Crypto School
— extra material —

Universität Bonn
Bonn-Aachen International Center for Information Technology

Version: December 15, 2015

Chapter 21

Random oracle reductions

21.1. Random oracles

Security reductions in the standard model are hard to find. Researchers have tried to judiciously weaken assumptions so that on the one hand reductions can be exhibited and on the other hand the problems do not become trivial.

An example of such a relaxation is the *random oracle model*. It applies to situations where hash functions occur, say in signature schemes. Instead of having to use the values of a specified (probabilistic) hash function, the attacker is allowed to fabricate hash values himself. If he was given a completely free hand in doing so, then his task would be trivial in many cases. The restriction now is that the values he fabricates must be uniformly random. We will see in the next section how he can publish random values about which he knows a little secret that he can use to his advantage. Imagine a painter producing random colors whose composition from a few standard paints he notes; he will be able to recreate the color but others will have a hard time doing so.

DEFINITION 21.1. *We have a message space M and a finite set H (of "hash values"). A* random oracle *for (M, H) is a machine O interacting with a request machine $\mathcal{R}$ in rounds. In each round, $\mathcal{R}$ sends a request message $m \in M$ to O, and O returns a value in H, designated as $h(m)$. There are three requirements:*

(i) The length of each message in M and the running time of O are polynomial in the length of the elements of H.

(ii) If the same message is sent in different rounds, then the same value is returned each time.

(iii) The values returned are independent and uniformly random in H.

It is easy to implement such an oracle.

ALGORITHM 21.2. Random oracle O.

Given: M, H and $\mathcal{R}$ as above.

1. O keeps a list L of messages and values $(m, h(m))$ stored in its two fields. Initially, this list is empty.
2. On receiving a request message $m \in M$, O checks whether it occurs somewhere in the first field of L. If so, O returns the corresponding second field.
3. Otherwise, O produces a uniformly random element $h(m)$ in H, independent of previous choices, returns $h(m)$, and appends $(m, h(m))$ to L.

It is clear that Algorithm 21.2 implements a random oracle as defined. For an example, see the next section. The list may contain further fields besides m and $h(m)$, which help the algorithm using O in solving its task.

Several variations on this theme are possible. We might drop the restriction of polynomial time, thus allowing more than polynomially many requests. Also, O might outsource the task of checking for duplicates to $\mathcal{R}$, and then O would just be a random generator for H. Finally, we might allow other distributions on H, besides the uniform one.

If M is finite and we ignore the time restriction, $\mathcal{R}$ might just feed all values $m \in M$ to O and build the graph $\{(m, h(m)) \colon m \in M\}$ of a function $h \colon M \to H$, obeying the requirement of Definition 21.1 (iii). This ensures that h is a random function and thus a perfect example of a hash function. However, the space (and time) required would be exponential. In fact, one of the salient feature of the RSA permutation $x \mapsto x^d$ is that it looks almost random (except for the multiplicative property (3.4) in Crypto School) but requires only linearly many bits for storage.

Suppose that we have a signature or encryption scheme S with a reduction from a conjecturally hard problem X to breaking S, in the random oracle model. In order to actually implement S, we have to substitute the oracle calls by the evaluation of a hash function that is publicly known to all participants. One might use a hash function like SHA-256 (Section 7.6 in Crypto School) or a discrete-log based one (Section 7.2 in Crypto School). However, this fails miserably.

What is the relation between standard and random oracle reductions? In the former, such as Algorithm 9.8 in Crypto School, the attacker uses the publicly known hash function h that is part of the system. In the latter, he can fabricate his own h^*, subject to the constraints of Definition 21.1. We assume that the given h has the random property of Definition 21.1 (iii). Then the attacker might as well use h for h^*, and thus any standard reduction is also one in the random oracle model.

A standard reduction "$X \leq_p$ breaking S" implies that an efficient algorithm for breaking S yields an efficient algorithm to solve X. This important property

does not hold for random oracle reductions. The property is "true" if we replace the original h in S by h^* from the reduction. For this to be allowed, h^* has to be a fixed and publicly known function. This is not the case, for example, in reduction of Theorem 21.5 below.

Canetti, Goldreich & Halevi (2004) have put forth a rather generous notion of "implementing S" and shown that for any S, secure in the random oracle model, there exists a modification S' which is also secure in this model, and also demonstrably insecure. Thus there is no implication from security in the random oracle model to actual security. Of course, the reverse implication holds. Random oracle security is a necessary but not sufficient condition for actual security. It is like showing that a certain class of attacks do not work against S, which does not preclude other attacks from being successful. In fact, it is just that, with the class consisting of those attacks that work in the random oracle model. (Always assuming that the base problem X is actually hard.)

The assessment of Cramer & Shoup (1999) is that "a random oracle reduction is at best a heuristic device that gives strong evidence that a scheme cannot be broken—however, it is entirely possible that a scheme can be secure in the random oracle model, and yet be broken without violating any particular intractability assumption." Thus the random oracle assumption is problematic. However, as long as we are aware of its potential weakness and yet have no stronger tool available, we may still use the assumption and hope for better times.

It is better to have a random oracle reduction than none at all, provided one understands the limitations of this approach. The constructions in Canetti *et al.* (2004) are somewhat "unnatural", and none of the usual protocols shown secure in the random oracle model have actually been broken. One of the main advantages of this model is that we can do at least something for protocols that seem out of reach for standard security reductions as presented in Section 9.5 in Crypto School.

21.2. RSA signatures with full domain hashing

In this section, we provide a reduction from breaking RSA to forging a signature scheme involving RSA that avoids the ultra-short hash values of Section 9.6 in Crypto School. Alas, no such reduction is known without further assumptions. While the currently popular hash functions produce fairly short hash values, say of 256 or fewer bits, we have to assume hash values of the same length n as the RSA key. RSA is considered secure at $n = 2048$ or, better, at $n = 4096$ bits. That is, we now have ultra-long hash values. This is much less of worry than ultra-short ones, since we can extend short hash values to n bits by nontrivial padding, and we certainly have to assume that this is still secure; see the preceding section. A more restrictive assumption is that the hash function is

given by a random oracle under control of the attacker.

SIGNATURE SCHEME 21.3. Full Domain Hash RSA.

Key Generation.

Input: A security parameter n.
Ouput: An n-bit RSA modulus $N = pq$ with public key $K = (N, e)$ and secret key $S = (N, d)$ as in the RSA notation?, and a surjective hash function $h\colon \mathbb{B}^* \to \mathbb{Z}_N^\times$.

Signing.

Input: Message $m \in \mathbb{B}^*$.
Output: $\mathrm{sig}_S(m) \in \mathbb{Z}_N$.

1. $\mathrm{sig}_{(S)}(m) = h(m)^d$ in $\mathbb{Z}_N^\times$.

Verifying.

Input: $m \in \mathbb{B}^*$ and $u \in \mathbb{Z}_N^\times$.
Output: "true" or "false".

2. $\mathrm{ver}_K(m, u) =$ "$u^e = h(m)$ in $\mathbb{Z}_N^\times$".

We first check, as usual, soundness and efficiency. For the soundness, let $s = \mathrm{sig}_S(m)$. The verification step returns "true" if $u = s$, since then $u^e = t$. Furthermore, the power map $\pi\colon \mathbb{Z}_N^\times \longrightarrow \mathbb{Z}_N^\times$ with $\pi(a) = a^e$ is bijective, since $gcd(e, \varphi(N)) = 1$; see Corollary 15.61 in Crypto School. Therefore $u^e = s^e$ implies that $u = s$, and if $u \neq s$, the verification step returns "false". The efficiency is clear, since h is easy to evaluate and otherwise only exponentiations in G occur.

The reader may object that the assumptions on h are rather ludicrous. How could we possibly avoid the multiples $\mathbb{Z}_N \setminus \mathbb{Z}_N^\times$ of p or q? How to be surjective onto $\mathbb{Z}_N^\times$? We might have a randomized hash function $h_0\colon \mathbb{B}^* \to \mathbb{B}^n$, and for each n-bit modulus N take $h(m)$ as the remainder of $h_0(m)$ in $\mathbb{Z}_N$. If $h(m) \notin \mathbb{Z}_N^\times$, we hash again. However, we now work under the assumption as stated.

The security of FDH-RSA is supported by the following random oracle reduction; see Section 21.1 for various caveats. We start with an overview of the reduction and later discuss how to implement the various steps.

ALGORITHM 21.4. Random oracle reduction $\mathcal{A}$ from breaking RSA to existential forgery $\mathcal{F}$ with chosen messages of FDH-RSA.

Input: N, e, y as in the RSA notation ?.

Output: Either x with $x^e = y$ in $\mathbb{Z}_N^\times$ or "failure".

1. $\mathcal{A}$ sends (N, e) to $\mathcal{F}$.
2. $\mathcal{A}$ receives hash queries from $\mathcal{F}$ and delivers hash values to $\mathcal{F}$.
3. $\mathcal{A}$ receives signature queries from $\mathcal{F}$ and either delivers a valid signature to $\mathcal{F}$ or returns "failure".
4. $\mathcal{F}$ sends a "forged" signature (m^*, s^*) of FDH-RSA to $\mathcal{A}$. Its hash value $h(m^*)$ is among those requested in step 2.
5. $\mathcal{A}$ verifies (m^*, s^*) and returns "failure" if the verification answer is "false".
6. $\mathcal{A}$ returns either x with $x^e = y$ in $\mathbb{Z}_N^\times$ or "failure".

The various queries in step 2 and 3 may be interleaved. By assumption, the verification in step 5 succeeds with probability at least $\sigma_{\mathcal{A}}$.

THEOREM 21.5. *Suppose we have an existential forger $\mathcal{F}$ with chosen messages for FDH-RSA, using time $\tau_{\mathcal{F}}, q_{\text{sig}}$ signature queries and q_{hash} hash queries and having success probability $\sigma_{\mathcal{F}}$. Then the above reduction $\mathcal{A}$ can be implemented for n-bit RSA moduli in the random oracle model so that*

$$\sigma_{\mathcal{A}} \geq \frac{\sigma_{\mathcal{F}}}{4q_{\text{sig}}},$$
$$\tau_{\mathcal{A}} \leq \tau_{\mathcal{F}} + (q_{\text{sig}} + q_{\text{hash}} + 1) \cdot \text{poly}(n).$$

PROOF. We have to explain how steps Algorithm 21.4 step 2, 3, 5 and 6 are implemented. In step 2 of Algorithm 21.4, $\mathcal{A}$ implements the random oracle in the following way, which gives it the decisive advantage.

The idea is that $\mathcal{A}$ uses two strategies to deliver hash values to $\mathcal{F}$: in the first strategy (corresponding to $b = 1$, as below), the hash value is chosen so that $\mathcal{A}$ can easily deliver a valid signature to $\mathcal{F}$, if that is requested later. In the second strategy (corresponding to $b = 0$), the hash value is chosen so that when $\mathcal{F}$ returns, at the end, a valid signature of that particular message, then $\mathcal{A}$ can easily compute x with $x^e = y$ in $\mathbb{Z}_N^\times$. More precisely, $\mathcal{A}$ has an internal parameter δ with $0 < \delta < 1$ and starts with the empty list L. On a hash query for a message m in step 2, it first checks whether m occurs as a first field somewhere in L. If so, it delivers the corresponding hash value h_0. Otherwise, it chooses $r \xleftarrow{\$} \mathbb{Z}_N^\times$ and $b \xleftarrow{\$} \{0, 1\}$ with $\text{prob}\{b = 1\} = \delta$. If $b = 1$, then $h_0 = r^e$ in $\mathbb{Z}_N^\times$ and $(m, h_0, 1, r)$ is added to L. If $b = 0$, then $h_0 = yr^e$ in $\mathbb{Z}_N^\times$ and $(m, h_0, 0, r)$ is added to L. $\mathcal{A}$ sends h_0 as $h(m)$ to $\mathcal{F}$.

If $\mathcal{F}$ requests in step 3 the signature of some message m, $\mathcal{A}$ first determines $h(m)$ as in step 2 above. If in the corresponding entry (m, h_0, b, r), $\mathcal{A}$ finds $b = 1$, then it delivers r to $\mathcal{F}$. If $b = 0$, $\mathcal{A}$ returns "failure".

In step 5, when $\mathcal{A}$ receives a signed message (m^*, s^*) from $\mathcal{F}$, there is, by assumption, an entry $(m^*, h(m^*), b^*, r^*)$ in L, and this is the only entry with m^*

in the first field. If $(s^*)^e = h(m^*)$ in $\mathbb{Z}_N^\times$, then $\mathcal{A}$ goes to step 6, otherwise $\mathcal{A}$ returns "failure". In step 6, $\mathcal{A}$ returns $x = s^*/r^*$ if $b^* = 0$ and reports "failure" if $b^* = 1$.

We have to prove three claims:

- $\mathcal{A}$ runs in polynomial time,
- $\mathcal{A}$ satisfies all of $\mathcal{F}$'s requests properly,
- $\mathcal{A}$ returns a correct solution to the RSA problem with probability at least some specified value $\sigma_{\mathcal{A}}$, namely at least $\sigma_{\mathcal{F}}/4q_{\text{sig}}$.

The first item is easy, since $\mathcal{A}$'s computations are just random choices and exponentations in $\mathbb{Z}_N^\times$, using $O(n(q_{\text{hash}} + q_{\text{sig}} + 1))$ operations in $\mathbb{Z}_N^\times$, where n is the bit length of e. As to the proper satisfaction of $\mathcal{F}$'s request, the values r^e and yr^e are uniformly random elements of $\mathbb{Z}_N^\times$ (by Corollary 15.62 in Crypto School, since $\gcd(e, \varphi(N)) = 1$), and so is h_0. Thus the hash values delivered in step 2 are as required in the random oracle model. In step 3, a signature r delivered by $\mathcal{A}$ is valid, since $r^e = h_0 = h(m)$ in $\mathbb{Z}_N^\times$, as required in the verification step of FDH-RSA. "Failure" occurs here at each signature request with probability $1 - \delta$, and success with probability δ. In step 5, $\mathcal{A}$ knows the entry $(m^*, h(m^*), b^*, r^*)$ in L from step 2. By the assumption on $\mathcal{F}$, $(s^*)^e = h(m^*)$ in $\mathbb{Z}_N^\times$ and hence $\text{ver}_{N,e}(m^*, s^*) =$ "true" with probability at least $\sigma_{\mathcal{F}}$. In step 6, if $b^* = 0$ then

$$x^e = (\frac{s^*}{r^*})^e = \frac{h(m^*)}{(r^*)^e} = \frac{y(r^*)^e}{(r^*)^e} = y \text{ in } \mathbb{Z}_N^\times,$$

so that x indeed solves the RSA problem. This happens with probability $1 - \delta$. If a total of q_{sig} signatures are requested by $\mathcal{F}$, then the success probability in the various passes through step 3 is $\delta^{q_{\text{sig}}}$, and overall we have

$$\sigma_{\mathcal{A}} \geq \delta^{q_{\text{sig}}} \cdot (1 - \delta)\sigma_{\mathcal{F}}.$$

We have to consider the function $u(\delta) = \delta^q(1 - \delta)$ for positive δ and a positive integer q. Its derivative with respect to δ is $\delta^{q-1}(q - (q + 1)\delta)$, and u takes its maximal value at

$$\delta_0 = \frac{q}{q+1} = 1 - \frac{1}{q+1}.$$

Furthermore,

$$u(\delta_0) = \frac{1}{q}(1 - \frac{1}{q+1})^{q+1} \geq \frac{1}{4q};$$

in fact, $(1 - 1/m)^m$ tends to e^{-1} for large m, and is at least $1/4$ for $m \geq 2$. It follows that, by choosing $\delta = \delta_0$ in $\mathcal{A}$, we have

$$\sigma_{\mathcal{A}} \geq \frac{\sigma_{\mathcal{F}}}{4q_{\text{sig}}}.$$

□

Notes 21.1. In Definition 21.1 (iii), it is sufficient that the distribution of the values be indistinguishable in polynomial time from one that satisfies the requirements; we neither elaborate nor use this in the following.

21.2. The FDH-RSA signature scheme and its security reduction were presented by Bellare & Rogaway (1996).

The proof of Theorem 21.5 is due to Coron (2000).

Bibliography

The numbers in brackets at the end of a reference are the pages on which it is cited. Names of authors and titles are usually given in the same form as on the article or book.

Mihir Bellare & Phillip Rogaway (1996). The Exact Security of Digital Signatures — How to Sign with RSA and Rabin. In *Advances in Cryptology: Proceedings of EUROCRYPT 1996*, Saragossa, U. Maurer, editor, volume 1070 of *Lecture Notes in Computer Science*, 399–416. Springer-Verlag, Berlin. ISBN 354061186X. ISSN 0302-9743. URL http://dx.doi.org/10.1007/3-540-68339-9_34. [9]

Ran Canetti, Oded Goldreich & Shai Halevi (2004). The Random Oracle Methodology, Revisited. *Journal of the ACM (JACM)* **51**(4), 557–594. URL http://portal.acm.org/citation.cfm?coll=GUIDE&dl=GUIDE&id=1008734. Article home page: http://www.wisdom.weizmann.ac.il/~oded/p_rom.html. [5]

Jean-Sébastien Coron (2000). On the Exact Security of Full Domain Hash. In *Advances in Cryptology: Proceedings of CRYPTO 2000*, Santa Barbara, CA, Mihir Bellare, editor, volume 1880 of *Lecture Notes in Computer Science*, 229–235. Springer-Verlag, Berlin. ISBN 3-540-67907-3. ISSN 0302-9743 (Print) 1611-3349 (Online). URL http://www.springerlink.com/content/v5pn8pp8nmfk675w/?p=b5cab6950f154a24ba652e70b22eafda&pi=0. [9]

Ronald Cramer & Victor Shoup (1999). Signature Schemes Based on the Strong RSA Assumption. In *ACM Conference on Computer and Communications Security*, 46–51. URL http://citeseer.nj.nec.com/article/cramer98signature.html. [5]