The Random Oracle Model

Advanced Topics in ~~Cybersecurity~~Cryptography (7CCSMATC)

Martin R. Albrecht

Main References

Mihir Bellare and Phillip Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In: *ACM CCS 93*. Ed. by Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby. ACM Press, Nov. 1993, pp. 62–73. DOI: 10.1145/168588.168596

Arno Mittelbach and Marc Fischlin. Chapter 9: The Full Power of Random Oracles. In: *The Theory of Hash Functions and Random Oracles - An Approach to Modern Cryptography*. Information Security and Cryptography. Springer, 2021. ISBN: 978-3-030-63286-1. DOI: 10.1007/978-3-030-63287-8. URL: https://doi.org/10.1007/978-3-030-63287-8

The Random Oracle Model

Summary

The Random Oracle Model (ROM) defines (some) hash functions as random functions.

- The idea is to design definitions and protocols in an embellished model of computation in which all parties, including the adversary, are given access to a common random oracle.
- This is a map $H : \{0,1\}^* \to \{0,1\}^n$ that associates a random bitstring to each and every string.
- We then prove the protocol secure in this enriched model of computation.
- As a heuristic final step, we instantiate the oracle H by something like a cryptographic hash function.

Analogy: Dynamic Malware Analysis

We start with some intuition about how such ROM proofs go by means of an analogy to dynamic malware analysis.

IND-CCA (Recap) I

IND:

The thought experiment in a typical game-based cryptographic proof starts by assuming that there is indeed an adversary that breaks the security goal of our cryptographic construction.

- Assume this adversary can decide if some message m_0 or some message m_1 was encrypted in ciphertext c.
- We are not even asking the adversary to decrypt c but we are merely asking it to decide which of two messages of its choosing we encrypted.
- If it cannot even do that, it cannot decrypt or learn anything about the underlying plaintext of a ciphertext. So this is the adversary's goal: to distinguish.

IND-CCA (Recap) II

CCA:

We need to decide what capabilities our hypothetical adversary has.

- The adversary gets to ask for decryptions of any ciphertexts it wants except for the "target" ciphertext c we are challenging it to make a distinguishing decision about.

We taunt the adversary:

- *"We're giving you the ability to decrypt anything you like except this one ciphertext but you still cannot decrypt it. In fact, we let you choose two messages m_0 and m_1 and we will encrypt one of them for you, you won't even be able to decide which one we picked. Good luck, mate!"*

IND-CCA (Recap) III

IND-CCA	C(m_0, m_1)	D(c)
1 : $\mathcal{C} \leftarrow \emptyset$	1 : **if** $\lvert m_0 \rvert \neq \lvert m_1 \rvert$ **then**	1 : **if** $c \in \mathcal{C}$ **then**
2 : $\mathsf{pk}, \mathsf{sk} \leftarrow\!\$\ \mathsf{KeyGen}(1^\lambda)$	2 : **return** $\perp$	2 : **return** $\perp$
3 : $b \leftarrow\!\$\ \{0,1\}$	3 : $c \leftarrow\!\$\ \mathsf{Enc}(\mathsf{pk}, m_b)$	3 : **return** $\mathsf{Dec}(\mathsf{sk}, c)$
4 : $b' \leftarrow \mathcal{D}^{\mathrm{C},\mathrm{D}}(\mathsf{pk})$	4 : $\mathcal{C} \leftarrow \mathcal{C} \cup \{c\}$	
5 : **return** $b = b'$	5 : **return** c	

$$\mathsf{Adv}^{\text{ind-cca}}_{C,D}(\mathcal{D}) = \Pr[\text{IND-CCA}^{\mathcal{D}} = 1].$$

Analogy: Dynamic Malware Analysis I

Think of the adversary as a piece of malware.

- We put it in a sandboxed virtual machine.
- Use power over the sandbox to subject the adversary to various conditions and observe its behaviour.

$\Rightarrow$

The first goal of such a cryptographic security proof is to show that we can simulate the "world" that our malware-née-adversary expects.

- Like malware our adversary may decide to behave differently when it detects a simulation to avoid being analysed.

"To defend and discover 0-day attacks, sandbox system is used more and more widely. It quite efficient because it does not heavily rely on virus signatures. When a sample need to be analyzed, it will be put into a virtual machine. It makes believe that this is a real victim machine and run the sample. And the monitor program will record all the behaviors that the sample shows. If any harmful behavior is found, this sample will be marked as malware or virus.
[...]
You can imagine the malware and virus do not want to be treated like that, they try many ways to defeat sandbox system. One way that is used most commonly is to detect if there exists any hooks in the system. If there is, they will consider this as a trap and could behave accordingly."
— uty & saman in Phrack 0x45

Analogy: Dynamic Malware Analysis II

Task Ahead

In our setting the adversary expects two things – a random oracle and a decryption oracle – and we better simulate those (nearly) perfectly.

Analogy: Dynamic Malware Analysis III

In this view, the ROM is Hashing-as-a-Service (HaaS)

- Instead of specifying a hash function like SHA3 with all details allowing anyone ship their own implementation, we define some API with a single calling point $H()$:
- Put some string in, receive a digest back: $y := H(x)$.
- In the ROM, our HaaS realises a perfect hash function: for each fresh input x it returns a completely random y; the only way to know the output y is to call $H(x)$.
- Of course, if we call $H()$ again on the same x we get the same y, just as we would expect from a hash function.

Random Oracle

We have something "random" (random output) from an "oracle" (can only call the API).

Analogy: Dynamic Malware Analysis iv

- Similar to ransomware countermeasures that intercept calls to the cryptographic API provided by the OS.
- The difference is that ransomware may implement and ship its own cryptography, but in our thought experiment the only way to get access to $H()$ is via our API.
- Another practical analogy: HMAC with a secret key running on an HSM.

"A primary strength of crypto-ransomware is its ability to use well-known and reputable crypto libraries to perform encryption. Interestingly, early families such as CryptoLocker and CryptoWall relied on Microsoft CryptoAPI, which may be considered a drawback since it is trivial to hook encryption routines. That makes early detection easier and allows intercepting session keys. Other families switched to statically linking the encryption functions to address this problem."

— Bromium. Understanding Crypto-Ransomware

```
$cur = 'plaintext'
$cur = md5($cur)
$salt = randbytes(20)
$cur =  hmac_sha1($cur, $salt)
$cur = cryptoservice::hmac($cur)
$cur = scrypt($cur, $salt)
$cur = hmac_sha256($cur, $salt)
```

— Alec Muffett (then Facebook) at Real World Crypto 2015.

Analogy: Dynamic Malware Analysis v

Returning to our proof sketch, we want to show that the ability to decrypt every ciphertext except c does not buy the adversary anything.

- We can accomplish this by making our construction dependent on our API so that the only way to produce a valid ciphertext is to call $H()$ on the message (and other randomness used during encryption), everything else produces an error on decryption.
- Then the adversary has two choices:
  - submit whatever it wants for decryption which will just produce an error or
  - call our API $H()$ when producing a ciphertext.
- The key observation now is that in the latter case it sends us the message (and associated randomness) it might ask us to decrypt later.
- So we can provide plaintexts in response to correctly formed ciphertexts: We are cheating by knowing the answers before seeing the question.

IND-CCA Proof Strategy Conclusion:

- If there was an adversary against our scheme that requires a decryption oracle we can run this adversary against our scheme without actually having access to such a decryption oracle by simulating it using the information the adversary helpfully sends us via calls to $H()$.
- This implies that CCA attacks, i.e. active attacks – in the ROM and for schemes where such proofs exists –, are no more powerful than CPA attacks, i.e. passive attacks.
- To drive home this point, this is not a claim that we prevent the adversary from running specific attack strategies but it rules out any attack using such a decryption oracle.
- If we can fake it, it offers no advantage.

More Power: Planting Solutions

Once we have HaaS we can play all kinds of tricks with the adversary.

- For example, we can start cheating and send specifically chosen answers in response to strategically chosen queries.
- When $H()$ is used to check the integrity of some input x against some known digest y we can simply make our API return y on input x or z, it is up to us.
- This is known as "programming the random oracle".
- An analogy from dynamic analysis would be to provide bad randomness to a piece of malware to break its encryption or to return incorrect time/date information from a system call to trigger some behaviour.

Hash-and-Sign

We will discuss this strategy in more detail in a future lecture.

Summary

The Random Oracle Model states that the only way to know what H will output is to query the oracle:

- This allows the "oracle provider" to know x when $H(x)$ was called.
- In the Random Oracle Model $\mathcal{A}$ has no "ground truth" to compare against. This allows us to "plant" certain values as return values for $H(x)$.

The ROM is Unsound

Beef I

"[I]n our opinion, the Random Oracle Model has caused more harm than good, because many people confuse it for the "real thing" (while it is merely an extremely idealized sanity check). Needless to say, as in the case of the bronze serpent, the blame is not with its creators (who meant well), and the danger could not have been foreseen a priori. Still, given the sour state of affairs, it seems good to us to abolish the Random Oracle Model. At the very minimum, one should issue a fierce warning that security in the Random Oracle Model does not provide any indication towards security in the standard model."

Oded Goldreich. On Post-Modern Cryptography. Cryptology ePrint Archive, Report 2006/461. 2006. URL: `https://eprint.iacr.org/2006/461`

Beef II

"Of course, not everyone likes the RO model. Indeed some people are so anti-RO that they don't even want to call proofs in the RO model 'proofs'. They'll use words like 'heuristic arguments'. I personally find this a little bit silly. A proof in the RO model already has a name: it's called a 'proof'. Something doesn't stop being a proof because you're unhappy with the definition; it stops being a proof when it has a bug."

"Serious criticism of the RO model begins with the 1998 paper of Canetti, Goldreich, and Halevi. But concern over random oracles goes back much further. Adi Shamir once mentioned to me that the journal submission for Feige-Fiat-Shamir a (1988) dropped the RO-model proof of the Fiat-Shamir scheme (1986) because a referee wanted the thing removed."

Phillip Rogaway. Practice-Oriented Provable Security and the Social Construction of Cryptography. In: *IEEE Secur. Priv.* 14.6 (2016), pp. 10–17. DOI: 10.1109/MSP.2016.122. URL: https://doi.org/10.1109/MSP.2016.122

Oded Goldreich

Oded Goldreich is a founding figure of cryptography.

- He received the Knuth prize in 2017 for “fundamental and lasting contributions to theoretical computer science in many areas including cryptography, randomness, probabilistically checkable proofs, inapproximability, property testing as well as complexity theory in general.”
- His textbooks *Foundations of Cryptography* are considered *the* books by some.

Unsound and Ubiquitous

Unsound

Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. In: *Journal of the ACM* 51.4 (July 2004), pp. 557–594. ISSN: 0004-5411 (print), 1557-735X (electronic)

Ubiquitous

- Many widely relied upon primitives protocols are proven secure only in the random oracle model
- Essentially, the Internet runs on the Random Oracle Model

Unsound

$H : \{0,1\}^\lambda \to \{0,1\}^\lambda$ is a hash function

$\mathrm{Enc}(k, m)$

$r \xleftarrow{\$} \{0,1\}^\lambda$

$M \leftarrow m$ as the encoding of a boolean circuit

if $M(r) = H(r)$ **then**

  $iv \leftarrow k$

else

  $iv \xleftarrow{\$} \{0,1\}^\lambda$

$c \leftarrow$ encrypt m under CTR mode with iv, k

return iv, c

Credit: https://crypto.stackexchange.com/a/109272/76582

- If H is modelled as a random oracle, then the construction is CPA-secure. With overwhelming probability, Enc chooses an r that no one has ever queried to the random oracle, so $H(r)$ is independent of whatever $M(r)$ is. With overwhelming probability, Enc behaves just like random-IV CTR mode, which is CPA-secure.

Unsound

$H : \{0,1\}^\lambda \to \{0,1\}^\lambda$ is a hash function

$\mathrm{Enc}(k, m)$

$r \xleftarrow{\$} \{0,1\}^\lambda$

$M \leftarrow m$ as the encoding of a boolean circuit

if $M(r) = H(r)$ **then**

  $iv \leftarrow k$

else

  $iv \xleftarrow{\$} \{0,1\}^\lambda$

$c \leftarrow$ encrypt m under CTR mode with iv, k

return iv, c

Credit: https://crypto.stackexchange.com/a/109272/76582

- If H is any public, efficiently computable function, then the construction is not CPA-secure. Ask for an encryption where the plaintext is a boolean circuit that implements H. The resulting ciphertext will contain the encryption scheme's key.

Those Hash Function Calls

The reason why your favourite cryptographic construction has a hash function call is probably not the reason you think it is.

References I

[BR93] Mihir Bellare and Phillip Rogaway. **Random Oracles are Practical: A Paradigm for Designing Efficient Protocols.** In: *ACM CCS 93*. Ed. by Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby. ACM Press, Nov. 1993, pp. 62–73. DOI: 10.1145/168588.168596.

[CGH04] Ran Canetti, Oded Goldreich, and Shai Halevi. **The random oracle methodology, revisited.** In: *Journal of the ACM* 51.4 (July 2004), pp. 557–594. ISSN: 0004-5411 (print), 1557-735X (electronic).

[Gol06] Oded Goldreich. **On Post-Modern Cryptography.** Cryptology ePrint Archive, Report 2006/461. 2006. URL: https://eprint.iacr.org/2006/461.

References II

[MF21] Arno Mittelbach and Marc Fischlin. **Chapter 9: The Full Power of Random Oracles.** In: *The Theory of Hash Functions and Random Oracles - An Approach to Modern Cryptography*. Information Security and Cryptography. Springer, 2021. ISBN: 978-3-030-63286-1. DOI: 10.1007/978-3-030-63287-8. URL: https://doi.org/10.1007/978-3-030-63287-8.

[Rog16] Phillip Rogaway. **Practice-Oriented Provable Security and the Social Construction of Cryptography.** In: *IEEE Secur. Priv.* 14.6 (2016), pp. 10–17. DOI: 10.1109/MSP.2016.122. URL: https://doi.org/10.1109/MSP.2016.122.