeds. The scheme is therefore non-malleable with respect to relations over messages.

Why is this scheme strongly malleable for relations spanning over the public key? Clearly, an adversary $\mathcal{A}$ that gets a public key $pk = pk_0||r$ and a ciphertext $c = c_0||b$ of an unknown message m can output the ciphertext $c^* = c_0||b \oplus 1$ of the message $m^* = m \oplus r$. This message is related to the original message in a non-trivial way via the given public key, of course. Hence, the scheme is strongly malleable with respect to relation R_{pk}. □

6 Hardness of Constructions for General Relations

In this section we discuss the hardness of constructions of completely non-malleable schemes *for general relation classes*. For encryption schemes we show that for certain relations over pk, pk^* no solutions relying on black-box simulators can exist. This argument can be easily transfered to similar relations over pk, m^* as well as to signatures. For signatures we can even prove unconditionally that completely non-malleable schemes do not exist, although this result relies on more complex relations which are more likely to be excluded by restricted classes $\mathcal{R}$.

We start by defining black-box simulation. While this formalization is necessary to state the impossibility results precisely, the definition follows the usual setting and the reader may initially skip this part.

6.1 Black-Box Simulations

The basic idea of black-box simulations for interacting machines [GMR89] is that simulator can only exploit the adversary's program and the relation's pro-

gram via the input/output behavior (but the simulator can for example rewind executions)

Machine Model. We assume that the simulator $\mathcal{S}$ is an interactive Turing machine [GMR89] connected by individual communications tapes to q^2 other interactive Turing machines $\mathcal{A}_{i,j}$, $i, j = 1, 2, \ldots, q$, for some polynomial $q = q(k)$ in the security parameter. At the outset the random tape of $\mathcal{S}$ is filled up with randomness σ. Also, q random strings α_i are chosen and α_i is written on all q tapes of machines $\mathcal{A}_{i,1}, \ldots, \mathcal{A}_{i,q}$, i.e., all machines with index i are initialized with the same random string, yet for different indices i the random tapes are independent.

Some input I is written to the input tape of $\mathcal{S}$ which is then woken up. Repeat the following steps until the simulator enters a halting state and writes some output O on its output tape. $\mathcal{S}$ first performs some internal computation. Then it writes a message on the communication tape of some machine $\mathcal{A}_{i,j}$, unless it is the first time $\mathcal{S}$ communicates with machine $\mathcal{A}_{i,j}$; in this case $\mathcal{S}$ must write some input $I_{i,j}$ on the input tape of $\mathcal{A}_{i,j}$ and a special symbol on the communication tape to start machine $\mathcal{A}_{i,j}$. In both cases, $\mathcal{S}$ goes idle and the corresponding machine $\mathcal{A}_{i,j}$ wakes up.

Once activated, $\mathcal{A}_{i,j}$ internally computes the next message to be written on the shared communication tape with $\mathcal{S}$ and returns to an idle state again after having written the message. If $\mathcal{A}_{i,j}$ halts instead and writes something on the output tape, then this message is written on the communication tape with a special symbol signalizing $\mathcal{S}$ that $\mathcal{A}_{i,j}$ has stopped. $\mathcal{S}$ eventually wakes up again and continues the computation.

Our impossibility result also relies on black-box relations, i.e., instead of getting a description of the relation as input, we assume that another interactive machine R is connected to $\mathcal{S}$ via a communication tape. This machine is initialized with a random tape ρ at the beginning of the execution and then the simulator can interact with this machine as with $\mathcal{A}_{i,j}$ through the communication tape.

We remark that the running time of $\mathcal{S}$ is the sum of all steps of all machines, i.e., we charge the simulator also for the steps of machines $\mathcal{A}_{i,j}$ and R. Unless stated differently, $\mathcal{S}$ is polynomially bounded. We further note that we can extend the setting to non-uniform algorithms such that each machine gets an auxiliary input on a special tape (where $\mathcal{A}_{i,1}, \ldots, \mathcal{A}_{i,q}$ all get the same auxiliary information).

Having access to q identical copies $\mathcal{A}_{i,1}, \ldots, \mathcal{A}_{i,q}$ allows the simulator to mount reset attacks, despite the fact that interactive Turing machines keep state. The simulator can somewhat reset the machine if it re-runs an execution with some copy $\mathcal{A}_{i,j}$ by giving the same input and answers to another copy of $\mathcal{A}_{i,k}$ up to a certain point.

Black-Box Simulations. Consider an encryption or signature scheme and an adversary $\mathcal{A}$ with oracle access to a decryption or signature oracle. We say that $\mathcal{S}$ *runs a black-box simulation of* $\mathcal{A}$ *and* R *on input* I if $\mathcal{S}$ gets I as input, if R

efficiently computes the function R for data written on the communication tape (where the result is also written on the communication type), and if all machines $\mathcal{A}_{i,j}$ run $\mathcal{A}$'s polynomial-time program with the only exception that each of $\mathcal{A}$'s oracle query is written on the communication tape instead, and that each answer of the oracle is read from this communication tape instead. We write $\mathcal{S}^{\mathcal{A},\mathsf{R}}(I)$ for both the actual process and the output O of $\mathcal{S}$ in such an execution.

We say that an encryption scheme is *completely non-malleable according to black-box simulations for distribution class $\mathcal{M}$ and relation class $\mathcal{R}$* if there is a probabilistic polynomial-time interactive Turing machine $\mathcal{S}$ such that the following holds. For any adversary $\mathcal{A}$ mounting a related-ciphertext attack, any distribution $\mathsf{M} \in \mathcal{M}$, any relation $\mathsf{R} \in \mathcal{R}$, the probability that $\mathcal{S}^{\mathcal{A},\mathsf{R}}(pk)$ outputs (pk', c', m', r') such that $c' = \textsc{enc}(pk', m'; r')$ and that $\mathsf{R}(pk, m, pk', m', c') = 1$, is negligibly close to $\pi_{\text{enc}}(\mathcal{A}, \mathsf{M}, \mathsf{R})$. Here the probability is taken over the choice of sk, pk (generated by $\textsc{kgen}$), m and the random tapes $\sigma, \alpha_1, \ldots, \alpha_q, \rho$.

Analogously, we call a signature scheme *completely non-malleable according to black-box simulations for relation class $\mathcal{R}$* if there is a probabilistic polynomial-time interactive Turing machine $\mathcal{S}$ such that the following holds. For any adversary $\mathcal{A}$ and any relation $\mathsf{R} \in \mathcal{R}$, the probability that $\mathcal{S}^{\mathcal{A},\mathsf{R}}(vk)$ outputs (vk', s', m') such that $\textsc{vf}(vk', s', m') = 1$ and $\mathsf{R}(vk, vk', s', m') = 1$, is negligibly close to $\pi_{\text{sig}}(\mathcal{A}, \mathsf{R})$.

For encryption the model allows furthermore to distinguish between stand-alone and assisted simulators; in the latter case the simulator is also given access to the decryption oracle.

6.2 Encryption and Stand-Alone Simulators

For our negative result we switch to the non-uniform machine model, i.e., we assume that the adversary $\mathcal{A}$, the simulator $\mathcal{S}$ and the relation R are all given by a family of polynomial-size circuits. For the adversary the auxiliary information consists of a sequence $\boldsymbol{key}$ of keys $key \in \{0,1\}^k$ of a pseudorandom function $\textsc{prf}$. The pseudorandom function $\textsc{prf}$ takes a key $key \in \{0,1\}^k$ and any public key pk generated by $\textsc{kgen}(1^k)$ as input. Varying over all such key sequences gives us an infinite set of adversaries. We also index the relations by such key sequences.

The pseudorandom function $\textsc{prf}$ takes a key $key \in \{0,1\}^k$ and any public key pk generated by $\textsc{kgen}(1^k)$ as input. Since the function is pseudorandom for any polynomial-size family of circuits the following holds. The circuit for parameter k is given oracle access to a function $\textsc{prf}(key, \cdot)$ for a random key $key \leftarrow \{0,1\}^k$ and is allowed to adaptively query this oracle for values of its choice. Then the circuit outputs a value pk not among those queries, and is either given a random string as reply or the value $\textsc{prf}(key, pk)$. The circuit may continue to query the oracle about different inputs and finally outputs a guess bit. The advantage is the absolute difference between the probabilities that the circuit outputs 1 in either case. This advantage is known to be negligible for pseudorandom functions.

Now we define the adversary and the relation in detail. The message distribution is arbitrary and we assume that hist returns the empty output. On input $\overline{pk}$, possibly different from the simulator's input pk, and oracle access to

$\text{DEC}(\overline{sk}, \cdot)$ the adversary $\mathcal{A}_{\boldsymbol{key}}$ for security parameter k and key $key \in \{0,1\}^k$ works as follows:

- $\mathcal{A}_{\boldsymbol{key}}$ computes $(m_{\text{test}}, r_{\text{test}}, \omega) = \text{PRF}(key, \overline{pk})$ and encrypts the first part to $c_{\text{test}} = \text{ENC}(\overline{pk}, m_{\text{test}}; r_{\text{test}})$ and passes c_{test} to the decryption oracle.
- If the oracle's answers is different from m_{test} then $\mathcal{A}_{\boldsymbol{key}}$ halts with output $\perp$.
- Else $\mathcal{A}_{\boldsymbol{key}}$ computes $(sk^*, pk^*) = \text{KGEN}(1^k, \omega)$, i.e., the key generator's output for (pseudo)randomness ω.
- $\mathcal{A}_{\boldsymbol{key}}$ computes $c^* \leftarrow \text{ENC}(pk^*, 0^k)$ and outputs (pk^*, c^*).

The relation $\mathsf{R}_{\boldsymbol{key}}$ for input (pk, m, pk^*, c^*, m^*) returns 1 if and only if the pseudorandom function $\text{PRF}(key, pk)$ returns $(m_{\text{test}}, r_{\text{test}}, \omega)$ such that $\text{KGEN}(1^k, \omega)$ yields the public key pk^* (and an arbitrary secret key). In particular, $\mathsf{R}_{\boldsymbol{key}}$ only depends on the keys pk, pk^*. One can easily modify the construction to get relations depending on pk and m^* only (as done for instance for the result in Section 6.5).

For the proof we also require that the scheme is semantically secure (against key-only attacks) with respect to the uniform distribution M_{unif} on messages of superlogarithmic length, and for the equality relation $\mathsf{R}_{\text{msg-eq}}(pk, m, pk^*, m^*, c^*) = 1$ iff $m = m^*$. Complete non-malleability for this distribution and this relation already implies semantic security against key-only attacks.

Theorem 6. *Encryption schemes which are completely non-malleable according to black-box stand-alone simulators for $\mathcal{M} \ni \mathsf{M}_{unif}$ and $\mathcal{R} \supseteq \{\mathsf{R}_{\boldsymbol{key}} | \boldsymbol{key}\} \cup \{\mathsf{R}_{msg\text{-}eq}\}$ do not exist.*

The theorem even holds if the simulator is only supposed to output pk', c' (but not m', r' as required by Definition 1). The intuition of the construction is that the black-box simulator is not able to answer the test ciphertext for the given public key pk, but only on self-chosen keys $\overline{pk}$ different than pk. Thus, the simulator never gets to see the adversary's key pk^* for pk, and predicting this key otherwise is infeasible because of the pseudorandomness of PRF.

Proof. Assume that there exists a completely non-malleable encryption scheme which is provably secure via a (non-uniform) black-box simulator $\mathcal{S}$. The existence of a secure encryption scheme implies that there are pseudorandom functions [IL89,HILL99,GGM86] and therefore adversaries $\mathcal{A}_{\boldsymbol{key}}$ and relations $\mathsf{R}_{\boldsymbol{key}}$ as described above exist. Note that any adversary $\mathcal{A}_{\boldsymbol{key}}$ outputs a related input with probability 1 in such an attack for relation $\mathsf{R}_{\boldsymbol{key}}$ for the same key sequence.

By assumption the black-box simulator approximates the success probability of *any* adversary well. Here we look at the behavior of this simulator when run on the adversary $\mathcal{A}_{\boldsymbol{key}}$ and the relation $\mathsf{R}_{\boldsymbol{key}}$ for the same key sequence $\boldsymbol{key}$, but this time each key *key* in the sequence is chosen at random. Then the simulator must also succeed with respect to such an adversary and relation, specified by the sequence of random keys.

We first claim that the probability that there is some execution $\mathcal{A}_{i,j}$ for input pk for which the black-box simulator is able to return m_{test} to the adversary, is negligible. Presume for sake of contradiction that this probability is noticeable.

We show that we can then use the simulator to distinguish the value of the pseudorandom function with noticeable advantage from a random value. The description below can be easily turned into a corresponding circuit.

We are given oracle access to a pseudorandom function $\text{PRF}(key, \cdot)$ for a random secret key key. We first generate a random key pair $(sk, pk) \leftarrow \text{KGEN}(1^k)$ and pass pk as the challenge to the function oracle. We receive either a truly random values $(m_{\text{test},0}, r_0)$ or the pseudorandom values $(m_{\text{test},0}, r_0)$; in both cases ω_0 is considered to be pseudorandom.[4]

Next, we pass pk to the simulator and use the pseudorandom function oracle to run an emulation of the black-box simulator by supplying all actions of the (copies of the) adversary $\mathcal{A}_{\boldsymbol{key}}$ and $\mathsf{R}_{\boldsymbol{key}}$. To do so we perfectly imitate the behavior of the adversary and the relation for random tapes $\alpha_1, \ldots, \alpha_q, \rho$ except for two points: First, if in some execution the machine is supposed to evaluate $\text{PRF}(key, \cdot)$ for some $\overline{pk} \neq pk$ then we simply query the given pseudorandom function oracle instead. Second, if a machine is supposed to evaluate the function at $\overline{pk} = pk$ then we use the given values $(m_{\text{test},0}, r_0, \omega_0)$ instead of querying $\text{PRF}(key, \cdot)$ about pk. We continue the simulation until the simulator eventually stops and returns pk', c' (and possibly m', r').

If the probability that the simulator answers a decryption query c_{test} correctly for an execution involving pk drops significantly when the values $(m_{\text{test},0}, r_0)$ are truly random, then it is easy to derive a distinguisher for the pseudorandom function. Namely, return 1 if and only if the simulator returns such a valid decryption. For pseudorandom values $(m_{\text{test},0}, r_0)$ this would happen with noticeable probability by assumption, whereas for random $(m_{\text{test},0}, r_0)$ this would then occur with negligible probability only.

In conclusion, if the probability that the simulator decrypts correctly c_{test} for pseudorandom values is noticeable, then it must also be noticeable if we use random values. This, however, would contradict the semantic security for distribution M_{unif} and relation $\mathsf{R}_{\text{msg-eq}}$. Specifically, given a public key pk and ciphertext c of a random message, run the emulation above for pk and guess the execution (i, j) for which the simulator first returns such a valid decryption. In this execution return $c_{\text{test}} = c$ on the first initialization of $\mathcal{A}_{i,j}$. Stop and return the simulator's decrypted message for this execution if, at some point, $\mathcal{S}$ returns a message in this execution. Return a random message otherwise. Apparently, the probability of satisfying the relation $\mathsf{R}_{\text{msg-eq}}$ in this case would be noticeably larger than if one is not given the ciphertext c at all (in which case equality holds with negligible probability only for the unknown random message with superlogarithmic length). This refutes semantic security of the scheme.

From now on we can condition on the event that the simulator never returns a valid decryption for a run on pk. Under this condition the probability (over the choice of the pseudorandom function key) that the simulator at some point runs (either at the end or during the execution) the relation algorithm on some

[4] Usually, the output is either completely random or pseudorandom. It is not hard to see that for pseudorandom functions it is also hard to distinguish outputs which are partially random from entirely pseudorandom outputs.

pk' such that $\text{KGEN}(1^k, \omega_0) = pk'$, is negligible. If this probability was noticeable then it would remain noticeable if the value pseudorandom function $\text{PRF}(key, \cdot)$ would return $(m_{\text{test},0}, r_0, \omega_0)$ for a truly random ω_0 instead. Else, following the argument above, it would be easy to derive a distinguisher for the pseudorandom function.

So suppose that the probability (now over the random choice of ω_0) that the simulator runs the relation algorithm on some pk' such that $\text{KGEN}(1^k, \omega_0) = pk'$ is noticeable. Recall that this probability is under the condition that the simulator never answers the adversary's decryption request correctly, and therefore the simulator never gets any information about ω_0. Hence, it suffices to show that for each pk' the probability that $\text{KGEN}(1^k, \omega_0)$ returns this public key is negligible, where the probability is over the secret choice of ω_0.

If the probability for $\text{KGEN}(1^k, \omega_0)$ returning a given public key was noticeable then we can easily break the semantic security of the scheme. Namely, if this was the case then for a given challenge pair pk', c' we pick ω_0 at random and run $\text{KGEN}(1^k, \omega_0)$. By assumption this yields some output (sk, pk) such that $pk' = pk$ with noticeable probability, and we are therefore able to recover the secret key to the given key pk' with noticeable probability. This, of course, contradicts the semantic security of the scheme.

Overall, the black-box simulator's success probability is negligible, whereas the adversary always succeeds. Hence, the encryption scheme cannot be completely non-malleable. □

6.3 Encryption and Assisted Simulators

In this section we show that black-box simulations remain infeasible for relations which are efficiently computable relative to an oracle $\mathcal{O}$, even if the simulator gets oracle access to the decryption oracle. Concerning the relations, the relativized result is slightly weaker than the result for stand-alone simulators (albeit the simulator now has more power). We remark that the simulator can still efficiently compute the relation via black-box access. Yet, since we consider the oracle to be a part of the relation, the simulator is not given access to this oracle directly, but merely through the relation.

We specify the adversary and the relation in detail. The message distribution is again the uniform distribution M_{unif} on strings of superlogarithmic length and we assume that hist returns the empty output. The adversary does not make any decryption queries and thus expects to receive a key $\overline{pk}$, possibly different from the simulator's input pk, and a challenge ciphertext $\bar{c}$ immediately. Then, the adversary $\mathcal{A}_{key}$ for security parameter k and key $key \in \{0,1\}^k$ works as follows:

- $\mathcal{A}_{key}$ first computes $m^* = \text{PRF}(key, 0, \overline{pk})$ and $(\omega^*, r^*) = \text{PRF}(key, 1, \overline{pk}, \bar{c})$.
- Next it computes $(sk^*, pk^*) = \text{KGEN}(1^k, \omega^*)$, i.e., the key generator's output for (pseudo)randomness ω^*, and $c^* = \text{ENC}(pk^*, m^*, r^*)$.
- The adversary outputs pk^*, c^* and stops.

The relation $\mathsf{R}^{\mathcal{O}}_{key}$ for input (pk, m, pk^*, c^*, m^*) returns 1 if and only if $m^* = \text{PRF}(key, 0, pk)$ and if there exists an r such that, for $c = \text{ENC}(pk, m, r)$, we have

$(\omega, r) = \text{PRF}(key, 1, pk, c)$ and $(sk^*, pk^*) = \text{KGEN}(1^k, \omega)$. This relation is computable in exponential time or, given on oracle $\mathcal{O}$ checking the second condition on input (pk, m, pk^*, key), in polynomial time relative to the oracle.

We again assume that the encryption scheme is semantically secure (against key-only attacks) with respect to M_{unif} and the equality relation defined by $\mathsf{R}_{\text{msg-eq}}(pk, m, pk^*, m^*, c^*) = 1$ iff $m = m^*$.

Theorem 7. *Encryption schemes which are completely non-malleable according to black-box assisted simulators for $\mathcal{M} \ni \mathsf{M}_{unif}$ and $\mathcal{R} \supseteq \{\mathsf{R}^{\mathcal{O}}_{\boldsymbol{key}} | \boldsymbol{key}\} \cup \{\mathsf{R}_{msg\text{-}eq}\}$ do not exist.*

In contrast to the case of stand-alone simulators, we are not aware if the theorem remains true if the simulator is only supposed to output pk', c' (and not m', r' as we require here).

Proof. It is clear that the adversary $\mathcal{A}_{\boldsymbol{key}}$ in an attack always outputs pk^*, c^* which satisfy the relation $\mathsf{R}^{\mathcal{O}}_{\boldsymbol{key}}$. We show that nay black-box simulator $\mathcal{S}$, even if given access to DEC, fails with overwhelming probability for randomly chosen key $\boldsymbol{key}$. Note that we apply the pseudorandom function at two places; prepending 0 or 1, respectively, makes these evaluations essentially independent. We therefore assume below for simplicity that we deal with two independent pseudorandom functions, described by keys key_0 and key_1. Formalizing and proving this is easy and omitted.

We first claim that the simulator, even with access to DEC, fails to ask the relation $\mathsf{R}^{\mathcal{O}}_{\boldsymbol{key}}$ for an input $(\overline{pk}, \overline{m}, \overline{pk}^*, \overline{c}^*, \overline{m}^*)$ during the black-box simulation such that the relation returns 1. Assume that this was not the case, and the probability was noticeable. In particular, $\mathcal{S}$ manages to find an input such that $\overline{m}^* = \text{PRF}(key_0, \overline{pk})$ with noticeable probability. We show that this refutes the unpredictability of the pseudorandom function.[5]

For deriving the contradiction to the unpredictability we first change the black-box simulation stepwise. In the first step, we let (each instance (i, j) of) the adversary on input $\overline{pk}, \overline{c}$ pick independent random values (ω^*, r^*) instead of computing it as the output $(\omega, r^*) = \text{PRF}(key_1, \overline{pk}, \overline{c})$ of the pseudorandom function. There is one exception, though: If an instance is run for a previously queried input $\overline{pk}, \overline{c}$ then we let this copy use the same pair (ω^*, r^*) as before, even if this input appeared for a different instance. In any case, $\mathcal{A}_{key}$ always uses $\text{PRF}(key_0, \cdot)$ to compute the value m^*.

With this modification the probability that $\mathcal{S}$ queries the relation about an input such that $\overline{m} = \text{PRF}(key_0, \overline{pk})$ remains noticeable according to the assumption. If this were not the case it would be easy to build a distinguisher for the pseudorandom function. Namely, we are given oracle access to either $\text{PRF}(key_1, \cdot)$ or a random function and are supposed to tell both cases apart. For this, we "simulate the simulator" by picking a key key_0, generating a key pair $(sk, pk) \leftarrow \text{KGEN}(1^k)$ and picking at random an index j between 1 and the

[5] Unpredictability says that it is hard to output x and an correct image y for the pseudorandom function, even if one has queried the function about other values $x' \neq x$ before. This property is implied by the pseudorandomness of the function.

(at most polynomial) number of queries the simulator makes to $\mathsf{R}^{\mathcal{O}}_{\boldsymbol{key}}$. We run the black-box simulation but use the given (random or pseudorandom) function oracle to compute the pairs (ω^*, r^*) if the adversary is invoked on $\overline{pk}, \overline{c}$. We answer the first $j-1$ queries to $\mathsf{R}^{\mathcal{O}}_{\boldsymbol{key}}$ with 0 and use the secret key sk to reply to decryption queries. If $\mathcal{S}$ puts the j-th query $(\overline{pk}, \overline{m}, \overline{pk}^*, \overline{c}^*, \overline{m}^*)$ to the relation then we stop with output 1 if and only if $\overline{m} = \textsc{prf}(key_0, \overline{pk})$ for the chosen key key_0.

Clearly, if $\mathcal{S}$ manages to put a query such that $\overline{m}^* = \textsc{prf}(key_0, \overline{pk})$ in the original setting, then we also output 1 with noticeable probability if the given function oracle is pseudorandom. Namely, in this case we guess the smallest query for which this happens correctly with noticeable probability, and in this case we perfectly simulate the original setting. If, on the other hand, the probability that the simulator fails to find such values if we use random values (ω^*, r^*) instead, except with negligible probability, then we output 1 with negligible probability only in case the given function oracle is random. Hence, both cases would be easy to distinguish.

In the next modification of the simulation experiment we let each adversarial instance output an encryption $c^* = \textsc{enc}(pk^*, 0^k, r^*)$ instead of $c^* = \textsc{enc}(pk^*, m^*, r^*)$. We claim again that this cannot decrease the simulator's success probability for a relation query with $\overline{m}^* = \textsc{prf}(key_0, \overline{pk})$ by more than a negligible amount. Else it would be straightforward to derive a contradiction to the semantic security of the encryption scheme, with a similar "simulation of the simulator" technique as before and access to an encryption oracle for random key pk^*.

Note that the adversary's behavior is now independent of m^* and of the pseudorandom function $\textsc{prf}(key_0, \cdot)$. Hence, we can mount the following attack against a given function oracle $\textsc{prf}(key_0, \cdot)$, trying to predict a value $\textsc{prf}(key_0, \overline{pk})$ with noticeable probability. We initially guess again an index j for the simulator's query to the relation. We start the experiment for random values $(sk, pk) \leftarrow \textsc{kgen}(1^k)$ and key_1. Each decryption query is answered with the help of sk, each run of an adversarial instance is simply answered by (consistent yet) independent random values $pk^*, \textsc{enc}(pk^*, 0^k, r^*)$. Observe that $\mathcal{A}$'s reply is computed independently of m^* and we do not query the the function $\textsc{prf}(key_0, \cdot)$ at any point. For the first $j-1$ queries of $\mathcal{S}$ to the relation return 0; for the j-th query $(\overline{pk}, \overline{m}, \overline{pk}^*, \overline{c}^*, \overline{m}^*)$ we stop immediately with output $(\overline{pk}, \overline{m}^*)$.

Clearly, by the assumption about the simulator, we have a noticeable success probability that indeed $\overline{m}^* = \textsc{prf}(key_0, \overline{pk})$. And since we have never queried the function oracle before, we derive a contradiction to the unpredictability of the pseudorandom function. Hence, our initial assumption must have been wrong. This also shows shows that the simulator will not manage to eventually find values pk', c', m', r' satisfying the relation, except with negligible probability. □

6.4 Signatures

The result from the previous sections (for stand-alone simulators) can be easily transfered to signatures. This time the adversary tests the given oracle by letting

it sign a fixed messages $m_{\text{test}} = 0^k$ and finally outputs a key vk^* (together with a signature s^* for message m_{test}) such that $\text{KGEN}(1^k, \omega)$ returns vk^*. Details are omitted.

Proposition 6. *Signature schemes which are completely non-malleable according to black-box simulations for $\mathcal{R} \supseteq \{\mathsf{R}_{\boldsymbol{key}} | \boldsymbol{key}\}$ do not exist.*

Yet, for signatures we can show that completely non-malleable systems for general relations are impossible at all, even when allowing non-black-box constructions or if the simulator depends on the relation. For this let $\mathsf{R}_{\text{encode}}$ be such that $\mathsf{R}_{\text{encode}}(vk, vk^*, m^*, s^*) = 1$ if and only if $m^* = vk||m||s$ encodes a valid signature s for m under vk, i.e., such that $\text{VF}(vk, m, s) = 1$.

Theorem 8. *There do not exist completely non-malleable signature schemes with respect to class $\mathcal{R} \ni \mathsf{R}_{encode}$.*

Proof. Suppose that there is such a signature scheme. Consider the following adversary $\mathcal{A}$ which, on input vk, first queries the signature oracle $\text{SIG}(sk, \cdot)$ about $m = 0^k$ to get a valid signature s for m under vk. The adversary then generates a key pair $(sk^*, vk^*) \leftarrow \text{KGEN}(1^k)$ and a signature s^* for $m^* = vk||m||s$ under vk^*. She finally outputs vk^*, m^*, s^*.

The adversary above generates a related output with probability 1 for relation $\mathsf{R}_{\text{encode}}$. But then there must be a simulator $\mathcal{S}$ such that $\mathcal{S}$ for input vk also outputs a valid tuple vk', m', s' under relation $\mathsf{R}_{\text{encode}}$ with probability negligibly close to 1. That is, the simulator's output m' must encode a valid signature s of some message m under the given public key vk with probability almost 1. In contrast to the adversary, the simulator does not get access to a signature oracle, though. Hence, with high probability $\mathcal{S}$ produces a valid signature given the public key vk only. This, however, contradicts the extra security property of the completely non-malleable scheme being unforgeable in key-only attacks, and there cannot exist such a signature scheme. □

The usage of relation $\mathsf{R}_{\text{encode}}$ for our negative result is not surprising: a signature oracle always leaks some information in the sense that it reveals that the signature s for a message m stands in a relation to the public key vk via $\text{VF}(vk, m, s)$. These information is not available to the simulator, though. As for our positive result in the next section, we will therefore exclude most of the relations based on VF.

6.5 Revisiting Relations over Messages Only

Inspecting the impossibility proof for encryption schemes and black-box simulators enables us to extend our separation example of Section 5 for non-malleability for relations over messages only. There we have shown that, if the relation can depend on the public key, the notion of non-malleability for such relations constitutes a strict extension for chosen-ciphertext pre-processing attacks (unconditionally). Here we show that the same is true for post-processing attacks for black-box (stand-alone) simulators.

We slightly change the construction of the previous sections for stand-alone simulators and let the pseudorandom function return tuples $(m_{\text{test}}, r_{\text{test}}, m^*, r^*)$ instead of $(m_{\text{test}}, r_{\text{test}}, \omega)$. The adversary finally outputs $c^* = \text{ENC}(pk, m^*; r^*)$ and we define the relation to be satisfied iff c^* is the ciphertext under pk obtained by encrypting the message m^* contained in the pseudorandom value for pk, using randomness r^*. The same argument as before shows that the simulator will not be able to learn this ciphertext from the adversary. Hence, the simulator's probability of accidently outputting a ciphertext of this message m^* is significantly less than 1, the adversary's success probability. We will now take advantage of this modification for our negative result about basic non-malleable encryption schemes and relations ranging over the public key.

Let $\{\mathsf{R}_{pk,\boldsymbol{key}}|\boldsymbol{key}\}$ be the set of relations such that $\mathsf{R}_{pk,\boldsymbol{key}}(pk, m, m^*) = 1$ iff m^* is part of the output $\text{PRF}(key, pk)$. Basically, the result then says that one cannot construct black-box simulators for these relations, even under chosen-ciphertext post-processing attacks. On the other hand, the [DDN00] construction is in fact a black-box construction for such attacks and relations $\mathcal{R}_{\text{msg}}$ over messages only.

Proposition 7. *Assume trapdoor permutations exist. Then there is an encryption scheme which is (completely) non-malleable according to black-box, stand-alone simulators for $\mathcal{R}_{msg}$ and $\mathcal{M}_{pk\text{-}ind}$. Yet, encryption schemes which are (completely) malleable according to black-box, stand-alone simulators for $\mathcal{R} \supseteq \{\mathsf{R}_{pk,\boldsymbol{key}}|\boldsymbol{key}\}$ do not exist.*

We are not aware if our results for assisted simulators can be applied as well, and if similar separations hold for such simulators.

7 Solutions in the Random Oracle Model

In this section we show how to modify schemes like RSA-OAEP and Fiat-Shamir signatures to get efficient completely non-malleable schemes.

7.1 Encryption

Recall from Section 2.2 that, in RSA-OAEP, a message m is encrypted under public key (N, e) and in presence of two random oracles G, H by computing

$$y = \big(m||0^k \oplus G(r)\big) \;\; || \;\; \big(r \oplus H(m||0^k \oplus G(r))\big) \in \mathbb{Z}_N^*.$$

and then outputting the ciphertext as $c = y^e \bmod N$. Here we slightly modify this by incorporating the public key into the hash function evaluations for random oracle G. Call this $\text{RSA-OAEP}_{\text{cnm}}$:

Key Generation: Generate an RSA modulus N and an RSA exponent e. The public key is given by $pk = (N, e)$ while the secret key is given by $d = e^{-1} \bmod \varphi(N)$.

Encryption: To encrypt a message m pick a random string r and compute

$$y = \big(m||0^k \oplus G(pk,r)\big) \;||\; \big(r \oplus H(m||0^k \oplus G(pk,r))\big) \in \mathbb{Z}_N^*.$$

Return $c = y^e \bmod N$. We further assume that the position of the (at least) k zero-bits in the left part is uniquely determined by the public key.

Decryption: To decrypt $c \in \mathbb{Z}_N^*$ compute $y = c^d \bmod N$ and parse y as $(m||0^k \oplus G(pk,r)) \;||\; (r \oplus H(m||0^k \oplus G(pk,r)))$. Compute the hash value of the left part under H and xor it to the right part to retrieve r. Then compute the exclusive-or of the left part and $G(pk,r)$. If the least significant k bits are all 0 then output m.

We also assume that the adversary may only output *well-formed* RSA keys, i.e., where e is relatively prime to $\varphi(N)$, independently of N. Put differently, we presume that decryption is unique. As discussed in Section 3.1 we do not consider the enforcement of this property to be an issue of non-malleability.

Proposition 8. *Under the RSA assumption RSA-OAEP$_{cnm}$ with well-formed keys is a completely non-malleable encryption scheme for any distribution class $\mathcal{M}$ and any relation class $\mathcal{R}$.*

Proof. The proof makes use of the plaintext-extractor in [BR95,FOPS01] which basically allows to replace the decryption oracle by inspecting the queries to the random oracles and thereby retrieving the plaintext from a ciphertext. Here, we slightly modify the original extractor and take the public-key prefix for hashes into account and also let the extractor return both the plaintext and the randomness.

The plaintext-extractor $\mathcal{PE}$ here takes as input the well-formed RSA-key (N,e), a value $c \in \mathbb{Z}_N^*$ and two lists $(\gamma_i, G_{\gamma_i})_{i=1,2,\ldots}$ and $(\delta_i, H_{\delta_i})_{i=1,2,\ldots}$ of queries and answers to oracles G and H. From the lists it first filters out those where (N,e) is not a prefix of γ_i. With the remaining pairs it generates all possible ciphertexts $\big(\delta_i||\gamma_j \oplus H_{\delta_i}\big)^e \bmod N$ and checks if the given c is among them, and if the k least significant bits of $G_{\gamma_j} \oplus \delta_i$ are all zero. If it finds such a pair then it outputs $G_{\gamma_j} \oplus \delta_i$ (without the k least significant 0-bits) as well as γ_j as the random bits. Otherwise $\mathcal{PE}$ returns $\perp$.

As in [FOPS01] it follows that the extractor's answer is unique and, given that oracles G, H have been queried to produce c under (N,e), then the extractor perfectly simulates the decryption oracle. This holds even if the extractor is given an arbitrary, yet well-formed key (N^*, e^*) instead.

Our proof also relies on the observation in [FOPS01] that RSA is a so-called *set partial-domain one-way function*: Given an algorithm A that gets as input some well-formed key (N,e) and a random $c = (s||t)^e \bmod N$ the algorithm is supposed to output a list including s. That is, the algorithm should find a list including parts of the preimage of c. Denote by $\mathsf{Succ}^{s-pd-ow}(A)$ the success probability of A. Fujisaki et al. [FOPS01] show that, as long as the length of s is large enough, e.g., as in case of RSA-OAEP, every polynomial-time algorithm A only has negligible success probability under the RSA assumption.

We now show how to construct a simulator $\mathcal{S}$ from any adversary $\mathcal{A}$ such that the simulator outputs related data almost with the same probability as $\mathcal{A}$. Fix some distribution M and relation R. The simulator gets a public key $pk = (N, e)$ and oracle access to G, H. It starts a simulation of $\mathcal{A}$ on pk. The simulator records each hash oracle query of $\mathcal{A}$ to G or H, retrieves the answer for this query from its own oracle and hands this reply back to the adversary. Each decryption query about some ciphertext is answered by running $\mathcal{PE}$ on (N, e), the ciphertext as well as the list of queries and answers to G and H. We assume that the simulator strips off the randomness from the answer of $\mathcal{PE}$.

In the second phase a message $m \leftarrow \mathsf{M}(pk)$ is sampled and $h \leftarrow \mathsf{hist}(m)$ is given to the simulator. $\mathcal{S}$ passes a random value $c \in \mathbb{Z}_N^*$ and h to the adversary. Each further oracle calls, either to G, H or the decryption oracle, are answered as before. If the adversary finally outputs a well-formed $pk^* = (N^*, e^*)$ and a ciphertext c^* then the simulator invokes once more $\mathcal{PE}$ on pk^*, c^* and the query lists to get m^*, r^*. It outputs $pk' = pk^*$, $c' = c^*$, $m' = m^*$, $r' = r^*$ and stops.

To show that the simulator approximates the adversary's success probability closely, we use the approach in [FOPS01] and consider the following sequence of games, starting from $\mathcal{A}$'s attack with decryption queries and transforming it into the simulation above with plaintext-extractions. In each game GAME_i we denote by π_i the adversary's success probability $\pi_{\text{enc}}(\mathcal{A}, \mathsf{M}, \mathsf{R})$ *in this game*. Below we also use the notation $s = m||0^k \oplus G(pk, r)$ and $t = r \oplus H(s)$ for the left and right part of ciphertexts.

In $\mathcal{A}$'s original attack (GAME_0) and the subsequent games we add an artificial step at the end, when the adversary has output pk^*, c^*. We then run an all-powerful decryption oracle DEC* which, given a well-formed key (N^*, e^*), computes (presumably in exponential time) the decryption key d^* and decrypts c^* to m^* and r^*. Note that all values d^*, m^*, r^* are uniquely determined given pk^*, c^* and H, G because the key is well-formed. This extra steps enables us to relate the simulator's final run of $\mathcal{PE}$ with a decryption request to DEC*, i.e., while morphing the actual attack into a simulation we will replace this decryption oracle DEC* by the (efficient) plaintext-extractor.

GAME_0: Describes the actual attack of $\mathcal{A}$ when given pk, oracle access to G, H and a decryption oracle DEC$(sk, \cdot)$, a ciphertext $c = (s||t)^e \bmod N$ for the sampled message $m \leftarrow \mathsf{M}(pk)$ and $h \leftarrow \mathsf{hist}(m)$. When the adversary has output pk^*, c^* run DEC* to get m^*, r^*. Let $\pi_0 = \pi_{\text{enc}}(\mathcal{A}, \mathsf{M}, \mathsf{R})$.

GAME_{0^*}: [6] Execute GAME_0 but if the adversary finally outputs $pk^* \neq pk$ but $c^* = c$ then let DEC* return $\perp$ instead. Note that for a well-formed key $pk^* = (N^*, e^*)$ the ciphertext c uniquely determines $c = (s^*||t^*)^{e^*} \bmod N^*$. Given s^*, t^* oracles G, H then pin down r^* and m^*. In other words, s^*, r^* are a function of pk^* (for given parameters c, G, H). Hence, the probability that the adversary finds some $pk^* \neq pk$ during the attack such that the least k significant bits of $s^* \oplus G(pk^*, r^*)$ —whose positions are fixed once the key

[6] Our games here correspond to the games in [FOPS01]. Yet, this is an extra step which does not appear there.

is chosen— equal 0^k for the given c, is negligible. Note that for $pk^* = pk$ the bits are certainly 0^k, but for $pk^* \neq pk$ the value $G(pk^*, r^*)$ is independently distributed. We derive that $|\pi_0 - \pi_{0^*}|$ is negligible, too.

GAME_1**:** Run GAME_{0^*} with one exception: Preselect a value r and answer $G_r = G(pk, r)$ at the outset of the execution, use them for creating the ciphertext c and substitute each further query to G about (pk, r) by the answer G_r. Note that $\pi_1 = \pi_{0^*}$ because r and G_r are still random.

GAME_2**:** Run GAME_1, only this time do *not* replace further queries to G about r by the preselected value G_r but answer with the true value $G(pk, r)$ instead. Still, we use G_r to compute the ciphertext. Then, unless the adversary or the decryption oracles query oracle G about pk, r during the execution (event AskG_2), we have $\pi_1 = \pi_2$. Therefore, $|\pi_1 - \pi_2| \leq \mathrm{Prob}[\mathsf{AskG}_2]$. Note that if the adversary was allowed to submit the challenge ciphertext c to the decryption oracle or if we ran DEC* on c, then the $\mathrm{Prob}[\mathsf{AskG}_2]$ could be 1.

GAME_3**:** Execute GAME_2. But this time preselect independent random values s and H_s at the beginning and a substitute each query to H about s by answering with H_s. It follows that, since $G(pk, r) = s \oplus m || 0^k$ is also chosen at random, $\mathrm{Prob}[\mathsf{AskG}_2] = \mathrm{Prob}[\mathsf{AskG}_3]$ and $\pi_2 = \pi_3$.

GAME_4**:** This game is almost identical to the previous one, only drop the requirement about answering $H(s)$ consistently, i.e., if the decryption algorithm or the adversary at some point query H about s (event AskH_4) return the genuine value $H(s)$ instead of H_s. Given that AskH_4 does not occur we have $\mathrm{Prob}[\mathsf{AskG}_3] = \mathrm{Prob}[\mathsf{AskG}_4]$ and $\pi_3 = \pi_4$, and thus $|\mathrm{Prob}[\mathsf{AskG}_4] - \mathrm{Prob}[\mathsf{AskG}_3]|, |\pi_3 - \pi_4| \leq \mathrm{Prob}[\mathsf{AskH}_4]$.
Also, the value $r = t \oplus H_s$ is uniformly distributed an independent of the adversary's view. Hence, the probability $\mathrm{Prob}[\mathsf{AskG}_4]$ that the adversary or the decryption oracle queries G about r among the at most polynomial number of queries is negligible.

GAME_5**:** Replace the ciphertext c by an independent and randomly selected value in $\mathbb{Z}_N^*$. Clearly, $\mathrm{Prob}[\mathsf{AskH}_4] = \mathrm{Prob}[\mathsf{AskH}_5]$ and $\pi_4 = \pi_5$.

GAME_6**:** Run GAME_5 but slightly modify the decryption oracles (i.e., the original decryption oracle and the special oracle DEC*). Each time a ciphertext $\overline{c}$ is submitted (for key $\overline{pk} \in \{pk, pk^*\}$) that would decrypt to message $\overline{m}$ for randomness $\overline{r}$, reject this ciphertext with output $\perp$ if the adversary has not queried $G(\overline{pk}, \overline{r})$ before. Note that $G(\overline{pk}, \overline{r})$ is uniformly and independently distributed and hence, if $G(\overline{pk}, \overline{r})$ has not been queried before, then only with probability 2^{-k} the least significant bits of $\overline{s} \oplus G(\overline{pk}, \overline{r})$ equal 0^k. Hence, except with negligible probability, none of these at most polynomially many submitted ciphertexts would have been decrypted correctly by the decryption oracle. Therefore, $|\mathrm{Prob}[\mathsf{AskH}_5] - \mathrm{Prob}[\mathsf{AskH}_6]|$ and $|\pi_5 - \pi_6|$

are negligible.

GAME_7: Again, run GAME_6 but this time let the decryption oracles also reject ciphertexts $(\overline{s}||\overline{t})^{\overline{e}} \bmod \overline{N}$ where the encrypted part $\overline{s}$ has not been obtained previously from oracle H. The only difference to the previous game stems from submitted ciphertexts such that $G(\overline{pk}, \overline{r})$ has been queried but $H(\overline{s})$ has not. For the ciphertext it must hold $\overline{r} = \overline{t} \oplus H(\overline{s})$. But $H(\overline{s})$ is independently and uniformly distributed, and the probability that $\overline{t} \oplus H(\overline{s})$ equals one of the at most polynomially many queries before is negligible, and so are $|\operatorname{Prob}[\mathsf{AskH}_6] - \operatorname{Prob}[\mathsf{AskH}_7]|$ and $|\pi_6 - \pi_7|$.

GAME_8: Replace the decryption oracles by $\mathcal{PE}$. In GAME_7 only decryption request with previously queried values $\overline{r}, \overline{s}$ are decrypted. For such queries the plaintext-extractor simulates the oracles perfectly, i.e., $\operatorname{Prob}[\mathsf{AskH}_7] = \operatorname{Prob}[\mathsf{AskH}_8]$ and $\pi_7 = \pi_8$.
We claim that $\operatorname{Prob}[\mathsf{AskH}_8]$ is negligible. For this, note that it is easy to modify the adversary $\mathcal{A}$ for GAME_8 into an attacker A on the partial-domain one-wayness of RSA. Algorithm A gets as input (N, e) and a value $c \in \mathbb{Z}_N^*$ and simulates $\mathcal{A}$ by running GAME_8, providing and recording random but consistent answers to $\mathcal{A}$'s oracle queries G, H and applying the plaintext-extractor. A also uses c as the fake challenge ciphertext. Algorithm A finally outputs the list of queries to oracle H. Recall that AskH_8 is the event that the value s encrypted in the value c given to A is passed to H in GAME_8. Hence, $\operatorname{Prob}[\mathsf{AskH}_8] \leq \mathsf{Succ}^{s-pd-ow}(A)$.

Putting all probabilities together we obtain for a negligible function $\nu(k)$:

$$\begin{aligned}
|\pi_0 - \pi_8| &\leq |\pi_0 - \pi_{0^*}| + |\pi_{0^*} - \pi_1| + \sum_{i=1}^{7} |\pi_i - \pi_{i+1}| \\
&= \nu(k) + |\pi_1 - \pi_2| + |\pi_5 - \pi_6| + |\pi_6 - \pi_7| \\
&\leq |\pi_1 - \pi_2| + 3\nu(k)
\end{aligned}$$

where

$$\begin{aligned}
|\pi_1 - \pi_2| &\leq \operatorname{Prob}[\mathsf{AskG}_2] = \operatorname{Prob}[\mathsf{AskG}_3] \\
&\leq \operatorname{Prob}[\mathsf{AskG}_4] + \operatorname{Prob}[\mathsf{AskH}_4] = \nu(k) + \operatorname{Prob}[\mathsf{AskH}_4] \\
&= \nu(k) + \operatorname{Prob}[\mathsf{AskH}_5] \leq = 2\nu(k) + \operatorname{Prob}[\mathsf{AskH}_6] \\
&\leq 3\nu(k) + \operatorname{Prob}[\mathsf{AskH}_7] \\
&= 3\nu(k) + \operatorname{Prob}[\mathsf{AskH}_8] \\
&\leq 3\nu(k) + \mathsf{Succ}^{s-pd-ow}(A)
\end{aligned}$$

is negligible.

Hence, the probability of $\mathcal{A}$ outputting related data in GAME_8 is negligibly close to the one in the actual attack. But GAME_8 perfectly describes the simulation of $\mathcal{A}$ by $\mathcal{S}$, including the final step where $\mathcal{S}$ runs the plaintext-extractor once

more, but on the adversary's output pk^*, c^*. Note that if $\text{ENC}(pk^*, m^*; r^*) = c^*$ then the all-powerful decryption oracle DEC^* would recover m^* and r^* for the well-formed key $pk^* = (N^*, e^*)$. So does the plaintext-extractor in GAME_8, and the simulator approximates therefore π_0 closely. □

7.2 Signatures

We consider the classical Fiat-Shamir like signature schemes in the random oracle model. Recall that we have presented the Schnorr signatures as a concrete instantiation of such signatures in Section 2.3. More formally, we require the following common properties of the underlying identification schemes:

Definition 3. *A canonical Fiat-Shamir identification scheme is a tuple* $(\text{KGEN}_{id}, \text{COM}_{id}, \text{RESP}_{id}, \text{VF}_{id})$ *of efficient algorithms such that*

- *(Completeness). For any security parameter* k*, any keys* $(vk, sk) \in [\text{KGEN}(1^k)]$*, any commitment* $(com, r) \in [\text{COM}_{id}(vk, sk)]$*, any challenge* $c \in \{0,1\}^t$*, any response* $resp \in [\text{RESP}_{id}(vk, sk, r, c)]$*, we have* $\text{VF}_{id}(vk, com, c, resp) = 1$*.*
- *(Passive Security). For any pair of probabilistic polynomial-time algorithms* $(\mathcal{A}_0, \mathcal{A}_1)$ *the probability that* $\text{VF}_{id}(vk, com, c, resp) = 1$ *for* $(vk, sk) \leftarrow \text{KGEN}(1^k)$*,* $(com, \alpha) \leftarrow \mathcal{A}_0(vk)$*,* $c \leftarrow \{0,1\}^t$*,* $resp \leftarrow \mathcal{A}_1(\alpha, c)$*, is negligible.*
- *(Entropy of Commitment). For any value* com_0 *the probability that* $(com, r) \leftarrow \text{COM}_{id}(vk, sk)$ *yields a collision* $com = com_0$ *is negligible.*
- *(Recovery of Commitment). For any security parameter* k*, any keys* $(vk, sk) \in [\text{KGEN}(1^k)]$*, any commitment* $(com, r) \in [\text{COM}_{id}(vk, sk)]$*, any challenge* $c \in \{0,1\}^t$*, any response* $resp \in [\text{RESP}_{id}(vk, sk, r, c)]$*, one can recover* com *from* $(vk, c, resp)$ *in polynomial time.*
- *(Special Zero-Knowledge). There exists an efficient algorithm* $\mathcal{Z}$*, the zero-knowledge simulator, such that for any* vk*, for* $c \leftarrow \{0,1\}^t$ *the output* $resp \leftarrow \mathcal{Z}(vk, c)$ *is identically distributed*[7] *to* $resp \leftarrow \text{RESP}_{id}(vk, sk, r, c)$ *for* $(com, r) \leftarrow \text{COM}_{id}(vk, sk)$*.*
- *(Quasi Unique Responses). For any efficient adversary* $\mathcal{A}$ *the probability that for* $(vk, sk) \leftarrow \text{KGEN}_{id}(1^k)$*,* $(com, r) \leftarrow \text{COM}_{id}(vk, sk)$*,* $c \leftarrow \{0,1\}^t$*,* $resp \leftarrow \text{RESP}_{id}(vk, sk, r, c)$*,* $resp^* \leftarrow \mathcal{A}(vk, com, c, resp)$*,*

$$resp \neq resp^* \qquad \text{and} \qquad \text{VF}_{id}(vk, com, c, resp^*) = 1$$

is negligible.

The original Fiat-Shamir transformation [FS86] turns the interactive identification scheme into a non-interactive signature scheme by replacing the challenge c by the hash value over com and the message m. Here, we slightly change this by hashing the verification key vk in addition to com and m. That is, the challenge is given by $c = H(vk, com, m)$; all other steps remain. We call this the *Fiat-Shamir signature protocol with public-key hashing.*

[7] Computational indistinguishability actually suffices for our result. But most schemes indeed achieve perfect indistinguishability and this simplifies the proof of complete non-malleability.

Definition 4. *Let* $(\text{KGEN}_{id}, \text{COM}_{id}, \text{RESP}_{id}, \text{VF}_{id})$ *be a canonical Fiat-Shamir identification scheme. Then the derived Fiat-Shamir signature scheme* $(\text{KGEN}, \text{SIG}, \text{VF})$ *with public-key hashing is described by the following algorithms in the random oracle model:*

Key Generation: *Algorithm* $\text{KGEN}^H(1^k)$ *runs* $\text{KGEN}_{id}(1^k)$ *to get keys* (sk, vk).

Signing: *Algorithm* SIG^H *on input the signing key* sk *and a message* $m \in \{0,1\}^*$ *computes* $(com, r) \leftarrow \text{COM}_{id}(sk)$, $c \leftarrow H(vk, com, m)$ *and* $resp \leftarrow \text{RESP}_{id}(sk, r, c)$. *It outputs the signature* $s = (c, resp)$.

Verifying: *Algorithm* VF^H *on input the verification key* vk, *a message* m *and a signature* $s = (c, resp)$ *first recovers* com, *then compares* $c{=}H(vk, com, m)$ *and, in case of equality, finally returns* $\text{VF}_{id}(vk, com, c, resp)$.

The class $\mathcal{R}_{\text{ro,sig}}$ for which show complete non-malleability consists of relations of the form

$$\mathsf{R}^H(vk, vk^*, m^*, s^*) \\ = \left[\text{VF}^H(vk, m^*, s^*) \ \wedge \ \mathsf{R}_0^H(vk, vk^*, m^*, s^*)\right] \ \vee \ \mathsf{R}_1^{H_{vk}}(vk, vk^*, m^*, s^*)$$

where $H_{vk}(x)$ is a restricted oracle which returns $\perp$ if vk is a prefix of x, and $H(x)$ else. In particular, relation R_1 cannot include verification checks VF^H with respect to the random oracle and the given key vk, and the only possible verification step is to verify m^*, s^* under vk and via relation R_0.

Proposition 9. *Fiat-Shamir signature protocols with public-key hashing are completely non-malleable with respect to class* $\mathcal{R}_{ro,sig}$.

Note that the class $\mathcal{R}_{\text{ro,sig}}$ is comprehensive enough to include the relations $\mathsf{R}_{\text{str-unf}}$ and $\mathsf{R}_{\text{key-sub}}$ for thwarting strong-unforgeability and key-substitution attacks. Hence, before giving the proof of Proposition 9 we conclude:

Corollary 1. *Fiat-Shamir signature schemes with public-key hashing are strongly unforgeable and secure under key-substitution attacks, both under adaptive chosen-message attacks.*

Proof. (of Proposition 9) It has been shown in [PS00] that Fiat-Shamir signature schemes are secure against adaptive chosen-message attacks and, in particular, under key-only attacks. This easily transfers to the case of public-key hashing. It remains to prove the "actual" property of complete non-malleability.

Assume that there is an adversary $\mathcal{A}$ that mounts a complete non-malleability attack on input vk and access to a signature oracle $\text{SIG}^H(sk, \cdot)$ and a random oracle H. We construct a simulator $\mathcal{S}$ which gets vk as input and access to random oracle H. It runs an execution of $\mathcal{A}$ on input vk and also simulates a random oracle H' for $\mathcal{A}$ which we define next.

To define oracle H' we first specify how to deal with signature requests m of the adversary. For such a query the simulator $\mathcal{S}$ picks c at random, runs

the special zero-knowledge simulator $\mathcal{Z}$ for the underlying Fiat-Shamir protocol to derive *resp* matching this challenge. $\mathcal{S}$ replies with $s = (c, resp)$ and sets $H'(vk, com, m) \leftarrow c$, i.e., if the adversary later submits a query (vk, com, m) to H' then $\mathcal{S}$ answers with c on behalf of H'. Analogously, any direct query to oracle H' involving prefix vk is answered by picking a random value c and re-using this value c (if either the adversary later resubmits this value to H', or if this value appears during a signature simulation). The simulator answers any other query x of $\mathcal{A}$ to H' by forwarding x to H and handing the oracle reply back to $\mathcal{A}$. When the adversary finally outputs vk^*, m^* and s^* the simulator too returns these values if they have never been answered by the simulated signature oracle, and the simulator returns $\perp$ otherwise.

Fix some relation $\mathsf{R} \in \mathcal{R}_{\mathrm{ro,sig}}$. Recall that

$$\begin{aligned}&\mathsf{R}^H(vk, vk^*, m^*, s^*)\\ &\quad = \left[\mathrm{VF}^H(vk, m^*, s^*) \ \wedge \ \mathsf{R}_0^H(vk, vk^*, m^*, s^*)\right] \ \vee \ \mathsf{R}_1^{H_{vk}}(vk, vk^*, m^*, s^*)\end{aligned}$$

Let (vk, m_i, s_i) with $s_i = (c_i, resp_i)$ be the sequence of answers of the simulated signature oracle. We show that, compared to the adversary, the simulator only fails to generate a good output with very small probability.

The first observation is that the simulator perfectly simulates a random oracle from the adversary's viewpoint, unless the simulator preselects some hash value c for some query (vk, com, m) such that the same tuple appears later in one of the signature queries. In this case, the simulation above yields some inconsistent answer. By the entropy of the commitment the probability that this occurs during an attack is negligible, though, and we will from now on neglect executions with such inconsistencies.

We have to ensure that the adversary hardly outputs a key vk^*, a message m^* and a signature $s^* = (c^*, resp^*)$ such that the relation holds with respect to the simulated oracle H' but not for the given oracle H. But the only difference between the two functions originates from hash values with prefix vk. Such values, however, do not affect $\mathsf{R}_1^{H_{vk}} \equiv \mathsf{R}_1^{H'_{vk}}$; only the outcome of the first part of the relation may depend on them.

Assume that the adversary runs a successful attack and outputs $vk^*, m^*, s^* = (c^*, resp^*)$ such that $\mathrm{VF}^{H'}(vk, m^*, s^*)$ holds. Let com^* and $\overline{com}$ be the recovered commitments from $(vk^*, c^*, resp^*)$ and from $(vk, c^*, resp^*)$, respectively. Since the attack is successful s^* must by definition be a valid signature for m^* under vk^* and (vk^*, m^*, s^*) has never been returned by the signature oracle (which, otherwise, would have been the only case in which the simulator does not copy the adversary's output). Below we distinguish between four cases, depending on vk, vk^*, m^* and the signature queries m_i.

Case 1. Suppose first that $vk^* \neq vk$. The hash value c^* in $s^* = (c^*, resp^*)$ must be valid for both (vk^*, com^*, m^*) and $(vk, \overline{com}, m^*)$, i.e,

$$\begin{aligned}&\mathrm{Prob}_{H'}\left[H'(vk^*, com^*, m^*) = H'(vk, \overline{com}, m^*)\right]\\ &\qquad = \mathrm{Prob}_{H}\left[H(vk^*, com^*, m^*) = H'(vk, \overline{com}, m^*)\right].\end{aligned}$$

The latter probability is negligible, and the probability of finding such a tuple among the polynomial number of hash queries during the attack is therefore negligible, too. We can therefore assume that $vk^* = vk$.

Case 2. If $vk^* = vk$ and $m^* \neq m_i$ for all signature replies then the adversary can only output a valid signature under vk with negligible probability. If this were not the case then it would be straightforward to construct a successful attacker forging signatures under adaptive chosen-message attacks on the basic Fiat-Shamir scheme (without public key hashing). Namely, the forger is given a public key vk, a signature oracle and a random oracle H and runs the adversary by relaying all messages between the oracles, except if the adversary submits a message m to the signature oracle; then the forger first modifies it to a message $vk||m$. Note that the given random oracle H has the same distribution as the simulated one H' and the adversary therefore would also succeed for $vk^* = vk$ and $m^* \neq m_i$ here with noticeable probability, as in the simulation. The constructed forger, though, would contradict the unforgeability of the underlying scheme against adaptive chosen-message attacks [PS00].

Case 3. The case $vk^* = vk$, $m^* = m_i$ for one signature query m_i (or even some queries), implies that $s^* = (c^*, resp^*) \neq s_i = (c_i, resp_i)$ for all such indices; else the adversary would merely duplicate a signature. Denote by com^* and com_i the commitments recovered from $vk^*, c^*, resp^*$ and $vk, c_i, resp_i$, respectively. If $com^* \neq com_i$ for one of these indices i with noticeable probability then we construct an impersonater successfully attacking the passive security of the underlying identification scheme, as explained next.

For deriving the impersonater we presume that the adversary never queries the random oracle twice about a value, and that the adversary queries the random oracle about the about the final output (vk, com^*, m^*) at some point; if not, the probability that this tuple is sent to c^* by H' is negligible. We presume that the impersonater correctly guesses the corresponding query among the polynomial number of hash queries.

The impersonater is given a public key vk and starts an emulation of the adversary. This emulation perfectly mimics the simulator's behavior. Each hash oracle query with prefix $vk^* \neq vk$ is answered by returning a random value. Furthermore, except for the guessed query (vk, com^*, m^*), the impersonater simulates signature requests like the simulator by picking the challenges in advance and running the zero-knowledge simulator. For the query (vk, com^*, m^*) the impersonater invokes into an execution with the verifier $\mathrm{VF}_{\mathrm{id}}$ on key vk. It sends com^* to this verifier and receives a challenge c^* as reply. The impersonater returns c^* as the hash function value to the adversary. We remark that, if $com^* \neq com_i$, the impersonater never has to give a signature for this commitment and does not have to pick the value ahead of the commitment.

At the end of the execution, the impersonater gets a valid signature $s^* = (c^*, resp^*)$ with noticeable probability. It forwards $resp^*$ to the verifier. The verifier accepts this answer with noticeable probability, and this refutes the passive security of the identification scheme.

Case 4. Finally, suppose $vk^* = vk$, $m^* = m_i$ for some i, $com^* = com_i$ and thus $c^* = c_i$, but $resp^* \neq resp_i$. By the property of quasi unique responses of the underlying identification protocol the probability of finding such a valid response $resp^* \neq resp$ for (vk, com_i, c_i) is negligible.

Altogether, the simulator's success probability is negligibly close to the adversary's success probability, and the scheme with public-key hashing is therefore completely non-malleable. □

It is interesting of course to pin down the step in the proof where we escape the negative result from the previous section. In the case of black-box simulations, the simulator essentially cannot run the adversary on the given public key vk because it is necessary to provide a valid signature under vk for this. Here, the simulator can indeed provide such a valid signature by modifying the random oracle. The unconditional impossibility result is bypassed by restricting the relations such that altering the hash values does not affect the outcome.

Acknowledgments

We would like to thank Yevgeniy Dodis, Alejandro Hevia, Bogdan Warinschi and the reviewers for helpful input.

References

[Bar02] Boaz Barak. *Constant-Round Coin-Tossing With a Man in the Middle or Realizing the Shared Random String Model*. Proceedings of the Annual Symposium on Foundations of Computer Science (FOCS) 2002. IEEE Computer Society Press, 2002.

[BR95] Mihir Bellare and Phillip Rogaway. *Optimal Asymmetric Encryption — How to Encrypt with RSA*. Advances in Cryptology — Eurocrypt'94, Volume 950 of Lecture Notes in Computer Science, pages 92–111. Springer-Verlag, 1995.

[BWM99] Simon Blake-Wilson and Alfred Menezes. *Unknown Key-Share Attacks on the Station-to-Station (STS) Protocol*. Public-Key Cryptography (PKC)'99, Volume 1560 of Lecture Notes in Computer Science, pages 154–170. Springer-Verlag, 1999.

[CIO98] G. Di Crescenzo, Y. Ishai, and R. Ostrovsky. *Non-interactive and Non-Malleable Commitment*. Proceedings of the Annual Symposium on the Theory of Computing (STOC) 1998, pages 141–150. ACM Press, 1998.

[CKOS01] G. Di Crescenzo, J. Katz, R. Ostrovsky, and A. Smith. *Efficient And Non-Interactive Non-Malleable Commitment*. Advances in Cryptology — Eurocrypt 2001, Volume 2045 of Lecture Notes in Computer Science, pages 40–59. Springer-Verlag, 2001.

[CS98] Ronald Cramer and Victor Shoup. *A Practical Public Key Cryptosystem Provably Secure Against Adaptive Chosen Ciphertext Attacks*. Advances in Cryptology — Crypto'98, Volume 1462 of Lecture Notes in Computer Science, pages 13–25. Springer-Verlag, 1998.

[DDN00] D. Dolev, C. Dwork, and M. Naor. *Non-Malleable Cryptography*. *SIAM Journal on Computing*, 30(2):391–437, 2000.

[FF00] Marc Fischlin and Roger Fischlin. *Efficient Non-Malleable Commitment Schemes*. Advances in Cryptology — Crypto 2000, Volume 1880 of Lecture Notes in Computer Science, pages 414–432. Springer-Verlag, 2000.

[FF02] Marc Fischlin and Roger Fischlin. *The Representation Problem Based on Factoring*. Topics in Cryptology — Cryptographer's Track, RSA Conference (CT-RSA) 2002, Volume 2271 of Lecture Notes in Computer Science, pages 96–113. Springer-Verlag, 2002.

[FOPS01] E. Fujisaki, T. Okamoto, David Pointcheval, and Jacques Stern. *RSA-OAEP is Secure Under the RSA Assumption*. Advances in Cryptology — Crypto 2001, Volume 2139 of Lecture Notes in Computer Science. Springer-Verlag, 2001.

[FS86] A. Fiat and A. Shamir. *How to Prove Yourself: Practical Solutions to Identification and Signature Schemes*. Advances in Cryptology — Crypto'86, Volume 263 of Lecture Notes in Computer Science, pages 186–194. Springer-Verlag, 1986.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. *How to Construct Random Functions. Journal of the ACM*, 33:792–807, 1986.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. *The Knowledge Complexity of Interactive Proof Systems. SIAM Journal on Computing*, 18(1):186–208, 1989.

[HILL99] J. Håstad, R. Impagliazzo, L. Levin, and M. Luby. *A Pseudorandom Generator from any One-way Function. SIAM Journal on Computing*, 28(4):1364–1396, 1999.

[IL89] R. Impagliazzo and M. Luby. *One-way Functions are Essential for Complexity Based Cryptography*. Proceedings of the Annual Symposium on Foundations of Computer Science (FOCS) '89, pages 230–235. IEEE Computer Society Press, 1989.

[Kal02] Burton Kaliski. *On Hash Function Firewalls in Signature Schemes*. Topics in Cryptology — Cryptographer's Track, RSA Conference (CT-RSA) 2002, Volume 2271 of Lecture Notes in Computer Science, pages 1–16. Springer-Verlag, 2002.

[MS04] Alfred Menezes and Nigel Smart. *Security of Signature Schemes in a Multi-User Setting*. Designs, Codes and Cryptography, Volume 33, pages 261–274. Springer-Verlag, 2004.

[Nie02] Jesper Nielsen. *Separating Random Oracle Proofs from Complexity Theoretic Proofs: The Non-committing Encryption Case*. Advances in Cryptology — Crypto 2002, Volume 1442 of Lecture Notes in Computer Science, pages 111–126. Springer-Verlag, 2002.

[PS00] David Pointcheval and Jacques Stern. *Security Arguments for Digital Signatures and Blind Signatures. Journal of Cryptology*, 13(3):361–396, 2000.

[RSA02] RSA Laboratories. *PKCS #1 v2.1*. RSA Cryptography Standard. RSA Security, June 2002.

[Sch91] C.P. Schnorr. *Efficient Signature Generation by Smart Cards. Journal of Cryptology*, 4:161–174, 1991.