T-79.511 Special Course on Cryptology / Zero Knowledge: Random Oracles and Fiat-Shamir Heuristic

Jan Hlinovsky
jan.hlinovsky@hut.fi

Draft version November 14, 2001

1 Introduction

To make actual use of cryptography we need protocols. Traditionally the way to design protocols has been just to create one and see if somebody can break it. If many clever people try to break it without success we can believe that the protocol might be "reasonably secure". Now it is clear that this is not good enough for theoretically minded people like theoretical computer scientists, and methods for providing *provable security* were developed.

Provable security is a method based on reduction. [1] The idea is to first formally model the property we want to prove, then analyze some particular scheme, based on some particular primitive (such as DES or RSA) and show that the only way to break the scheme is to break the underlying primitive. This way the security of the scheme can be reduced to the security of the primitive, a little bit like reducing some NP-complete problem to some other problem to show that the other problem is hard provided that the NP-complete problem is hard.

The problem with provably secure protocols is that they tend to be too inefficient compared to what we would like to use in practise [1]. Somewhere between provable security and pure heuristics lies the topic of this survey, the random oracle methodology. The random oracle methodology gives efficiency compared to provable security, and on the other hand some security compared to pure heuristics. There is a cost, however, and we will have a look at some criticism of the model and see where it fails.

2 Random Oracles

The name and explicite formulation of random oracle method was given by Bellare and Rogaway [2].

The method is stated as follows: Suppose we have a protocol problem Π and we want to design a good protocol P for Π. With an effective cryptographic primitive, some hash function h, we do the following [2]:

1. Find a formal definition for Π in the model of computation in which all parties (including the adversary) share a random oracle R.

2. Devise an efficient protocol P for Π in this random oracle model.

3. Prove that P satisfies the definition for Π.

4. Replace oracle access to R by computation of h.

This can be thought as a combination of provable security and heuristics, the first 3 steps being provable security with random oracle as the primitive, and then adding heuristic step to replace the ideal primitive with something practical and claiming that it still works. [2] says that the protocol problem Π and the protocol P must be "independent" from the hash function h. The authors of [2] then state that it is their "thesis that this method, when properly caried out, leads to secure and efficient protocols".

The idea of using a random oracle and then substituting it with an appropriate primitive appears first in [6, 7]. However, in their model the oracle is not available to the adversary, because the computation of the function replacing the oracle would requre knowing the seed and thus the function would not seem random at all.

Fiat and Shamir [5] were the first to use a emph-public random oracle, i.e. "the bad guys" also have access to the same oracle. In [5] the random oracle method is used to transform an identification scheme into a digital signature scheme. This special case of random oracle methodology is known as *Fiat-Shamir heuristic*.

3 Fiat-Shamir heuristic

In [5] Fiat and Shamir present a zero knowledge identification scheme. The protocol consists of three messages between the prover and verifier, $P \to V$, $V \to P$ and $P \to V$. The second message, from V to P, consists of a random challenge. This is the only message that the verifier needs to originate, but it is crucial to prevent the prover from cheating. Fiat and Shamir then go on to prove that the verifier's part in this protocol can be replaced by a truly random function f. Thus the scheme can be made non-interactive, although it is not zero knowledge anymore. (Everyone can verify a signature's validity but no one can forge them, so the verifier can not generate by himself the prover's messages with the same probability distribution [5].)

The Fiat-Shamir heuristic applies to any 3-round public-coin identification scheme (not necessarily zero knowledge) to construct signature scheme. Fiat and Shamir say that the function f_0 that is used in practice to replace truly random f must be "sufficiently strong". Practical implementations may be vulnerable but this only shows that f_0 is demonstrably non-random [5].

4 Magic functions

Magic functions, investigated by Dwork et.al [4] are defined as functions that fill the requirements to be used in Fiat-Shamir heuristic as replacements for the random function f.

Dwork et.al. prove an interesting connection between three problems from different areas, namely selective decommitment, 3-round weak zero knowledge arguments, and Fiat-Shamir heuristics. We will have a breaf look at the two later problems.

Let GAP denote the statement "there exists a 3-round public-coin argument for some $L \notin$ BPP that is zero knowledge according to the weakest definition in the hierarchy". The problem is to prove or disprove this statement.[4]

Let MAGIC be the statement that every 3-round public-coin argument for a language not in BPP has a magic function.

Dwork et. al. prove that

- GAP $\to$ $\neg$MAGIC
  If any language $L \notin$ BPP has a 3-round public-coin argument system (P, V) satisfying even the weakest type of ZK in their hierarchy, then the Fiat-Shamir methodology fails completely on this argument system.
- $\neg$GAP $\to$ MAGIC
  If GAP is false, i.e. if 3-round public-coin arguments satisfying even the lowest level of the hierarchy exist only for languages in BPP, then for every 3-round public-coin argument for a language not in BPP there is a magic function that at least weakly supports the Fiat-Shamir methodology.

We know that no language outside BPP has a 3-round black box ZK argument [8], but it is interesting that whether this is true for non-black box ZK arguments as well has an impact on the validity of Fiat-Shamir heuristic.

5 Defining zero knowledge in the random oracle

In [2] Bellare and Rogaway define zero knowledge arguments in the random oracle and show how to transform interactive zero knowledge arguments into non-interactive in the model. The following presentation follows closely their work.

5.1 Notation

As usual, zero knowledge argument means here computational zero knowledge argument of membership. The relation ρ defines the language L, and a string w is a witness for the membership of x in L when $\rho(x, w) = 1$. A *witness selector* W is a function that when given any input $x \in L$ returns a witness for the membership of x in L.

If A is a probabilistic algorithm, then for any inputs $x, y, \ldots$ the notation $A(x, y, \ldots)$ refers to the probability space which to the string σ assigns the probability that A, on input $x, y, \ldots$ outputs σ. If S is a probability space then $x \leftarrow S$ denotes the algorithm which assigns to x an element randomly selected according to S. For probability spaces $S, T, \ldots$ the notation $\Pr[x \leftarrow S; y \leftarrow T; \cdots : p(x, y, \ldots)]$ denotes the probability that the predicate $p(x, y, \ldots)$ is true after the ordered execution of algorithms $x \leftarrow S, y \leftarrow T$, etc. Let f be a function. The notation $\{x \leftarrow S; y \leftarrow T; \cdots : f(x, y, \cdots)\}$ denotes the probability space which to the string σ assigns the probability $\Pr[x \leftarrow S; y \leftarrow T; \cdots : \sigma = f(x, y, \ldots)]$.

A random oracle is seen as a map from $\{0,1\}^*$ to $\{0,1\}^\infty$ chosen by selecting each bit of $R(x)$ uniformly and independently, for every x. The set of all random oracles is denoted by 2^∞.

5.2 Definitions

A verifier is a polynomial time function V which is given the common input x, the so far conversation $c \in \{0,1\}^*$ and a private random tape $r \in \{0,1\}^\infty$ The function's return value $V(x, c, r)$ is either the next message $V \to P$ or a bit indicating whether the proof was accepted or not. A prover is a probabilistic polynomial time function P which gets the common input x, the conversaation c so far, and auxiliary input a (if $x \in L$ then a is a witness to this, else it is the empty string e), and returns the next message $P_a(x, c)$ to the verifier. Let $ACC(P_a, V, x)$ denote the probability that V accepts in an interaction with P_a on common input x.

Definition 5.1 We say that (P, V) is an interactive proof for L, in the random oracle model and with error probability $\epsilon(n)$, if $\epsilon(n) \leq 1/2$ and the following rwo conditions hold:

- *Completeness* If $x \in L$ then for all witnesses w to the membership of x in L it holds that $ACC(P_w, V, x) = 1$.
- *Soundness* For all probabilistic polynomial time provers P' and sufficiently long x it holds that $ACC(P'_e, V, x) \leq \epsilon(|x|)$.

Definition 5.2 For any oracle R let $\mathrm{conv}(V^R, P_a^R, x, r)$ be the space of all (transcripts of) conversations between P_a^R and V^R when the common input is x and V's random tape is $r \in \{0,1\}^\infty$.

Now we include the random oracle in the verifier's view and define

$\mathrm{rview}(V, P_a, x) = \{R \leftarrow 2^\infty; r \leftarrow \{0,1\}^\infty;$
$c \leftarrow \mathrm{conv}(V^R, P_a^R, x, r) : (c, r, R)\}$

Since the random oracle is part of the view, the simulator is allowed to construct a "simulation" of the oracle. This is similar to non-interactive zero knowledge and the "simulation" of a common random string, but here the random oracle is an infinite object and the simulator cannot be asked to output it. The simulator is allowed to prescribe a polynomial sized piece of the oracle and the rest is filled at random. A simulator is formally defined as a probabilistic, polynomial time algorithm which on any input x outputs a triple (c, r', T) where $T = (x_1, y_1), \ldots, (x_t, y_t)$ is a sequence of string pairs with the property that $x_1, \ldots, x_t$ are distinct. Now we define two new operations. The *random oracle completion* ROC takes T as input and returns an oracle R which is random subject to the constraint that $R(x_i)$ is prefixed by y_i for all $i = 1, \ldots, t$. Similarly the *random string completion* operation RSC takes a string $r' \in \{0,1\}^*$ and appendices an infinite sequence of random bits.

Now we can define the *completion* of $S(x)$ to be the probability space

$$S^\chi(x) = \{c, r', T) \leftarrow S(x); R \leftarrow \mathrm{ROC}(T);$$
$$r \leftarrow \mathrm{RSC}(r') : (c, r; R)\}$$

A *distinguisher* is a polynomial sized oracle circuit family $D = \{D_x\}_{x \in L)}$. We write $D_x^R(c, r)$ for the output of circuit D_x when given oracle R and inputs c, r.

Definition 5.3 We say a that simulator S is a P-simulator for a verifier V' over L if for every distinguisher D, every witness selector W, every constant d and all sufficiently long $x \in L$ it is the case that

$$|\Pr\left[(c, r; R) \leftarrow S^\chi(x) : D_x^R(c, r) = 1\right] - \Pr\left[(c, r; R) \leftarrow \mathrm{rview}(V, P_{W(x)}, x) : D_x^R(c, r) = 1\right]| < |x|^{-d}$$

.

Finally, we say that P defines a (computational) zero knowledge protocol over L in the random oracle model if for every verifier V' there exists a P-simulator for V' over L. (P, V) is a zero knowledge proof for L, in the random oracle model and with error ϵ if it is a proof system for L with error ϵ and P defines a zero knowledge protocol over L.

5.3 Protocol transformation

Bellare and Rogaway show how to transform an interactive ZK proof (P",V") into a non-interactive ZK proof (P,V) in random oracle, so that given $k(n) = \omega(log\ n)$ the the new proof achieves error $\epsilon(n) = 2^{-k(n)}$ while increasing the computing time and communicated bits by a factor of at most $O(k(n))$.

For simplicity, a three move protocol (P'', V'') is assumed, the moves being $P''_w \to V'' : \alpha$ followed by $V'' \to P''_w : b$ followed by $P''_w \to V'' : \beta$. Here b is a random bit that is used as a challenge.

Let $H : \{0,1\}^* \to \{0,1\}^2k$ be a random hash function. The basic idea of the transformation is to have the new prover P^H_w compute $\alpha_1 \leftarrow P''_w(x, e); \ldots; \alpha_{2k} \leftarrow P''_w(x, e)$ and set b'_i to i-th bit of $H(\alpha_1 \ldots \alpha_{2k}$. Then P^H_w continues to compute $\beta_1 \leftarrow P''_w(x, \alpha_1 b'_1); \ldots; \beta_{2k} \leftarrow P''_w(x, \alpha_{2k} b'_{2k})$; and sends $(\alpha_1, \ldots, \alpha_{2k}, \beta_1, \ldots, \beta_{2k})$ to V^H. V^H sets b_i to the i-th bit of $H(\alpha_1 \ldots \alpha_{2k})$ and accepts iff $\mathrm{ACC}_{V'}(\alpha_i b_i \beta_i, b_i) = 1$ for all i. This (P, V) is then claimed to be a zero knowledge proof system with error $2^{-k(n)}$. We skip the proof, but a sketch can be found in [2]. One thing that should be noted is that the protocol above is not *black box* zero knowledge, but weaker and thus does not contradict the result of Goldreich and Krawczyk [8] .

6 Where Random Oracle methodology fails

Canetti, Goldreich and Halevi give quite heavy criticism to the random oracle methodology [3]. They show that there exist encryption and signature schemes that are secure in the random oracle model, but have no secure implementation in the real model.

The basic idea is to create a malicious scheme, for example a signature scheme could be roughly something like:

- Calculate $f(msg)$, where f is a random function.
- If some sign of nonrandomness is detected, output the secret key.
- Else behave "normally" (output a real signature).

The idea is clear, substituting the ideal random function with some real world substitute will cause the scheme to behave nastily. It is also clear that this kind of schemes are highly artificial and the result of [3] does not mean that existing schemes are broken.

For more rigorous discription and proofs the reader is advised to see [3].

7 Conclusions

What are random oracles good for? In practice they seem to help produce simple and efficient protocols against which no attacks are known. We know, however, that proof of security in the random oracle model does not mean that a protocol would be secure. Even Goldreich (who, in a footmark in [3], suggests not including proofs of security in random oracle model in currently published work) agrees that random oracle paradigm can be useful as a sanity check. In [1] Bellare points out that some attacks, e.g. against SHA-1 will often treat the function as random themselves, so random oracle does show that this kind of generic attacks fail. The bottom line is that random oracle methodology is better than not formally modelling the security at all, but it should not be seen as a guarantee of security.

References

[1] Mihir Bellare, Practice-Oriented Provable-Security. In the Proceedings of First International Workshop on Information Security (ISW 97), Lecture Notes in Computer Science Vol. 1396, E. Okamoto, G. Davida and M. Mambo eds., Springer Verlag, 1998.

[2] Mihir Bellare and Phillip Rogaway, Random oracles are practical: a paradigm for designing efficient protocols. Proceedings of the

First Annual Conference on Computer and Communications Security, ACM, 1993.

[3] Ran Canetti, Oded Goldreich, Shai Halevi, The Random Oracle Methodology, Revisited. Revised version, February 2001.

[4] Cynthia Dwork, Moni Naor, Omer Reingold, Larry Stockmeyer, Magic Functions. April 2000.

[5] Amos Fiat and Adi Shamir, How to prove yourself: Practical solutions to identification and signature problems. CRYPTO 86, 1987.

[6] Oded Goldreich, Shafi Goldwasser, and Silvio Micali, How to Construct Random Funcions. Journal of the ACM, Vol. 33, No. 4, 1986.

[7] Oded Goldreich, Shafi Goldwasser, and Silvio Micali, On the cryptographic applications of random functions, CRYPTO 85 Proceedings, LNCS 196, Springer, 1985.

[8] O. Goldreich and H. Krawczyk, On the composition of zero knowledge proof systems. SIAM J. Computing 25(1), pp. 169-192, 1996.