Random Oracle Reducibility

Paul Baecher and Marc Fischlin

Darmstadt University of Technology, Germany
www.minicrypt.de

Abstract. We discuss a reduction notion relating the random oracles in two cryptographic schemes A and B. Basically, the random oracle of scheme B reduces to the one of scheme A if *any* hash function instantiation of the random oracle (possibly still oracle based) which makes A secure also makes B secure. In a sense, instantiating the random oracle in scheme B is thus not more demanding than the one for scheme A. If, in addition, the standard cryptographic assumptions for scheme B are implied by the ones for scheme A, we can conclude that scheme B actually relies on weaker assumptions. Technically, such a conclusion cannot be made given only individual proofs in the random oracle model for each scheme.
The notion of random oracle reducibility immediately allows to transfer an uninstantiability result from an uninstantiable scheme B to a scheme A to which the random oracle reduces. We are nonetheless mainly interested in the other direction as a mean to establish hierarchically ordered random-oracle based schemes in terms of security assumptions. As a positive example, we consider the twin Diffie-Hellman (DH) encryption scheme of Cash et al. (Journal of Cryptology, 2009), which has been shown to be secure under the DH assumption in the random oracle scheme. It thus appears to improve over the related hashed ElGamal encryption scheme which relies on the random oracle model and the strong DH assumption where the adversary also gets access to a decisional DH oracle. As explained above, we complement this believe by showing that the random oracle in the twin DH scheme actually reduces to the one of the hashed ElGamal encryption scheme. We finally discuss further random oracle reductions between common signature schemes like GQ, PSS, and FDH.

Keywords Random Oracle Model, Uninstantiability, Diffie Hellman, Encryption.

1 Introduction

Suppose you have a cryptographic scheme A which can be shown to be secure in the random oracle model [4] under some assumption $\mathbb{A}$, say, the RSA assumption. Assume furthermore that someone presents to you a scheme B for the same purpose which is also secure in the random oracle, but now under the potentially weaker assumption $\mathbb{B}$ like factoring. Clearly, if it was not for the random oracle,

and scheme B would also improve over A in other relevant aspects like efficiency, then scheme B should be preferred. Unfortunately, the random oracle model introduces some undesirable uncertainty when simply following the strategy of picking the scheme with the weaker assumption.

Formally, proofs in the random oracle model (ROM) all rely on equally powerful random hash functions, but very often the *exact* requirements for the hash functions to conduct a security proof for a scheme remain unclear. This is all the more true since the random oracle in some schemes is uninstantiable in the sense that no efficient hash function can securely replace the random oracle [9]. For our example of schemes A and B above this means that scheme B may rely on a weaker assumption $\mathbb{B}$, but the actual requirements on the hash function may be much stronger than the ones for A. In the extreme, the hash function in scheme B may be uninstantiable, whereas the hash function for A may rely on a very mild cryptographic assumption like collision-resistance (albeit no proof has been found for this so far).

A natural approach to overcome the problem would be to determine the exact requirements on the hash function and to show that scheme B also relies on weaker assumptions for the hash function than scheme A. However, pinning down these properties of random oracles is often tedious and does not yield the desired result, especially since one would also need to show that the properties are necessary. One example are the hash function properties for OAEP, where Boldyreva and Fischlin [6,7] and later Kiltz et al. [19] gave necessary and, for much weaker security notions than IND-CCA, sufficient conditions on the hash function (in combination with further assumption about the underlying trapdoor permutation). None of these results, however, shows the desired kind of strong security. To complement these results, Kiltz and Pietrzak [20] claimed that for arbitrary trapdoor permutations the hash function in OAEP cannot be instantiated securely to derive IND-CCA security. The latter result is not known to be applicable to specific trapdoor permutations like RSA, though.

Random Oracle Reducibility. The strategy we suggest here is based on the classical reductionist approach to relate cryptographic assumptions: Show that any hash function H, ranging from efficient instantiations to random oracles, which makes scheme A secure under assumptions $\mathbb{A}$ also makes scheme B secure under assumptions $\mathbb{B}$. Technically, this seems to be too optimistic because hash functions in different schemes often cannot be used unchanged but rely on different domains, ranges etc. We thus allow for a "structural" transformation T^H of H for scheme B, possibly depending on the specific hash function. There are three possibilities to relate the hash functions in the schemes:

Definition 1 (Random Oracle Reducibility — Informally). *Let $\mathbb{A}$ and $\mathbb{B}$ be some sets of assumptions. A random oracle in scheme B strictly resp. strongly resp. weakly reduces to the random oracle in scheme A if for every hash function H there exists a transformation T such that*

$$\textit{strictly: } A^H \textit{ secure under } \mathbb{A} \Longrightarrow \textit{scheme } B^{T^H} \textit{ secure under } \mathbb{B}$$

$$\textit{strongly: } A^H \textit{ secure under } \mathbb{A} \Longrightarrow \begin{cases} \textit{scheme } B^{T^H} \textit{ secure under } \mathbb{A} \cup \mathbb{B} \\ \textit{and } B^{T^{H'}} \textit{ secure under } \mathbb{B} \textit{ for some } H' \end{cases}$$

$$\textit{weakly: } A^H \textit{ secure under } \mathbb{A} \Longrightarrow \textit{scheme } B^{T^H} \textit{ secure under } \mathbb{A} \cup \mathbb{B}$$

Several details are hidden in this informal definition, of course, e.g., what a "secure" scheme is, which properties the transformation must satisfy, or how cryptographic assumptions of hash function instantiations are dealt with. We fill in these details in the formal definition and keep it informally for now. We note, however, that the formal definition covers *any* type of hash function, i.e., both oracle-based ones, as well as keyed hash functions with succinct descriptions, or mixtures thereof. In particular, the security of scheme B in the strong case may only be known for a random oracle H'.

For identical assumptions $\mathbb{A} = \mathbb{B}$, or even if $\mathbb{A} \subseteq \mathbb{B}$, all three notions coincide. The difference can be best explained for the case $\mathbb{B} \subsetneq \mathbb{A}$, i.e., that the assumptions $\mathbb{A}$ are strictly stronger than $\mathbb{B}$. The strict notion can in this case be put informally as saying "Scheme B is strictly superior to scheme A in regard of the assumptions, even for the hash function." The strong and presumably more accessible approach can be described as "Scheme B is at least as good as scheme A in regard of the assumptions, but potentially superior." The weak case says that "Scheme B is at least as good as scheme A." In terms of security assumptions it seems that the strict and strong versions are the interesting ones (hence the names); the weak version does not provide any potential improvement concerning the assumption. We note that in the strong case often a security proof for scheme B in the random oracle model can be given without assuming security of A. We merely introduced the dependence via the prerequisite of the implication to make the notions comparable.

As a first sanity check note how previous uninstantiability results relate to either kind of definition. If the hash function security of B can be (weakly, strongly, or strictly) reduced to the hash function security of A and B turns out to be uninstantiable, then this also follows for scheme A (else T^H would be a valid instantiation for B). In this regard the reduction approach also allows to extend uninstantiability results without directly showing the ineffectiveness of efficient hash functions. Vice versa, any new result about secure instantiations of A would immediately transfer to B. Also, uninstantiability immediately implies that there are schemes A (allowing efficient instantiations) and B (being uninstantiable) such that the random oracle for B does not (even weakly) reduce to the one for scheme A.

Example: Hashed ElGamal Encryption. To show that the strong approach is applicable and the definition non-trivial we discuss the case of Hashed ElGamal encryption [1] and its chosen-ciphertext security proof under the strong Diffie-Hellman (DH) assumption in the random oracle model [12]. Here the strong DH assumption says that computing DH keys is infeasible even if given (restricted) access to a decisional DH oracle. Cash et al. [10] present a variant which can be shown to be CCA secure under the (regular) DH assumption in the random

oracle model. This is a clear example of two schemes where the variant seems to improve over the original one in terms of assumption, but where this conclusion is technically not known to be sound because of the random oracle model.

The original hashed ElGamal encryption scheme encrypts a message under public key $X = g^x$ as (Y, c), where $Y = g^y$ and $c = \mathsf{Enc}(k, m)$ for the hashed Diffie-Hellman key $k = H(Y, X^y)$ and the symmetric encryption scheme Enc. The variant in [10] instead computes two related ephemeral Diffie-Hellman keys from public keys $X_0 = g^{x_0}$ and $X_1 = g^{x_1}$, and derives a ciphertext (Y, c) for $Y = g^y$ and $c = \mathsf{Enc}(k, m)$ for $k = H(Y, X_0^y, X_1^y)$. We show (for a slight derivate of the scheme in [10]) that the random oracles can be strongly reduced to the one of hashed ElGamal. Ciphertexts in our variant are defined as

$$(Y, c, k_1), \quad \text{where } Y = g^y,\ c = \mathsf{Enc}(k_0, m) \text{ for } k_0 = H(Y, X_0^y),\ k_1 = H(Y, X_1^y),$$

i.e., we split the hashing into two evaluations, one for each public key part, and use the second key as a kind of confirmation that the first key is computed correctly. We can view this as the transformation

$$T^H(Y, Z_0, Z_1) = H(Y, Z_0)||H(Y, Z_1)$$

and where we use a special symmetric encryption scheme where the key part k_1 is output in clear.

We then prove that IND-CCA security for hashed ElGamal implies security of (our variant) of the twin DH scheme *for any hash function* H for the same assumptions that the hashed ElGamal is secure for. We also show that our variant is secure in the random oracle model assuming only the assumptions given in [10]. It follows that the random oracle in our scheme is strongly reducible to the one of hashed ElGamal.

Note that yet another hashed ElGamal scheme, related to the original scheme, has been shown to be uninstantiable [2]. The scheme differs in two important aspects from our scheme, though. First, their hashed ElGamal encryption does not use randomness and is thus deterministic. Second, the security notion considered in [2] is *IND-CCA-preservation* which gives the adversary simultaneously access to the algorithms of the public-key scheme and the symmetric scheme involving secret keys. In contrast, we use the standard notion of IND-CCA security for the hybrid (public-key) scheme.

We note that the security reduction for our variant to the underlying primitives like the Diffie-Hellman problem for random oracle H' is looser than the one in [10] in terms of concrete bounds.[1] At the same time our scheme relates the random oracle to the one in the original scheme. Of course, concreteness of security bounds is another important aspect, besides efficiency, when considering random oracle reducibility. In principle, it could be incorporated as an explicit requirement in the notion. We relinquish to do so because both aspects, tightness and efficiency, depend to some extend on the individual willingness to pay for the additional security guarantees through the random oracle reducibility.

[1] Note that both proofs are in the random oracle model where concrete bounds must be taken with a grain of salt anyway.

Reductions for Signature Schemes. We give further examples of the applicability of the notion of random oracle reducibility by considering common signature schemes like Guillou-Quisquater [17] or PSS [5] and showing that the random oracle of a probabilistic version of FDH [11] (Full-Domain Hash) reduces to the random oracles in these schemes. However, note that FDH signatures are only known to be uninstantiable according to [14,13] for plain hash evaluations over the message (i.e., no randomness). The first result only applies to random trapdoor permutations (i.e., the result is not known to apply to RSA), and the second more recent result holds when RSA is treated as a black-box group. Any progress in terms of uninstantiability of FDH signatures to our probabilistic case would thus immediately allow to conclude that the Guillou-Quisquater signature scheme and the PSS scheme are uninstantiable. This would somehow extend the uninstantiability result of Goldwasser and Kalai [16] about general (and somewhat contrived) Fiat-Shamir schemes to the "more natural" species.

We discuss another random oracle reduction of (probabilistic) BLS signatures [8] to the Schnorr signature scheme [25]. In this case, however, we need a non-standard assumption to make the reduction work, namely, the knowledge of exponent assumption KEA1 [18,3] which roughly says that when complementing a value X to a Diffie-Hellman tuple $(X, Y, \mathrm{DH}(X, Y))$ one must know the discrete logarithm y of Y. For the random oracles this means that, if our version of the BLS scheme is uninstantiable, then so is the Schnorr signature scheme, or the KEA1 assumption is false.[2]

Some Words of Caution. Just as reductions between number-theoretic assumptions merely relate problems like factoring and RSA, but do not touch the question if RSA is really hard, a reduction for random oracles does not mean that scheme B, in and of itself, is secure (under assumptions $\mathbb{B}$) or that the hash function can be securely instantiated. The reduction only says that scheme B can be made as secure as scheme A in regard of the hash function. Since we do not put any formal prerequisite about the security of scheme A, which could thus be insecure, the reduction could potentially be trivial.

However, as for relating number-theoretic assumptions, where the stronger assumption is usually accompanied by some hardness analysis, scheme A typically comes with some form of security guarantee. Often, this is at least a security proof in the random oracle model, or sometimes for a relaxation thereof like non-programmable random oracles [24,15], traceable random oracles [23], or leaky random oracles [26]. The advantage of our approach is that it follows immediately that B can also be shown secure under the corresponding assumption about the hash function.

One caveat is that the transformed hash function T^H, unlike random oracles, obeys some structure, as the "split" evaluation in our ElGamal example. Hence, when instantiated with some efficient hash function h, scheme B could become insecure for the transformed hash function T^h, despite the reduction and a proof

[2] To be precise need the KEA1 assumption to hold even if one can get additional Schnorr signatures under the key X.

that T^H makes B secure for random oracle H. Noteworthy, at the same time B could be secure when instantiated with h directly, instead of going through the transformation T! We observe, however, that this is an inherent limitation of the random oracle model: It solely provides a heuristic which does not allow to conclude security under concrete instantiations. Our approach at least gives some confidence in the choice of the hash function in the sense that the security is at least as good as the one of another, hopefully well-examined scheme.

2 Random Oracle Reducibility

Hash Functions. We consider families $\mathcal{H}$ of hash functions H where it is understood that H is (not necessarily efficiently) sampleable from $\mathcal{H}$ according to a security parameter λ. It is thus also clear that a hash function H may have a restricted input or output length, depending on λ. We write $H \leftarrow \mathcal{H}(1^\lambda)$ for the sampling. For example, to model a random oracle we let $\mathcal{H}(1^\lambda)$ be the family of all functions with the specified domain and range and the sampling picks a random function from this set. In the sequel we usually simply identify the hash function $H(\cdot)$ with its description H itself. We assume that hash functions are deterministic in the sense that, once a hash function has been sampled, its behavior is fixed. A hash function family may rely on some cryptographic assumptions $\mathbb{H}$; in case of random oracles no assumption $\mathbb{H}$ is necessary as the sampling of a random function already provides all desirable security properties.

Given a hash function H for a scheme A we write A^H for the scheme where each party or algorithm gets oracle access to H. Furthermore, the hash function H may include a public description part which is then also given to all parties and algorithms as additional input. This public part may be for example the full description of H, or only parts thereof, e.g., if H is a hybrid between a random oracle and a keyed hash function. A hash function family is *efficient* if it follows the usual notion of an efficient keyed hash function, i.e., the sampling is efficient, a sampled function H is efficiently computable and entirely described through a public part.

Transformations. A hash function H used in a cryptographic scheme A may not be immediately applicable to another scheme B for the mere fact that the domain and range do not fit. We therefore "slot in" a transformation algorithm T in, such that scheme B then uses the hash function T^H (with the semantic that any algorithm or party gets public descriptions of T^H as additional input). We write $T^{\mathcal{H}}$ for the corresponding hash function family (described by sampling $H \leftarrow \mathcal{H}(1^\lambda)$ and evaluating T^H). Ideally, the transformation should only make structural modifications (like adapting the domain and range) and should be deterministic.

There is, however, one technical subtlety concerning the statefulness of transformations. Namely, we explicitly require that the transformation is stateless. The reason is that if we would allow stateful transformations, it is possible to construct particular contrived transformations that trivialize our notion, in the

sense that some scheme B always reduces to any scheme A. To get a sense of the problem, consider arbitrary schemes A and B which are secure in the random oracle model. Let T denote the stateful transformation which ignores its oracle H and (efficiently) implements a random oracle via lazy sampling. Since B^{T^H} is then clearly secure, scheme B is —as per Definition 1— reducible to scheme A, despite the arbitrary choice of the two schemes. A similar issue arises with instantiations: Suppose scheme B is now instantiable for some hash function family $\mathcal{H}$. Construct the transformation T which again ignores its oracle and instead initially samples a function $H' \leftarrow \mathcal{H}$ and answers subsequent queries according to H'. Again, scheme B remains secure and reduces to any scheme A. Both examples rely on a stateful transformation function —to answer consistently and to remember the choice of the hash function, respectively. Thus, in order to rule out such trivial cases, we ask that transformations are always stateless. This may seem overly restrictive at first glance, but, in fact, is easily justified because hash functions are inherently stateless entities.

One can nonetheless allow for rather general transformations, possibly even considering transformations which themselves rely on assumptions $\mathbb{T}$.

Security of Schemes. We consider security of schemes to be defined via a general notion of games, albeit our games also allow to state simulation-based security properties (by saying that the game returns 1 iff for any simulator there exists a successful distinguisher). As we can subsume several games like the ones for blindness and unforgeability for blind signature schemes into a single game, with corresponding sub games for which the adversary initially decides to mount the attack against, we consider a single game G only. We let $\mathsf{Adv}(\mathcal{A}, G)$ denote the *advantage* of adversary $\mathcal{A}$ playing game G, i.e., the adversary's success probability of winning the game. This makes Adv an implicit part of G. Here, in decisional games the advantage usually denotes the adversary's success probability minus the trivial guessing probability of $1/2$, and in computational games the advantage is usually the adversary's probability of computing a solution.

Analogously to hash functions, we write G^H for a security game in which all parties and algorithms get access to H in the same manner. It is understood that the choice of $H \leftarrow \mathcal{H}(1^\lambda)$ is part of the security game. We write $G^{\mathcal{H}}$ for a game for which a hash function is chosen from the family $\mathcal{H}$.

We envision security assumptions $\mathbb{A}$ for a scheme A as a set of elementary properties such as unforgeability of an underlying MAC or number-theoretic assumptions like the hardness of factoring. We can then apply common set operations and relations to assumptions in a well-defined way, e.g., $\mathbb{A} \cup \mathbb{B}$ comprises all assumptions stated in $\mathbb{A}$ and $\mathbb{B}$, and $\mathbb{B} \subseteq \mathbb{A}$ means that assumptions in $\mathbb{B}$ hold if $\mathbb{A}$ holds. This approach is too applicable for the hash function assumptions $\mathbb{H}$ and possibly the transformation assumptions $\mathbb{T}$. We also assume that assumptions are "opt-in", i.e., need to specified in the set, or else the assumption does not hold. Formally we can define this by considering a universe $\mathbb{U}$ of assumptions and say that any assumption in $\mathbb{U} \setminus \mathbb{A}$ is false.

Note that we keep the formal specifications of games and assumptions at a minimal level. This is possible as we later demand random oracle reducibility

with respect to specific games and assumptions. It is thus up to the reduction statement to consider "reasonable" games and assumptions. We only need very limited syntactical requirements here and can, for example, even allow conflicting assumptions in $\mathbb{A} \cup \mathbb{B}$ (in which case, however, the claims usually become trivial).

Definition 2 (Game-based Security). *Let A denote a cryptographic scheme and $G_A^{\mathcal{H}}$ an associated security game for hash family $\mathcal{H}$. Scheme A is called* $G_A^{\mathcal{H}}$-secure under assumptions $\mathbb{A}$ *for hash family $\mathcal{H}$ relying on assumptions $\mathbb{H}$ if for any efficient adversary $\mathcal{A}$ we have that $\mathsf{Adv}(\mathcal{A}, G_A^{\mathcal{H}}) \approx 0$ is negligible in the security parameter, where the probability is over all random choices of the game (including the choice of the hash function), the algorithms, and the adversary.*

As an example consider the IND-CCA security game for an encryption scheme A (in the random oracle model), in which the game $G_A^{\mathcal{H}}$ proceeds in stages where $\mathcal{A}$ in the first phase receives a public key (in case of an asymmetric scheme) and gets access to a decryption oracle plus the random oracle, then outputs a pair of equal-length messages m_0, m_1 to receive a *single* challenge ciphertext of m_b for secret random bit b, and finally continues asking decryption queries except for the challenge ciphertext. The adversary wins if correctly predicts b, and the advantage of the adversary is the probability for a correct prediction minus $1/2$. In the notation above an IND-CCA secure encryption scheme relying on some cryptographic assumption $\mathbb{A}$ is $G_A^{\mathcal{H}}$-secure under $\mathbb{A}$ for random oracle $\mathcal{H}$.

Random Oracle Reducibility. As explained in the introduction we introduce a weak, strong, and strict notion of random oracle reducibility:

Definition 3 (Random Oracle Reducibility). *Let A be a cryptographic scheme with security game G_A and assumptions $\mathbb{A}$, and let B be a cryptographic scheme with game G_B and assumptions $\mathbb{B}$. Then the random oracle in scheme B (strictly resp. strongly resp. weakly)* reduces *to the random oracle in scheme A if for every hash function family $\mathcal{H}$ relying on assumptions $\mathbb{H}$ there exists a stateless transformation T such that*

$$\textit{strict: } A \textit{ is } G_A^{\mathcal{H}}\textit{-secure under } \mathbb{A} \Rightarrow B \textit{ is } G_B^{T^{\mathcal{H}}}\textit{-secure under } \mathbb{B}$$

$$\textit{strong: } A \textit{ is } G_A^{\mathcal{H}}\textit{-secure under } \mathbb{A} \Rightarrow \begin{cases} B \textit{ is } G_B^{T^{\mathcal{H}}}\textit{-secure under } \mathbb{A} \cup \mathbb{B} \textit{ and} \\ B \textit{ is } G_B^{T^{\mathcal{H}'}}\textit{-secure under } \mathbb{B} \textit{ for some } \mathcal{H}' \\ \qquad\qquad\qquad\qquad\qquad\qquad \textit{relying on } \mathbb{H}' \end{cases}$$

$$\textit{weak: } A \textit{ is } G_A^{\mathcal{H}}\textit{-secure under } \mathbb{A} \Rightarrow B \textit{ is } G_B^{T^{\mathcal{H}}}\textit{-secure under } \mathbb{A} \cup \mathbb{B}$$

We say that $(B, G_B^{\mathcal{H}}, \mathbb{B})$ is (weakly or strongly or strictly) random oracle reducible *to $(A, G_A^{\mathcal{H}}, \mathbb{A})$. It is* polynomial-time (weakly or strongly or strictly) random oracle reducible *if it is random oracle reducible via (deterministic) stateless polynomial-time transformations T for any hash function family $\mathcal{H}$.*

We occasionally simply say that B is random oracle reducible (RO-reducible) to A if the games and assumptions are clear from the context.

Some remarks about the definition and variations follow:

– The above does not rule out trivial examples where scheme B actually relies on stronger assumptions $\mathbb{B}$ than scheme A, e.g., if $\mathbb{A}$ is a subset of $\mathbb{B}$. As explained in the introduction, the most interesting examples seem to be the ones where assumptions $\mathbb{B}$ are weaker than $\mathbb{A}$ or at least incomparable. Occasionally, however, one may be interested in a scheme B which requires stronger assumptions $\mathbb{B}$ but which is more efficient (or has other desirable properties).
– We can devise stronger notions concerning the order of quantification for our reducibility notion. Above, the transformation can depend on the specific hash function family $\mathcal{H}$, and thus possibly specific properties of $\mathcal{H}$. One could alternatively demand that the transformation needs to be universal in the sense that it works for any $\mathcal{H}$.
– The above definition assumes that transformation T does not rely on additional assumptions. More generally, one could specify assumptions $\mathbb{T}$ and say that scheme B is secure under assumptions $\mathbb{B}' = \mathbb{B} \cup \mathbb{T}$.
– According to our syntax the adversary $\mathcal{B}$ in game G_B with the transformed random oracle would get access to T^H, but not H itself. This can be easily patched by letting the transformation T give direct access to H through a special query mode.

3 Basic Results

Relating the Reducibility Notions. We first show that strict reducibility implies strong reducibility which implies weak reducibility. The proof is rather syntactical and omitted for space reasons.

Proposition 1 (Strict ⇒ Strong ⇒ Weak Reducibility). *Let A be a cryptographic scheme with security game G_A and assumptions $\mathbb{A}$, and let B be a cryptographic schemes with game G_B and assumptions $\mathbb{B}$. If the random oracle in scheme B strictly reduces to the random oracle in scheme A, then it also strongly reduces to it. If it strongly reduces to it, then it also weakly reduces to it.*

We next discuss a scheme which supports a strong reduction, but not a strict one. Note that for $\mathbb{A} \subseteq \mathbb{B}$ this claim would be trivial because then the notions coincide. Instead, our separation example even holds for $\mathbb{B} \subsetneq \mathbb{A}$.

Proposition 2 (Strong ⇏ Strict Reducibility). *There exists schemes A, B for games $G_A^{\mathcal{H}}$ and $G_B^{\mathcal{H}}$ and assumptions $\mathbb{A}, \mathbb{B}$ such that $\mathbb{B} \subsetneq \mathbb{A}$, and the random oracle of B strongly reduces to the one of scheme A, but not strictly.*

Proof. Let scheme A run two copies of Lamport's one-time signature scheme [21], one based on an alleged one-way function f, and the other one by using the given hash function (oracle). Verification checks if both signatures are valid. Let $G_A^{\mathcal{H}}$ be the standard unforgeability game for one-time signature schemes, and let $\mathbb{A}$ be the assumption that an underlying function f is really one-way. Let B and $G_B^{\mathcal{H}}$ be the same scheme and game, but let $\mathbb{B}$ be the empty set.

Consider the hash function family $\mathcal{H}$ which samples trivial functions $H : \{0,1\}^* \to \{0\}$ only and where $\mathbb{H}$ is empty. Then scheme A is still unforgeable if f is one-way, independently of the H-part of the signature. In contrast, B would be insecure under $\mathbb{B}$ and for the trivial hash function family, because, by assumption about the "minimalistic" approach for the set $\mathbb{B}$, the function f is not one-way then. Hence, the random oracle in B cannot be strictly reduced to the one in A.

Finally, note that for a hash function family $\mathcal{H}'$ which is one-way the signature scheme B becomes secure even under $\mathbb{B}$, because any forger would need to forge the one-time signature scheme for the hash function. At the same time, for any hash function family scheme B is secure under $\mathbb{A} \cup \mathbb{B}$. These two properties show that the random oracle in B strongly reduces to the one in A. □

For the next separation we further need to exclude contrived examples where the hash function assumptions $\mathbb{H}$ "makes up" for assumptions in $\mathbb{A} \setminus \mathbb{B}$ to make scheme B secure. We say that $\mathbb{H}$ is *non-interfering* with $\mathbb{A}$ and $\mathbb{B}$ iff $\mathbb{H} \cap (\mathbb{A} \setminus \mathbb{B}) = \emptyset$. In this case we say that the random oracle in scheme B reduces to the one in scheme A under non-interfering hash assumptions if reducibility holds for all hash function families $\mathcal{H}$ with non-interfering assumption $\mathbb{H}$.

Proposition 3 (Weak $\not\Rightarrow$ Strong Reducibility). *There exists schemes A, B for games $G_A^{\mathcal{H}}$ and $G_B^{\mathcal{H}}$ and assumptions $\mathbb{A}, \mathbb{B}$ such that $\mathbb{B} \subsetneq \mathbb{A}$, and the random oracle of B weakly reduces to the one of scheme A, but not strongly for non-interfering hash functions.*

Proof. Consider again Lamport's one-time signature scheme as scheme A, relying on a one-way function f (whose one-wayness is postulated in $\mathbb{A}$). The scheme ignores the hash function. Let $G_B^{\mathcal{H}}$ be again the unforgeability game for one-time signature schemes. Let B the same scheme with the same security game, but let $\mathbb{B}$ be empty.

Any hash function makes both schemes secure under assumptions $\mathbb{A} \cup \mathbb{B}$ such that the (irrelevant) random oracle of B weakly reduces to the one of A. Since B cannot be secure assuming only $\mathbb{B}$, because the hash function cannot include the assumption about the one-wayness of f by the non-interference, the scheme cannot strongly reduce the random oracle. □

Uninstantiability Implications. In this section we briefly show fundamental results about (un)instantiable random oracles. We define uninstantiability with respect to a very loose requirement on the assumptions, leaving it up to the reduction statement to consider only "standard" cryptographic assumptions in $\mathbb{A}$ and $\mathbb{B}$.

Definition 4 (Uninstantiability). *Let A be $G_A^{\mathcal{H}}$-secure under assumptions $\mathbb{A}$ for random oracle $\mathcal{H}$. Then the random oracle is* uninstantiable *for $G_A^{\mathcal{H}}$ and $\mathbb{A}$ if for any efficient hash function family $\mathcal{H}$ with assumption $\mathbb{H}$ the scheme A is not $G_A^{\mathcal{H}}$-secure under assumptions $\mathbb{A}$.*

Proposition 4 (B uninstantiable $\Rightarrow$ A uninstantiable). *Assume that scheme B with game $G_B^{\mathcal{H}}$ and assumptions $\mathbb{B}$ is (strictly resp. strongly resp. weakly) polynomial-time RO-reducible to scheme A for $G_A^{\mathcal{H}}$ and (true) assumptions $\mathbb{A}$. If B is uninstantiable for $G_B^{\mathcal{H}}$ under $\mathbb{B}$ (for strict reductions) resp. $\mathbb{A} \cup \mathbb{B}$ (for strong or weak reduction), then so is A for $G_A^{\mathcal{H}}$ and assumptions $\mathbb{A}$.*

The proof is again rather straightforward from the definitions. It is clear that this, vice versa, implies that any instantiability result about A transfers accordingly to B.

Given the uninstantiability notion we next note that there are schemes for which the random oracles are not (even weakly) reducible to each other:

Proposition 5 (Impossibility of Reducibility). *There exists schemes A, B for games $G_A^{\mathcal{H}}$ and $G_B^{\mathcal{H}}$ and (true) assumptions $\mathbb{A}, \mathbb{B}$ such that the random oracle of B does not support a weak or strong or strict polynomial-time reduction to the one of scheme A, even though B is secure in the random oracle model.*

The proof appears in the full version of the paper.

4 Example: Hashed ElGamal

In this section we show that the hash function in (a variant) the Twin Diffie-Hellman encryption scheme is RO-reducible to the hash function in hashed ElGamal. We remark that we are not aware if the original twin DH scheme allows the same reduction.

Hashed ElGamal. We first review the classical hashed ElGamal-encryption scheme as presented in [1]. This scheme, denoted by $A = (\mathsf{KGen}_A, \mathsf{Enc}_A, \mathsf{Dec}_A)$ is based on the Diffie-Hellman problem and uses a hash function H and a symmetric cipher $(\mathsf{Enc}, \mathsf{Dec})$. Specifically,

Construction 1 (Hashed ElGamal Encryption Scheme). *The hashed ElGamal encryption scheme $A = (\mathsf{KGen}_A, \mathsf{Enc}_A, \mathsf{Dec}_A)$ in the ROM is defined as follows:*

$\mathsf{KGen}_A(\lambda)$	$\mathsf{Enc}_A(pk, m)$	$\mathsf{Dec}_A(sk, Y, c)$
pick $(\mathcal{G}, g, q)$	$y \leftarrow \mathbb{Z}_q$; $Y \leftarrow g^y$	$Z \leftarrow Y^x$
$x \leftarrow \mathbb{Z}_q$; $X \leftarrow g^x$	$Z \leftarrow X^y$; $k \leftarrow H(Y, Z)$	$k \leftarrow H(Y, Z)$
$sk \leftarrow x$; $pk \leftarrow (\mathcal{G}, g, q, X)$	$c \leftarrow \mathsf{Enc}_k(m)$	$m \leftarrow \mathsf{Dec}_k(c)$
Return (sk, pk)	*Return* (Y, c)	*Return* m

Assuming that the symmetric cipher is secure against chosen ciphertext attacks[3] and that the strong Diffie-Hellman assumption holds (where the adversary has access to a restricted DH decisional oracle), it is proven in [12] that scheme A is secure against chosen ciphertext attacks if H is modeled as a random oracle. The milder ordinary DH assumption is not known to be sufficient to prove CCA security, since the attacker obtains a decision oracle through the decryption oracle here, such that some information about the key may be leaked.

[3] We always refer to attacks involving a single challenge only throughout the paper.

Twin DH Scheme. Subsequently, Cash et al. [10] introduce the so-called strong twin DH assumption which holds if and only if the regular DH assumption holds. Their corresponding DH problems are equally hard but the twin case includes access to a decision oracle. This enables a clean security proof for a variant of the hashed ElGamal scheme, because the decryption oracle is not more powerful than the decision oracle in the strong twin DH case. Thus, the twin ElGamal scheme allows for milder number-theoretic assumptions while preserving CCA security.

However, the random oracle in the twin ElGamal scheme is used slightly differently than in the original scheme: Its domain is the set of group element triples, as opposed to tuples in the original scheme. While this is unproblematic in the ROM for hash functions $H : \{0,1\}^* \to \{0,1\}^\lambda$ with arbitrary input length, the implications for other security properties for instantiations are less clear. For example, it may be that the twin Diffie-Hellman scheme demands stronger properties from the hash function. We show via our notion of RO-reducibility that this is not the case, at least for our slight variation:

Construction 2 (Twin Diffie-Hellman Encryption Scheme). *The twin DH encryption scheme* $B = (\mathsf{KGen}_B, \mathsf{Enc}_B, \mathsf{Dec}_B)$ *in the ROM is defined as follows:*

$\mathsf{KGen}_B(\lambda)$
pick $(\mathcal{G}, g, q)$
$x_0 \leftarrow \mathbb{Z}_q$; $X_0 \leftarrow g^{x_0}$
$x_1 \leftarrow \mathbb{Z}_q$; $X_1 \leftarrow g^{x_1}$
$sk \leftarrow (x_0, x_1)$
$pk \leftarrow (\mathcal{G}, g, q, X_0, X_1)$
Return (sk, pk)

$\mathsf{Enc}_B(pk, m)$
$y \leftarrow \mathbb{Z}_q$; $Y \leftarrow g^y$
$Z_0 \leftarrow X_0^y$; $k_0 \leftarrow H(Y, Z_0)$
$Z_1 \leftarrow X_1^y$; $k_1 \leftarrow H(Y, Z_1)$
$c \leftarrow \mathsf{Enc}_{k_0}(m)$
Return (Y, c, k_1)

$\mathsf{Dec}_B(sk, Y, c, k_1)$
$Z_0 \leftarrow Y^{x_0}$
$Z_1 \leftarrow Y^{x_1}$
$k_0 \leftarrow H(Y, Z_0)$
$m \leftarrow \mathsf{Dec}_{k_0}(c)$
if $k_1 \neq H(Y, Z_1)$
set $m \leftarrow \perp$
Return m

We can view the transformation $T^H : \mathcal{G}^3 \to \{0,1\}^{2\lambda}$ of the hash function $H : \mathcal{G}^2 \to \{0,1\}^\lambda$ as follows:

$$T^H(Y, Z_0, Z_1) = H(Y, Z_0) || H(Y, Z_1).$$

Splitting the actual encryption of the message into an encryption for one key half and where we output the other half in clear can then be seen as a special encryption scheme (with double-length keys).

RO-Reducibility. We first show that our twin DH scheme *weakly* reduces the random oracle to the one of the hashed ElGamal scheme for IND-CCA security, i.e., assuming the strong DH assumption. We discuss afterward that the scheme is also secure in the random oracle model assuming the regular DH assumption, implying that the reducibility is also strong:

Theorem 3. *Consider the hashed ElGamal encryption scheme for the IND-CCA security game and the assumptions* $\mathbb{A}$ *that the symmetric encryption scheme*

is IND-CCA secure and the strong DH assumption holds. Then the twin DH encryption scheme B with the IND-CCA security game and the assumptions $\mathbb{B}$ that the symmetric encryption scheme is IND-CCA secure and that the DH assumption holds, is strongly RO-reducible to the hashed ElGamal encryption scheme via $T^H(Y, Z_0, Z_1) = H(Y, Z_0)||H(Y, Z_1)$.

The proof follows from the following two propositions.

Proposition 6. *Under the assumptions as in Theorem 3 the twin DH encryption scheme is weakly RO-reducible to the hashed ElGamal encryption scheme.*

Proof. Assume towards contradiction that there exists an algorithm $\mathcal{B}$ that breaks the CCA-security of B. We then describe an adversary $\mathcal{A}$ that breaks the CCA-security of A. This adversary essentially simulates the "second key half" of the scheme by itself.

Description. To initialize the simulation adversary $\mathcal{A}$ on input $(\mathcal{G}, g, q, X_0) = (\mathcal{G}, g, q, g^{x_0})$ chooses the other half of the secret key $x_1 \leftarrow \mathbb{Z}_q$ and calculates the corresponding public key $X_1 \leftarrow g^{x_1}$. Adversary $\mathcal{A}$ next runs adversary $\mathcal{B}$ with input $(\mathcal{G}, g, q, (X_0, X_1))$ and answers $\mathcal{B}$'s oracle queries as follows:

- First, $\mathcal{A}$ translates any hash query $H(A, B, C)$ from $\mathcal{B}$ into two queries to $\mathcal{A}$'s own hash oracle. More precisely, $\mathcal{A}$ answers an (A, B, C) query with $(H(A, B), H(A, C)) = T^H(A, B, C)$.
- In order to answer $\mathcal{B}$'s challenge query (m_0, m_1), the adversary submits (m_0, m_1) to his own challenge oracle and parses the corresponding ciphertext answer as (Y, c). It remains to compute the extra value by re-using the randomness Y obtained from the oracle. Adversary $\mathcal{A}$ thus computes $k_1 = H(Y, Y^{x_1}) = H(Y, X_1^y)$ and finally returns the ciphertext (Y, c, k_1) to $\mathcal{B}$.
- On a decryption query (Y, c, k_1) of $\mathcal{B}$ adversary $\mathcal{A}$ first checks if (Y, c) corresponds to the value in the challenge ciphertext, or if $k_1 \neq H(Y, Y^{x_1})$. If so, then $\mathcal{A}$ immediately returns $\perp$. Else $\mathcal{A}$ asks its own decryption oracle for the decryption m of (Y, c). To answer the query, it then returns m.
- Note also that we can grant $\mathcal{B}$ direct access to the H oracle. Adversary $\mathcal{A}$ would simply forward this query and hand back the answer.

When $\mathcal{B}$ eventually outputs a guess b then $\mathcal{A}$ outputs the same bit.

Analysis. The simulation is perfect in the following sense: $\mathcal{B}$ cannot submit a ciphertext (Y, c, k_1^*) to the decryption oracle (after receiving the challenge ciphertext (Y, c, k_1)) for $k_1^* \neq k_1$ which would decrypt correctly. Hence, $\mathcal{A}$ can reject such ciphertexts immediately and therefore only submits "pruned" ciphertexts to its decryption oracle which have never appeared before. Hence, $\mathcal{A}$, too, represents a successful attacker on the hashed ElGamal scheme if $\mathcal{B}$ is one for the twin DH scheme. Moreover, the advantages of both algorithms in their corresponding IND-CCA game are identical. □

To complete the proof for a strong reduction we finally show that our version is secure in the random oracle model:

Proposition 7. *The twin DH encryption scheme B with the assumptions $\mathbb{B}$ that the symmetric encryption scheme is IND-CCA secure and that the DH assumption holds, is IND-CCA-secure in the random oracle model.*

Proof. The proof is more involved that then one in [10], owned to the fact that the random oracle $H(X, Z_0, Z_1)$ in [10] ties together the twin DH tuples and that this property is required for the twin DH oracle. In contrast, in our scheme the pairs (X, Z_0) and (X, Z_1) are only loosely connected. We show that this loose connection can be made a strong one with multiple simulations of the adversary.

In a first step we can "normalize" an adversary $\mathcal{A}$ against IND-CCA of our twin DH scheme. First we may assume that $\mathcal{A}$ never makes a hash query twice. Second, we can assume that $\mathcal{A}$ never submits a tuple (Y_i, c_i, k_i) to the decryption oracle before receiving the challenge ciphertext (Y, c, k) where $Y_i = Y$. This decreases the adversary's success probability by a negligible amount D/q for the polynomial number D of $\mathcal{A}$'s decryption queries. Third, we can assume that adversary $\mathcal{A}$ never submits a decryption request (Y_i, c_i, k_i) such that, in case $Y_i \neq Y$, it has not queried the hash function about $(Y_i, Y_i^{x_1})$ for $Y_i \neq Y$ before. The loss is at most $D \cdot 2^{-\lambda}$ for this. Fourth, we assume that the adversary never submits (Y_i, c_i, k_i) to the decryption oracle where $Y_i = Y$ but $k_i \neq k$; such a query cannot be valid. Fifth, we assume that $X_0 \neq X_1$ which happens with probability $1 - 1/q$.

Taming Hash Queries. Consider a normalized adversary $\mathcal{A}$ against our twin DH scheme. We assume that $\mathcal{A}$ in addition to T^H also has direct access to the random oracle $H : \mathcal{G}^2 \to \{0,1\}^*$. In fact, we assume from now on that all algorithms, including the adversary and the scheme's algorithms, never call T^H, but use H to simulate T^H with two queries. Define the following event HashQuery that, during the IND-CCA attack, $\mathcal{A}$ at some point asks a query (Y, Z) to H such that Y appears in the challenge ciphertext, and $Z = Y^{x_0}$ or $Z = Y^{x_1}$ for the public key entries $X_0 = g^{x_0}$ and $X_1 = g^{x_1}$.

We show that the probability $\epsilon(\lambda)$ of event HashQuery must be negligible. Assume toward contradiction that this was not the case. We then show how to break the the twin DH problem (and thus the DH problem) via algorithm $\mathcal{B}$. Algorithm $\mathcal{B}$ receives a group description $(\mathcal{G}, g, q)$ and values Y, X_0, X_1 as input. It can also query a twin DH oracle about values (g^a, B_0, B_1) which outputs 1 iff $B_0 = X_0^a$ and $B_1 = X_1^a$. The values X_0, X_1 serve as the public key presented to $\mathcal{A}$, and Y will be placed in the challenge ciphertext.

Algorithm $\mathcal{B}$ runs $\mathcal{A}$'s attack by using the input data as the public key, and simulating the random oracle and decryption queries as follows:

- $\mathcal{B}$ will maintain a list $\mathcal{L}$ of tuples of the form (A, B, k) or $(\mathtt{dh}, A, X_b, k)$ where the former type corresponds to direct hash queries of $\mathcal{A}$ and the latter type to implicit hash queries. Initially, $\mathcal{B}$ sets $\mathcal{L} := \{(\mathtt{dh}, Y, X_0, k_0), (\mathtt{dh}, Y, X_1, k_1)\}$ for random values k_0, k_1 for the hash values to compute the challenge ciphertext (note that Y is already known at the outset).

– Whenever $\mathcal{A}$ makes a hash query (A, B) algorithm $\mathcal{B}$ first searches for an entry (A, C, k) in $\mathcal{L}$ such that (A, B, C) or (A, C, B) forms a correct twin DH tuple (under X_0, X_1). Since $X_0 \neq X_1$ only one case can happen. If found, and there exists an entry $(\mathtt{dh}, A, X_0, k)$ in $\mathcal{L}$ for the case (A, B, C) resp. $(\mathtt{dh}, A, X_1, k)$ for the case (A, C, B), then replace this entry by (A, B, k) in $\mathcal{L}$. In any other case, pick k at random and store (A, B, k) in $\mathcal{L}$. Return k.
– If $\mathcal{A}$ makes a decryption request (Y_i, c_i, k_i) then check whether $Y_i = Y$ or not. In case $Y_i = Y$ look up the entry $(\mathtt{dh}, Y, X_0, k_0)$ in $\mathcal{L}$ and use k_0 to decrypt c_i. (Note that, by assumption, k_1 must be correct.) Suppose $Y_i \neq Y$. Then, since the adversary is normalized, there must be an entry (Y_i, Z_1, k_1) in $\mathcal{L}$ already, caused by a hash query, where $Z_1 = Y_i^{x_1}$. (There cannot exist another entry (Y_i, Z_1, k_1') for $k_1' \neq k_1$ as hash queries never repeat.) Given (Y_i, Z_1, k_1) check for an entry (Y_i, Z_0, k_0) such that (Y_i, Z_0, Z_1) forms a valid twin DH tuple for X_0, X_1. If such an entry exist then use k_0 to decrypt c_i. If no such entry exist, check for a tuple $(\mathtt{dh}, Y_i, X_0, k_0)$ in $\mathcal{L}$ and use k_0 to decrypt. Else, pick a new value k_0, store $(\mathtt{dh}, Y_i, X_0, k_0)$ in $\mathcal{L}$, and use k_0 to decrypt. Return the decrypted message.

To prepare the challenge ciphertext $\mathcal{B}$ uses the previously chosen values k_0, k_1 placed in $\mathcal{L}$, also picks one of the two messages m_0, m_1 at random, and returns $(Y, \mathsf{Enc}(k_0, m_b), k_1)$.

If $\mathcal{A}$ finishes algorithm $\mathcal{B}$ records all entries (A, B) in $\mathcal{L}$ with $A = Y$ and now reruns the above procedure, with the same group but for re-randomized data $Y' = Y^s$, $X_0' := X_a^{s_a}$, $X_1' := X_{1-a}^{s_{1-a}}$ for random $s, s_0, s_1 \leftarrow \mathbb{Z}_q^*$ and random bit a. Every other random choice is based on fresh randomness. Any query (A, B, C) to the twin DH oracle in this second run is first transformed into $(A, B^{1/s_0}, C^{1/s_1})$ for $a = 0$ resp. $(A, C^{1/s_0}, B^{1/s_1})$ for $a = 1$. At the end, $\mathcal{B}$ transforms all pairs (A', B') in the list $\mathcal{L}$ of the second run by computing $((A')^{1/s}, (B')^{1/s_0})$ and $((A')^{1/s}, (B')^{1/s_1})$, effectively doubling the number of pairs. Sieve to keep only those with first element Y. Run on all combinations of the two (sieved) lists the twin DH oracle to find a solution (Y, Z_0, Z_1) to the twin DH problem.

Analysis. The maintenance of the hash list $\mathcal{L}$ provides a more fine-grained implementation of how a random oracle would behave: Since any decryption query for $Y_i \neq Y$ must already contain a corresponding entry $(Y_i, Y_i^{x_1}, k_1)$ by assumption, we can check via the twin DH oracle if we already have a matching entry (Y_i, Z_0, k_0). If not, we generate a fresh random string and store the implicit representation $(\mathtt{dh}, Y_i, X_0, k_0)$ in $\mathcal{L}$, and will later carefully check if a hash query for $Y_i^{x_0}$ is made (in which case we update the entry in $\mathcal{L}$ and re-use the value k_0).

As for $\mathcal{B}$'s success probability, we call a group $(\mathcal{G}, g, q)$ good if $\mathcal{A}$'s success probability conditioned on this group exceeds $\epsilon/2$. By an averaging argument a group is good with probability at least $\epsilon/2$. Hence, given such a good group, and the fact that $\mathcal{B}$ provides a perfect simulation, $\mathcal{B}$ obtains a valid entry (Y, Y^{x_0}) or (Y, Y^{x_1}) with probability at least $\epsilon/2$ in the first run. The same applies in the second run where the re-randomization is correctly undone for each twin DH

oracle query. With probability 1/2 algorithm $\mathcal{B}$ then obtains matching values (Y, Y^{x_0}) and (Y, Y^{x_1}) because the order bit a in the second run is information-theoretically hidden from $\mathcal{A}$. Overall, and neglecting the minor loss due to normalization of $\mathcal{A}$, algorithm $\mathcal{B}$ thus solves the twin DH problem with probability at least $\epsilon^3/16$. By assumption this is still non-negligible.

Conditioning on the adversary not making bad hash queries, it is now easy to give a reduction to the IND-CCA security of the symmetric cipher (with the attacker against the symmetric scheme providing all the public-key operations itself). □

5 Reductions among Signature Schemes

In this section we briefly outline a few more applications of our notion. Specifically, we give three relations among signature schemes including the Guillou-Quisquater (GQ) signature scheme [17] which we reduce to a probabilistic version of FDH, the PSS signature scheme [5] which we also reduce to a probabilistic FDH variation, and finally a reduction from Schnorr signatures [25] to a (probabilistic version of) BLS signatures [8].

GQ ⇒ FDH. We first consider the RSA-based Guillou-Quisquater identification scheme and its derived signature scheme via the Fiat-Shamir heuristic [17]. For public key $pk = (X, N, e)$ and secret key x with $X = x^e \bmod N$ the signer computes a signature as (R, y) for random $R = r^e \bmod N$, and where $y = r^c x \bmod N$ for $c = H(pk, R, m)$. A probabilistic full-domain hash (FDH) RSA signature scheme with signatures of the form (R, σ) for $\sigma = (H(pk, R, m))^d \bmod N$ is (strictly) random oracle reducible to the aforementioned Guillou-Quisquater scheme via the transformation $T^H(pk, R, m) = R^{H(pk,R,m)} X \bmod N$ for any type of forgery attack under the RSA assumption. The reason is that any Guillou-Quisquater signature for H can be seen as a FDH signature for $T^H : \{0,1\}^* \to \mathbb{Z}_N^*$, and any successful forgery for the FDH scheme for T^H is vice versa a valid forgery for the Guillou-Quisquater scheme.

PSS ⇒ FDH. The reduction of another probabilistic version of FDH to the PSS signature scheme is similar to the GQ case. Consider FDH signatures $(T^H(r, m))^d \bmod N$ for the PSS-encoding $T^H(r, m) = \texttt{str2int}(0||w||r^*||H_2(w))$ for $w = H_0(r, m)$ and $r^* r \oplus = H_1(w)$. Here, H_0, H_1, H_2 are hash functions derived from H as in the PSS scheme. Then any successful attack on FDH with hash function T^H easily yields a forgery against PSS with hash function H. Hence, PSS allows a strict random oracle reduction to the probabilistic version of FDH under the RSA assumption for any type of forgery attack.

Schnorr ⇒ BLS. Consider a probabilistic version of the BLS signature scheme [8], where signatures are of the form $\sigma = (R, H(R, X, m)^x)$ for randomness R, message m, private key x and public key $X = g^x$. Verification is performed analogously to the original scheme via a pairing computation. We argue that the Schnorr signature scheme (recall that a signature there is of the

form $\sigma = (c, r + cx \bmod q)$ for public key $x = g^x$, $R = g^r$, and $c = H(R, m)$) is (strictly) random oracle reducible to the BLS version via the transformation $T^H(R, X, m) = RX^{H(R,m)}$. This holds assuming the discrete logarithm assumption and under an augmented version of the KEA1 assumption [18,3] which states that, for any adversary $\mathcal{A}$ which for input a description of the group, g, X, and with access to a Schnorr signing oracle under key X and a hash function oracle, outputs a pair (Y, Y^x), there exists an adversary $\mathcal{A}'$ which, on the same input and with access to the same oracles, outputs y with $X^y = Y^x$. The probability that $\mathcal{A}$ succeeds, but $\mathcal{A}'$ does not, must be negligible for all $\mathcal{A}$.

Suppose now that there exists some successful adversary $\mathcal{B}$ against our version of BLS. Construct adversary $\mathcal{A}$ against the Schnorr scheme as follows. Whenever $\mathcal{B}$ makes some query m, adversary $\mathcal{A}$ forwards this query to its own signing oracle. It uses the answer (c, y) to calculate $h = X^y$, computes $R = g^y X^{-c}$ (such that $H(R, m) = c$) and finally answers $\mathcal{B}$'s query with (R, h). This simulates a correct signature since $\mathcal{B}$ expects R and $T^H(R, X, m)^x = (RX^{H(R,m)})^x = (g^y)^x = X^y = h$. It remains to construct a Schnorr forgery from $\mathcal{B}$'s forgery, denoted by (m^*, R^*, Z^*). To this end we note that, under the augmented KEA1 assumption, for $\mathcal{A}$ (running $\mathcal{B}$ as a subroutine) outputting $Y^* = T^H(R^*, X, m^*)$ and $Z^* = (Y^*)^x$ for the valid forgery (m^*, R^*, Z^*), there must exist an adversary $\mathcal{A}'$ returning y^* with $Z^* = X^{y^*}$. This must be true with non-negligible probability, because $\mathcal{A}$ succeeds with non-negligible probability, and otherwise the augmented KEA1 assumption would be false. Hence, there exists an adversary which creates a valid forgery $(m^*, H(R^*, m^*), y^*)$ for the Schnorr scheme with non-negligible probability.

Acknowledgments

We thank the anonymous reviewers for valuable comments, especially Mihir Bellare. We also thank Anja Lehmann, Adam O'Neill, and Tom Ristenpart for listening to this idea and providing feedback at an early stage. Both authors are supported by grants Fi 940/2-1 and Fi 940/4-1 of the German Research Foundation (DFG). This work was also supported by CASED (`www.cased.de`).

References

1. Abdalla, M., Bellare, M., Rogaway, P.: The oracle Diffie-Hellman assumptions and an analysis of DHIES. In: CT-RSA 2001. LNCS, vol. 2020, pp. 143–158.
2. Bellare, M., Boldyreva, A., Palacio, A.: An uninstantiable random-oracle-model scheme for a hybrid-encryption problem. In: EUROCRYPT 2004. LNCS, vol. 3027, pp. 171–188.
3. Bellare, M., Palacio, A.: The knowledge-of-exponent assumptions and 3-round zero-knowledge protocols. In: CRYPTO 2004. LNCS, vol. 3152, pp. 273–289.
4. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: ACM CCS 93. pp. 62–73.
5. Bellare, M., Rogaway, P.: The exact security of digital signatures: How to sign with RSA and Rabin. In: EUROCRYPT'96. LNCS, vol. 1070, pp. 399–416.

6. Boldyreva, A., Fischlin, M.: Analysis of random oracle instantiation scenarios for OAEP and other practical schemes. In: CRYPTO 2005. LNCS, vol. 3621, pp. 412–429.
7. Boldyreva, A., Fischlin, M.: On the security of OAEP. In: ASIACRYPT 2006. LNCS, vol. 4284, pp. 210–225.
8. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the Weil pairing. Journal of Cryptology 17(4), 297–319
9. Canetti, R., Goldreich, O., Halevi, S.: The random oracle methodology, revisited (preliminary version). In: 30th ACM STOC. pp. 209–218.
10. Cash, D., Kiltz, E., Shoup, V.: The twin DiffieHellman problem and applications. Journal of Cryptology 22(4), 470–504
11. Coron, J.S.: On the exact security of full domain hash. In: CRYPTO 2000. LNCS, vol. 1880, pp. 229–235.
12. Cramer, R., Shoup, V.: Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. SIAM Journal on Computing 33(1), 167–226 (2003)
13. Dodis, Y., Haitner, I., Tentes, A.: On the (in)security of rsa signatures. Cryptology ePrint Archive, Report 2011/087 (2011), http://eprint.iacr.org/
14. Dodis, Y., Oliveira, R., Pietrzak, K.: On the generic insecurity of the full domain hash. In: CRYPTO 2005. LNCS, vol. 3621, pp. 449–466.
15. Fischlin, M., Lehmann, A., Ristenpart, T., Shrimpton, T., Stam, M., Tessaro, S.: Random oracles with(out) programmability. In: ASIACRYPT 2010. LNCS, vol. 6477, pp. 303–320.
16. Goldwasser, S., Kalai, Y.T.: On the (in)security of the Fiat-Shamir paradigm. In: 44th FOCS. pp. 102–115. IEEE Computer Society Press
17. Guillou, L.C., Quisquater, J.J.: A practical zero-knowledge protocol fitted to security microprocessor minimizing both trasmission and memory. In: EUROCRYPT'88. LNCS, vol. 330, pp. 123–128.
18. Hada, S., Tanaka, T.: On the existence of 3-round zero-knowledge protocols. In: CRYPTO'98. LNCS, vol. 1462, pp. 408–423.
19. Kiltz, E., O'Neill, A., Smith, A.: Instantiability of RSA-OAEP under chosen-plaintext attack. In: CRYPTO 2010. LNCS, vol. 6223, pp. 295–313.
20. Kiltz, E., Pietrzak, K.: On the security of padding-based encryption schemes - or - why we cannot prove OAEP secure in the standard model. In: EUROCRYPT 2009. LNCS, vol. 5479, pp. 389–406.
21. Lamport, L.: Constructing digital signatures from a one-way function. Technical Report SRI-CSL-98, SRI International Computer Science Laboratory
22. Maurer, U.M., Renner, R., Holenstein, C.: Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In: TCC 2004. LNCS, vol. 2951, pp. 21–39.
23. Naito, Y., Yoneyama, K., Wang, L., Ohta, K.: How to confirm cryptosystems security: The original Merkle-Damgård is still alive! In: ASIACRYPT 2009. LNCS, vol. 5912, pp. 382–398.
24. Nielsen, J.B.: Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In: CRYPTO 2002. LNCS, vol. 2442, pp. 111–126.
25. Schnorr, C.P.: Efficient signature generation by smart cards. Journal of Cryptology 4(3), 161–174 (1991)
26. Yoneyama, K., Miyagawa, S., Ohta, K.: Leaky random oracle (extended abstract). In: ProvSec 2008. LNCS, vol. 5324, pp. 226–240.