On Instantiation of the Random Oracle

He Ge*

Abstract. In the current methodology of the Random Oracle Model, the random oracle is instantiated by a "good cryptographic hash function". However, due to the work of Canetti, Goldreich and Halevi, such methodology has been found problematic because there exist a construction secure in the Random Oracle Model, but any instantiation of the random oracle by any fully specified function which includes also any "good cryptographic hash function" will result in an insecure implementation. We investigate the Canetti-Goldreich-Halevi method, and propose a new method for the instantiation of the random oracle, in which the random oracle is instantiated by a floating pseudorandom function. Under this new method, Canetti, Goldreich and Halevi's construction will have a secure implementation. Our work puts the methodology of the Random Oracle Model on firm grounds.

Keywords: The Random Oracle Model, Instantiation, Floating Pseudo-Random Function, h-Box

1 Introduction

A common practice in the design of cryptographic protocols is using a hash function to "randomize" some parameters in a cryptographic scheme. For example, in the full domain hash RSA signature scheme, a message to be signed first gets hashed into the domain of Z_n^*, then the hash value is applied on the RSA function, obtaining a signature like

$$\sigma = h(m)^d \bmod n.$$

In 1996 Bellare and Rogaway formulated this type of practice as the Random Oracle Model [4]. A random oracle, commonly referred to as an abstraction of a hash function, is a black box which can output a random string for a query, and always produces the same result for the same query. Under such "idealization", a proof is carried out to demonstrate the security of a protocol. Due to simpleness and elegance of this model in the design of secure protocols, many new schemes have been proposed with the Random Oracle Model based security proofs (e.g. [5, 6]). However Canetti, Goldreich and Halevi have constructed a scheme which can be proved secure in such model while any real implementation will result in an insecure construction [7]. Their work shows the Random Oracle Model fundamentally has some flaws.

After their work, other research results have been published for the validity of the random oracle methodology [10, 8, 2]. Gradually, this methodology becomes less desirable as a technique in the design of cryptographic schemes in cryptography. However, it seems the exploit on the Random Oracle Model has not shaken the real world as one might expect. Some widely used cryptographic schemes in our daily life are still those which can only be demonstrated secure in the Random Oracle Model (e.g. the RSA PKCS #1 [9]).

In this work, we study the attacking technique on the random oracle methodology developed by Canetti, Goldreich and Halevi, which will be referred as the CGH technique in the rest of the paper. We propose a modified random oracle model and a new method for the instantiation of the random oracle. Our solution fixes the problem exposed by CGH technique. In general, our new model puts the random oracle methodology on firm grounds.

* Email: sea.gehe@gmail.com.

2 Canetti-Goldreich-Halevi's Attack on the Random Oracle Model

The random oracle methodology works like this way. A public random oracle is set up being accessed by all participants in a cryptographic scheme. The random oracle answers queries. When a query comes, the random oracle first checks its internal answer list if the query has been answered before. If there is no entry for this query, the random oracle produces a random value as the answer, records this $(query, answer)$ on its answer list, and returns the answer to who issues the query. If there is an entry on the list for this query, it returns the same answer. From outside, the random oracle is a black box. In such setting, one then can prove the security of a cryptographic scheme with respect to its security definition. We call the setting in which the random oracle is deployed as a ideal one.

When implementing the scheme, the random oracle is replaced, or instantiated by a well defined cryptographic hash function. This methodology seems working for a lot of schemes which are widely deployed in practice. However, there is no exact clarification or definition what a good implementation of the random oracle would be, or what a good cryptographic hash function is in the context of the Random Oracle Model.

If we compare the setting of a random oracle based security proof, and its counterpart in real implementation, it is not difficult for us to notice an obvious discrepancy between these two settings. In the proof, the random oracle is a black box which outputs random values. However, any real implementation of the random oracle is a fully specified function so its behavior is determined, which can be understood by all participants. The discrepancy potentially could be explored in any random oracle based schemes.

The attacking method by Canetti, Goldreich and Halevi is exactly the exploration of such discrepancy. They devised a scheme whose security will rely on black box properties of the random oracle. More specifically, they defined an evasive relation on a pre-image and its image for the random oracle. Such relation can not be satisfied in the ideal setting. However, it is easy to find an instance for such relation when the Random Oracle is implemented by a function ensemble. In the current methodology of the Random Oracle Model, a good cryptographic hash function is essentially being picked up by the method of a function ensemble. In their contrived schemes, there is an operation that when a pair of pre-image and image of the hash function satisfying the evasive relation, the schemes do an insecure operation which breaks the scheme, such as outputting the private key of the scheme. The probability of finding an instance to satisfy the evasive relation is negligible in the ideal setting. However, any fully specified functions will make this relation trivially being satisfied, so the scheme is broken.

Canetti, Goldreich and Halevi's work exposes the fundamental problem in the Random Oracle Model. That is, any implementation of the random oracle by a fully specified function will certainly make the scheme vulnerable. This was summarized in their paper as "having a description of a function is much powerful than just having a black box access to that function".

3 A New Way for Instantiation of the Random Oracle

There is a natural way to implement the random oracle without any security concerns. That is, the random oracle can be instantiated by an online party who implements a pseudorandom function, and keeps secret the seed of the function. This was even mentioned by Bellare and Rogaway when they formulated the concept of the random oracle [4]. Canetti, Goldreich and Halevi further iterated

the same idea in their work for attacking the random oracle methodology. The problem for this method is that a trusted third party needs to be online all the time, which is not a practical solution. So heuristically a "good cryptographic hash function" is used to replace the online party and a hash query is evaluated locally with the hope that this operation will not impair the scheme. As the analysis in the previous section, this operation exhibits a huge gap between the ideal system and the real one. However, we shows that there exits ways to remove the online third party while keeping most of its security property.

3.1 Floating Pseudo-Random Function

The key for the success of the CGH technique depends on the knowledge of description of a function ensemble. In a real system, the random oracle is implemented by a fully specified function in the setup stage for a scheme, and fixed ever after. To prevent attacks due to the CGH technique, a natural consideration is that if we could only partially specify a function, and the full specification is only fixed after the query appears.

If we look at how the random oracle works in any security proofs, it should not that difficult to realize that the random oracle in fact works in a "partially" specified manner. The answer list of the random oracle is empty initially. When a fresh query comes, the random oracle then produces a random answer, and adds the $(query, answer)$ pair on the answer list. At the time when there is no query, the behavior of the random oracle is undefined, or, "floating". It can associate any query to any random value on its domain.

For the instantiation of the random oracle through the online party, certainly the online party can works in a floating manner. Instead of having a fixed secret seed, it uses a different seed for a different query, and only produces a seed when a query appears. In the eyes of the outsider, there is no difference since all these seeds are secret. We call the pseudo-random function working in this manner as floating pseudo-random function, which is referred as f-PRF in the sequel.

We can think f-PRF as multiple instances of a pseudo-random function, each of which is associated a different seed. Each instance of the pseudo-random function is independent to each other. Internally, f-PRF mimics the random oracle much better than a pseudo-random function with a fixed seed.

3.2 Instant Black Box

For the online party who implements f-PRF, each seed is produced randomly and never repeats itself. Once a pair of $(query, answer)$ is obtained, it does not matter anymore whether the seed should continue to be kept secret or not since anyway all transactions are done. The publication of a seed certainly will not have any effect on the security of a scheme. Based on this observation, we let the online party publish its seed along with associated $(query, answer)$ pair.

This observation makes us re-consider what the black box means in the view of the adversary. Since the random oracle is a black box, any meaningful attacks on the random oracle can only be conducted through exploration the relationship between a pre-image and its image. Therefore, if a structure reveals its secret only after a query is fixed, it is still a black box in term of what the adversary can do. The online party deploying this strategy is certainly still a black box for any outsiders.

So far, our analysis still stays at the stage where the random oracle is implemented as an online party. We push our consideration further. One important observation of publishing a seed after the

transaction is finished is that the secret can only be known after the query is fixed. Thinking about this in another way, if we can ensure a seed for f-PRF can only be obtained after a query for f-PRF is fixed, then it does not matter f-PRF has to be deployed at an online party or not. What we are looking for is a mechanism in which a seed can only be produced and known by the adversary when the query is fixed. That is, a query goes first, a seed comes next. This implies at the very moment when the adversary comes up with a query, the seed for this query has yet to be determined, so the behavior of f-PRF is not known by the adversary beforehand. In the views of the adversary at this moment, f-PRF IS a back box. We call such a mechanism an instant black box.

This mechanism is achievable. In the simplest form, we could think a cryptographic hash function can achieve such goal. For conceptual purpose, we propose a prototype structure called h-Box which is showed in Figure 1. There are two inputs for h-Box, q-input for query and r-input for a random input. h-Box outputs a value s which is used as a seed for f-PRF. Once q, r is fixed, s is subsequently fixed, and so the behavior of f-PRF. In h-Box, a q-input first goes through a cryptographic hash function, and the result is further input into a oneway permutation function. Meantime, a r-input goes through a oneway permutation function. Two results from the oneway permutation functions are xor-ed to produce s. h-Box should ensure oneway permutation so it is infeasible to compute q, r from s. We require h-Box outputs s uniformly on its domain. It should be pointed out h-Box is a fully specified function, and not considered as the random oracle.

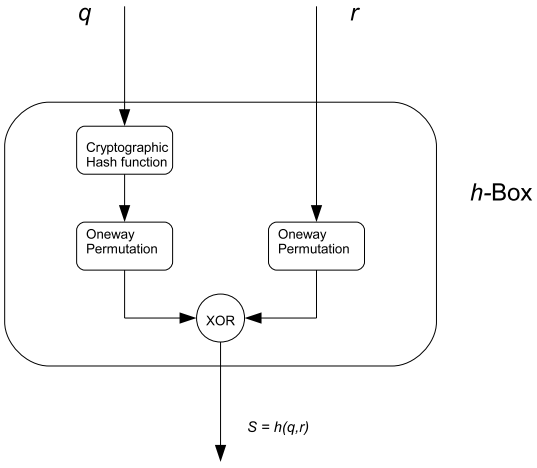

Fig. 1. The Implementation of h-Box

With h-box, the sequence of a query going ahead of a seed is enforced, which effectively carries out the mechanism of an instant black box. As a result, the online party is not really necessary any more. All participants in a scheme can evaluate f-PRF with the help of h-Box on their own. We will discuss h-Box more in the next section.

So far, our analysis presents a gradual transition for the instantiation of the random oracle, which is showed as in Figure 2.

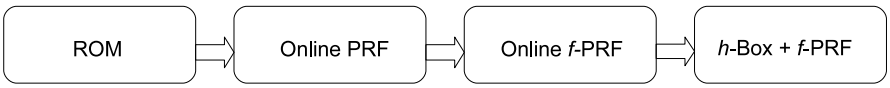

Fig. 2. The Transition of The Random Oracle

3.3 Isolation of the Random Oracle

There exists a pitfall which could be easily neglected when instantiating the random oracle. So far our discussion is in an abstract manner. That is, we discuss the random oracle without even considering a concrete setting. However, the random oracle is not a stand alone structure, and it resides in a specific scheme. One implication of the random oracle as a black box in the ideal system is that the random oracle is completely separated from the rest of the protocol/scheme. This further means there will be no exploration of relationship between the random oracle and the rest of the scheme, since no internal structure exists in the random oracle could be utilized to attack the scheme.

However, an instantiation of the random oracle could be correlated to the rest of the system in complicated way, and the correlation may expose vulnerable area for the designated scheme. The exploration of the random oracle methodology through this correlation attack is not a generic one. Its effectiveness depends on a specific implementation in which certain structural connection is introduced between a specific instantiation of the random oracle and the rest of the system, and such connection is explorable by a polynomial-time attacker. On the contrary, the CGH technique is considered as a generic attacking strategy since it only depends the description of the function which instantiates the random oracle.

In principle it is known on how to prevent such pitfall in practice. Any instantiation of the random oracle should be completely independent to the rest of the system. For instance, cryptographic primitives deployed in an instantiation should not have any inherent connections to those used by the rest of the system by using different mathematical structures, unrelated complexity assumptions, etc. We may call this principle as the isolation principle of the random oracle.

4 A New Random Oracle Model

Based on the analysis in the previous section, we propose a new version of the Random Oracle Model. The new model is showed as in Fig 3.

The key structure in the figure is h-Box. A scheme based on the new model should have such structure both in the ideal system and the real system. In the new model,

- The random oracle accepts two inputs: one input is for query, another is for seed. A query is the concatenation of q-input and r-input of h-Box.
- The seed must be produced by h-Box. In another word, no one can directly feed a seed into the random oracle.
- The random oracle is instantiated by a f-PRF.

In the ideal system of the new model, the output of h-box, i.e., s, is used by the random oracle to choose a random function, and an answer is further obtained when the query is applied on this function. In the real system, s binds f-PRF so a fully specified function is obtained. If the distributions

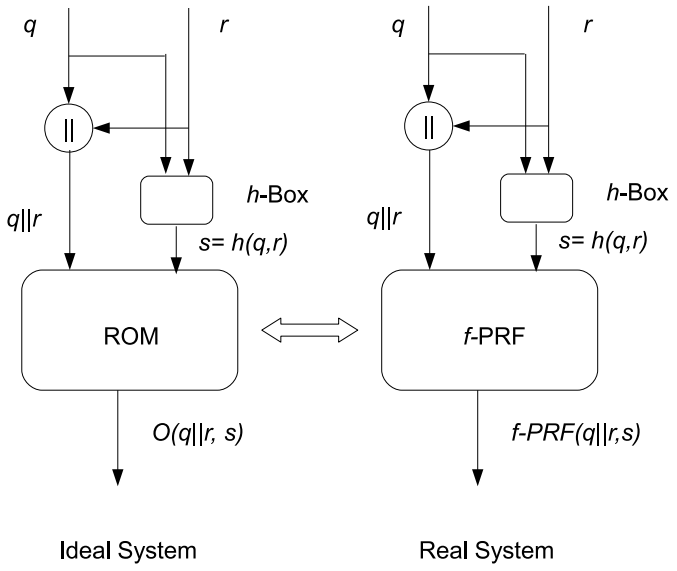

Fig. 3. The New Random Oracle Model

induced by outputs of the random oracle and h-PRF are statistically the same, we say f-PRF is a perfect instantiation of the random oracle. In such case, h-Box works as a pseudo-random function, and we say that it is equivalent for using s to pick up a random function in the ideal system and using s to bind f-PRF in the real system.

There exists discrepancy between the random oracle and f-PRF in the new model. We can think the random oracle internally has a pool of functions, which all map any query on its domain uniformly. No matter what an adversary can do through h-Box, the random oracle will always choose a function from its function pool, so the output of the random oracle is always uniformly distributed on its domain. In the real setting, s is directly bound to f-PRF. For a specific implementation, there could exist a set of s, and the output distribution induced by such set will exhibit statistical difference, which might be detected by a polynomial-time attacker with non-negligible probability. The r-input of h-Box controls the quality of s. If r is truly random, s is randomly distributed in the seed range of f-PRF. So f-PRF will be a perfect instantiation of the random oracle. r-input in a specific scheme can be controlled by the adversary, so the instantiation will not be a perfect one. In such case, the discrepancy indeed could happen. However, such discrepancy is less serious than the one between the black box and the fully specified function in the current random oracle model. We call such instantiation a weak one. In the next section, we will show the full domain RSA signature is a perfect instantiation, while the OAEP-like RSA scheme is a weak one.

To exploit weak instantiation in the new model, any attacking strategy would depend on

- whether there exists such a set of s in an implementation of f-PRF;
- or if it exists, whether it can be mapped on through h-Box by a polynomial-time attacker with non-negligible probability.

Any implementation of h-Box and f-PRF should ensures that these two conditions are hard to be satisfied, which are achievable based on some regular complexity assumptions. In practice,

a regular cryptographic hash function is supposed to map any pre-image uniformly on its domain among other properties such as collision-resistance, onewayness, etc.

Let's briefly explain the CGH techniques does not work any more in the new model. The CGH technique would define an evasive relation between a pre-image $(q||r, s)$ and its output with respect to the random oracle or f-PRF. The difficulty of finding an instance to satisfy such relation is based on the knowledge of s. In the current random oracle methodology, an instantiation of the random oracle happens at the system setup stage where a random function is picked up from a function ensemble, and fixed ever after. In the new model, this pickup operation dynamically happens when a query appears. The output s of h-Box is used to choose a function in the ideal system, or bind f-PRF in the real system. The oneway property of h-Box ensures s obtained after q, r. Both in the ideal system and the real system, it is infeasible to find a candidate for the evasive relation. Therefore the new model prevents the attack due to the CGH technique.

The last note for the new model is the reminder of the isolation principle of the random oracle model. The implementation of h-box and f-PRF should be independent to each other and the rest of the scheme.

5 The RSA Scheme in the New Random Oracle Model

In this section, we illustrate the idea of the new random oracle model through the RSA signature and encryption schemes. Our primary goal is for demonstration how a scheme based on the new model looks like. So we do not specify every possible details in the schemes, such as which specific f-PRF is needed, how to implement h-Box. These two schemes can be easily proved secure based on the RSA assumption when the pseudo-random function in the schemes is idealized as the random oracle. The proofs are essentially the same as those in the current random oracle model.

In the following, Let $n = pq$, where p, q are two random large prime numbers which makes the factorization of n infeasible. Let d is a random element in Z_n^* and relatively prime to $\phi(n)$, where $\phi(n) = (p-1)(q-1)$. Let $e = d^{-1} \mod \phi(n)$. Let h be h-box, $\mathcal{H}$ be a pseudo-random function. In practice, we can consider choosing a good cryptographic hash function as a h-Box, and *HMAC* algorithm as f-PRF [3, 1].

5.1 The New Full Domain RSA Signature Scheme

In the full domain RSA signature scheme, the pseudo-random function $\mathcal{H}$ produce a random element in $(0, n-1)$. To produce a signature on m, the signer picks a random r, and computes a signature as

$$\sigma = (\mathcal{H}(m||r, h(m, r)))^d \mod n.$$

The signature is (σ, r). A verify can do the verification as

$$s = \sigma^e \mod n, \quad s \stackrel{?}{=} \mathcal{H}(m||r, h(m, r)).$$

Here we give a proof in the random oracle model. The interesting part for this proof is that the reduction is tight from attacking the RSA problem to forging a signature. In comparison, the proof based the current random oracle model is not a tight one.

Theorem 1. *The new full domain RSA signature scheme is a perfect instantiation of the random oracle based RSA signature scheme, which is secure under the adaptive chosen message attack based on the RSA assumption.*

Proof. In the ideal setting of the RSA signature scheme based on the new random oracle model, the random oracle is set up to answer queries. A signing simulator is input a random u in Z_n^*, asked for a v such that $v^e \equiv u \mod n$.

The simulator plays the adaptive chosen message attack with an adversary, who is assumed able to forge a valid signature after the game is finished. This attacking game is played polynomial times with respect to a security parameter of the scheme. The game is conducted as follows. The adversary can pick any m, r, compute $s = h(m, r)$, and send $(m||r, s)$ to the random oracle asking for the answer. The random oracle internally picks a large random prime integer k, compute $w = u^k \mod n$, and gives w to the adversary.

The adversary asks the signing simulator for the signature on m. The signing simulator picks a random r, obtains $s = h(m, r)$, asking the random oracle to produce an answer. The random oracle picks a random σ, compute $w = \sigma^e \mod n$, and let $w = O(m||r, s)$. Therefore the signing simulator is able to produce a valid signature as (σ, r) since

$$\sigma^e \mod n = w = O(m||r, h(m, r)).$$

Notice the random oracle internally uses different way to calculate w when the query issuer is different. Since the random oracle is a black box, the adversary will not be able to notice such difference. When the adversary receives (σ, r) from the signature simulator, it asks the random oracle to produce the answer for (m, r). The random oracle already produces the answer for this query, so it returns the same answer on its answer list. Notice r from the signing simulator is independent to ones from the adversary.

If eventually, the adversary is able to forge a signature on a message m such that

$$\sigma^e \equiv O(m||r, h(m, r)) \mod n.$$

Since $O(m||r, h(m, r)) \equiv u^k \mod n$, and k is a prime integer, we have

$$\sigma^e \equiv u^k \mod n.$$

By Shamir's trick [11], when e is relatively prime to k, this equation can be solved and we are able to have a result like

$$v^e \equiv u \mod n,$$

which is a solution to the RSA problem. However, this is assumed impossible under the RSA assumption. Therefore the RSA signature scheme in the ideal setting is secure under the adaptive chosen message attack. Also, the reduction is tight since the probability of forging a signature is equal to that of solving the RSA problem.

We notice in the proof that r-input of h-Box for signature generation is controlled by the signing simulator, which is beyond the control of any adversary. This means in the real instantiation, h-Box is controlled by the real signer, who should obviously picks r-input truly random. Therefore, the real system is a perfect instantiation. □

5.2 The New RSA Encryption Scheme

In the RSA encryption scheme, we need pre-process a message before applying it on the RSA function. Let l_m be the bit length of a message, l_r be the bit length of random r and $l_{\mathcal{H}}$ be the bit

length of output of f-PRF. $l_n = l_m + l_r + l_{\mathcal{H}}$. A cryptographic hash function h_2 produces a hash value in the range of $(0, 2^{l_m+l_r})$.

A message m in the valid range is pre-processed as follows. Let $r_{\mathcal{H}} = \mathcal{H}(m||r, h(m, r))$. Compute $r_2 = h_2(r_{\mathcal{H}})$, and obtain $\sigma = ((m||r) \oplus r_2)||r_{\mathcal{H}}$. To check whether σ is produced properly following the above steps, one can reverse these steps. Let $\sigma = r_2||r_{\mathcal{H}}$. Compute $r_2' = h_2(r_{\mathcal{H}})$. Let $m||r = r_2' \oplus r_2$, and check $r_{\mathcal{H}} \stackrel{?}{=} \mathcal{H}(m||r, h(m, r))$.

To encrypt a message m in the required range, one prepares a random r, and calculate σ as above. Then a cipher text is obtained as

$$c = \sigma^e \mod n.$$

To decrypt a cipher text c, one computes

$$\sigma' = c^d \mod n,$$

and check if σ' is formed properly. If so, m is extracted. Otherwise, the cipher text is rejected.

The proof for the RSA encryption scheme in the new model is similar to the one in [5], and we do not elaborate it here. One thing we need to point out, in the attacking game specified by IND-CCA2 (Indistinguishability under chosen adaptive chosen cipher text attack), h-Box is controlled by the adversary, certainly the adversary would not pick up r-input randomly. Therefore, the instantiation of the RSA encryption scheme in the new model by the adversary is a weak one. The security of the scheme then needs to assume that h-box produces s "uniformly", which is a quite regular assumption. For any participants in the RSA encryption scheme, an honest party surely would pick up r-input truly random to encrypt its message, in such case its instantiation is a perfect one.

6 Conclusion

In this paper we propose a new Random Oracle Model, which overcomes the fundamental gap exposed by the CGH technique. That is, a black box random oracle model can never be implemented by a fully specified function. In our new model, a floating pseudo-random function, f-PRF, is idealized as the random oracle, and an h-Box is deployed to produce seeds for f-PRF. The new model successfully fills the gap in the current Random Oracle model, putting the methodology of the Random Oracle Model on firm grounds.

There are two instantiation mode in the new model. In the case of perfect instantiation, a scheme keeps all security property guaranteed by the ideal system, while in the case of weak instantiation, we need the assumption that h-Box should satisfy the conditions enumerated in this paper. In any case, we have a clear structural level understanding of the new model, which implies that any potential problems in instantiation is fully aware. On the contrary, in the current random oracle model, instantiation is heuristic. This is biggest improvement on the current random oracle methodology.

As we have mentioned, some widely deployed schemes can only be demonstrated secure in the current random oracle methodology. With the introduction of the new model, how do we evaluate them in the frame work of the new model? Is their security really ensured as in their current implementation? We will discuss these topics in the future.

References

1. M. Bellare. New proofs for nmac and hmac: Security without collision-resistance. In *Advances in Cryptology — Crypto'06, LNCS 4117*, pages 602–619, 2006.

2. M. Bellare, A. Boldyreva, and A. Palacio. An uninstantiable random-oracle-model scheme for a hybrid-encryption problem. In *Advances in Cryptology — Eurocrypt'04, LNCS 3027*, pages 171–188, 2004.
3. M. Bellare, R. Canetti, and H. Krawczyk. Keying hash functions for message authentication. In *Advances in Cryptology — Crypto'96, LNCS 1109*.
4. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *First ACM Conference on Computer and Communication Security*, pages 62–73, 1993.
5. M. Bellare and P. Rogaway. Optimal asymmetric encryption. In *A. de Santis, editor, Advances in Cryptology — EUROCRYPT 94, LNCS 950*, pages 92–111. Springer-Verlag, 1995.
6. M. Bellare and P. Rogaway. The exact security of digital signatures — how to sign with RSA and Rabin. In *Advances in Cryptology — Eurocrypt'96*, pages 399–416, 1996.
7. R. Canetti, O. Goldreich, and S. Halevi. The random oracle model, revisited. In *30th Annual ACM Symposium on Theory of Computing*, pages 209–218, 1998.
8. S. Goldwasser and Y. T. Kalai. On the (in)security of Fiat-Shamir paradigm. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science — FOCS'03*, pages 102– 114, 2003.
9. J. Jonsson and B. Kaliski. Public-key cryptography standards (pkcs) 1: Rsa cryptography specification version 2.1, 2003. http://tools.ietf.org/html/rfc3447.
10. J. B. Nielsen. Separating random oracle proofs from complexity theoretic proof: The non-committing encryption case. In *Advances in Cryptology — Crypto'02*, pages 111–126, 2002.
11. A. Shamir. On the generation of cryptographically strong pseudorandom sequences. *ACM Transaction on computer systems*, 1:38–44, 1983.