Cryptography and Security Protocols **IST-UL, 2024/2025**

The Random Oracle Model

Lecturer: João Ribeiro

Recommended reading: Katz-Lindell, Section 5.5, and Boneh-Shoup, Section 8.10.

Introduction

There are several practical protocols based on hash functions that cannot be proven secure based only on the assumption that the hash function is collision-resistant. For many applications, it is not clear how to distill the security proof into simple and reasonable properties of the hash function, which we could then try to achieve based on the conjectured hardness of certain mathematical problems.

This leaves us with two options. One option is to throw away those protocols and focus only on "provably secure" cryptography. A few problems arise – "provably secure" protocols are sometimes too inefficient to be used in practice, and sometimes we do not know any such protocols. Another option is to embrace such protocols. But this is dangerous without rigorous analysis. We should not accept the fact that no attack has yet been found for some ad hoc protocol as good justification for its security.

In these notes, we explore a third, middle-ground option. Namely, we work in an idealized model, where we replace the hash function by a completely random function which can be queried on arbitrary inputs. This is called the *Random Oracle Model* (ROM), and it captures the way many practitioners think about cryptographic hash functions. The ROM was introduced in seminal work of Bellare and Rogaway [BR93].

Of course, real-world hash functions are not random oracles. Therefore, security proofs in the ROM are only meant to be used as heuristics. A security proof in the ROM shows that any attack on the protocol must exploit the structure of the real-world hash function it is instantiated with. In particular, when instantiating the random oracle in the real-world, we must choose a hash function that we think emulates the random oracle sufficiently well. There exist many settings where we only know security proofs in the ROM.

1 The ROM in more detail

We model the random oracle as an oracle $\mathcal{H}$ that algorithms can query on arbitrary inputs $m \in \{0,1\}^*$. More precisely, set an output length ℓ. On a query m, $\mathcal{H}$ proceeds as follows:

- If m has been queried before, $\mathcal{H}$ outputs the same as in the previous m query.

- If m has not been queried before, $\mathcal{H}$ samples $z \leftarrow \{0,1\}^\ell$ and outputs z.

It is easy to adapt security definitions to the ROM. Essentially, we give oracle access to $\mathcal{H}$ to all players. For example, we could define security for a MAC $(\mathsf{Gen}, \mathsf{MAC}^{\mathcal{H}}, \mathsf{Ver}^{\mathcal{H}})$ in the ROM via the following game for an adversary $\mathcal{A}^{\mathcal{H}}$:

1. A key $k \leftarrow \mathsf{Gen}(1^n)$ is generated.
2. The adversary $\mathcal{A}^{\mathcal{H}}(1^n)$ has access to an authentication oracle $\mathcal{O}_k$ which on input $m\{0,1\}^*$ outputs $\mathsf{MAC}^{\mathcal{H}}(k,m)$. Note that the adversary and the MAC algorithm have query access to the random oracle $\mathcal{H}$.
3. Let Q denote the set of messages queried by $\mathcal{A}$ to $\mathcal{O}_k$. Then, $\mathcal{A}$ outputs $(m^\star, t^\star)$ and wins if and only if $m^\star \notin Q$ and $\mathsf{Ver}^{\mathcal{H}}(k, m^\star, t^\star) = 1$. Note that the Ver algorithm has query access to the random oracle $\mathcal{H}$.

Then, we say that the MAC is *secure in the ROM* if for any PPT adversary $\mathcal{A}$ there exists a negligible function $\varepsilon(n)$ such that $\mathcal{A}^{\mathcal{H}}(1^n)$ wins the game above with probability at most $\varepsilon(n)$.

2 A MAC in the ROM

Now that we have a definition of a secure MAC in the ROM above, let's try to construct such a MAC. Here is a very simple construction:

- $k \leftarrow \{0,1\}^n$
- $\mathsf{MAC}^{\mathcal{H}}(k,m) = \mathcal{H}(k\|m)$

Theorem 1 *The MAC above is secure in the ROM.*

Proof: To win the MAC game, the adversary $\mathcal{A}^{\mathcal{H}}$ must produce $m^\star$ that was not queried to the authentication oracle $\mathcal{O}_k$ and a tag $t^\star$. Note that any query to $\mathcal{H}$ that is not of the form $s\|m^\star$ for some $s \in \{0,1\}^n$ is independent of $\mathcal{H}(k\|m^\star)$, and so can be discarded without loss of generality. Suppose that $\mathcal{A}^{\mathcal{H}}$ makes q queries to $\mathcal{H}$ of the form $s_i\|m^\star$ for some $s_1, \ldots, s_q \in \{0,1\}^n$.

If $k \neq s_i$ for all i, then $\mathcal{H}(k\|m^\star)$ is independent of $\mathcal{H}(s_i\|m^\star)$ for all i. Therefore, in this case the tag $t^\star$ can only be valid (i.e., equal $\mathcal{H}(k\|m^\star)$) with probability at most $2^{-\ell}$. On the other hand, if $k = s_i$ for some i then $\mathcal{A}$ can produce a valid tag with non-negligible probability. But, by a union bound, this happens only with probability

$$\Pr[\exists i, k = s_i] \leq \sum_{i=1}^{q} \Pr[k = s_i] = q/2^n.$$

Therefore, the probability that $\mathcal{A}^{\mathcal{H}}$ wins the MAC game is at most $2^{-\ell} + q/2^n$, which is negligible in n if, say, $q \leq 2^{n/2}$ and ℓ is superlogarithmic in n. ■

2.1 How can we instantiate this MAC in the real world?

Now that we have a simple secure MAC in the ROM, we would like to use it in the real world. To that end, we must replace the random oracle by a real cryptographic hash function. How should we do this? Perhaps an arbitrary collision-resistant hash function would be good enough in practice?

We saw how to construct collision-resistant hash functions using the Merkle-Damgård transform. This transform is widely used in practice, starting with a fast collision-resistant compression function. Therefore, it may seem like a good idea to use such a hash function to instantiate the simple MAC above.

It turns out that this is a terrible idea! If H is constructed via the Merkle-Damgård transform from some compression function h and we use it to instantiate the MAC above as $\mathsf{MAC}(k, m) = H(k\|m)$, then we become vulnerable to *extension attacks*. More precisely, given only knowledge of $H(k\|m)$ and the lengths of k and m, the sequential structure of the Merkle-Damgård means that we can compute $H(k\|m\|y)$ for some (appropriately structured) additional block y. This is a valid tag for the new message $m^\star = m\|y$.

The question of how to instantiate the random oracle appropriately, and whether this is even always possible, is natural and has been studied extensively. In particular, we have known for a while that there exist cryptographic schemes that are secure in the ROM, but which become insecure as soon as we replace the random oracle by *any* family of hash functions [CGH04].

3 More on rigorous cryptography vs. the real world

As showcased by the ROM, the interaction between modern rigorous cryptography and the real world is inevitably imperfect. Cryptographers do not always agree on the best course of action.

If you are interested in the history and culture of cryptography, I suggest you read the Koblitz-Menezes position paper on the role of modern cryptography, and responses by Goldreich and Damgård.

For more fun reading on the cultural clashes within cryptography, have a look at an internal NSA account of Eurocrypt 1992 here (it starts on page 15 of the PDF).

References

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, CCS '93, page 62–73, New York, NY, USA, 1993. Association for Computing Machinery.

[CGH04] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *J. ACM*, 51(4):557–594, July 2004.