Cryptographic Hash Functions
Random Oracles

CS / ECE 407

Today's objectives

Introduce cryptographic hash function

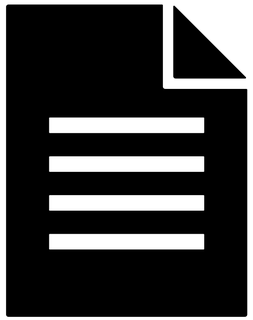
Understand the differences between hash functions and ciphers

Define the random oracle model; understand role of *idealized* models

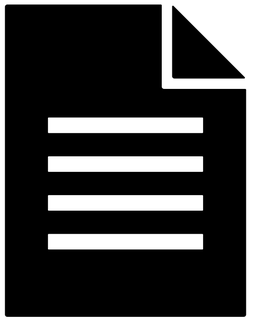
See some applications of hash functions

Alice

Bob



Are these files the same?



A.txt

B.txt

Hash Functions

There are many concrete security properties for hash functions, and we will see some of them later

Today, we will focus on a particularly intuitive notion

Alice **Bob**

Modern Cryptographic approach to provable security

0) Define assumptions
1) Define Security
2) Specify a system
3) Prove that if assumptions hold, system is secure

There’s a public function H, and on any input H returns a random output

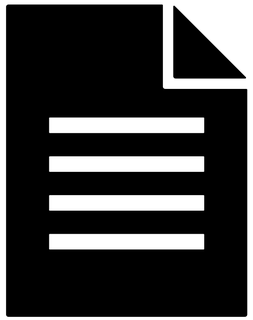
Desired property for Random Oracle

A random oracle H : $\{0,1\}^* \rightarrow \{0,1\}^\lambda$ is a uniformly random function

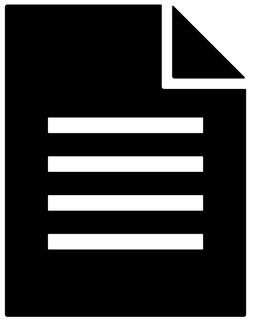
$$\left\{ f \;\middle|\; f \leftarrow \text{uniform function from } \{0,1\}^* \rightarrow \{0,1\}^\lambda \right\}$$

Alice

Bob



Are these files the same?



A.txt

1TB

B.txt

1TB

If H is RO, then hashing the files and comparing digests gives the right answer with overwhelming probability

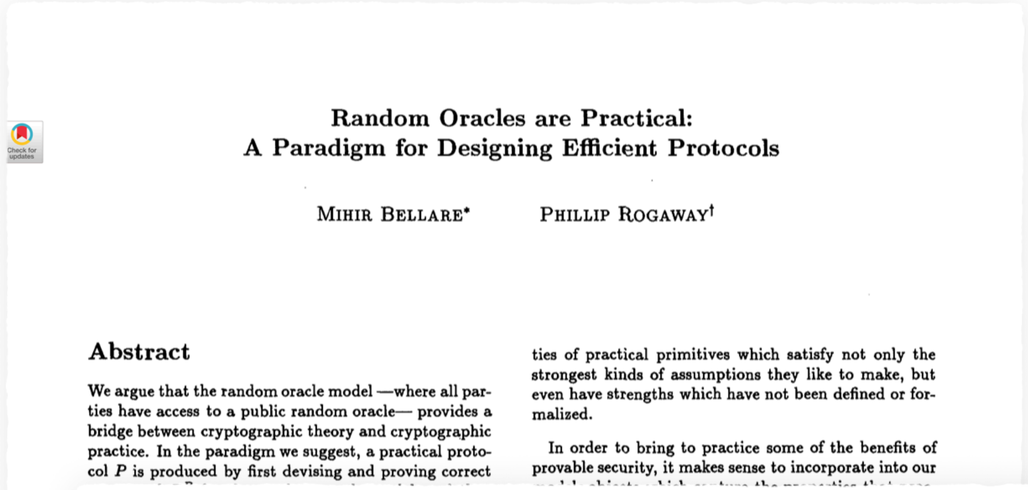
Desired property for Random Oracle

A random oracle H : $\{0,1\}^* \rightarrow \{0,1\}^\lambda$ is formalized as an oracle that implements a truly random function

$$\left\{ f \;\middle|\; f \leftarrow \text{uniform function from } \{0,1\}^* \rightarrow \{0,1\}^\lambda \right\}$$

Compare to PRF security

$$\left\{ F(k,\cdot) \;\middle|\; k \leftarrow \{0,1\}^\lambda \right\} \approx \left\{ f \;\middle|\; f \leftarrow \text{uniform function from } \{0,1\}^n \rightarrow \{0,1\}^m \right\}$$



Random Oracles are Practical:
A Paradigm for Designing Efficient Protocols

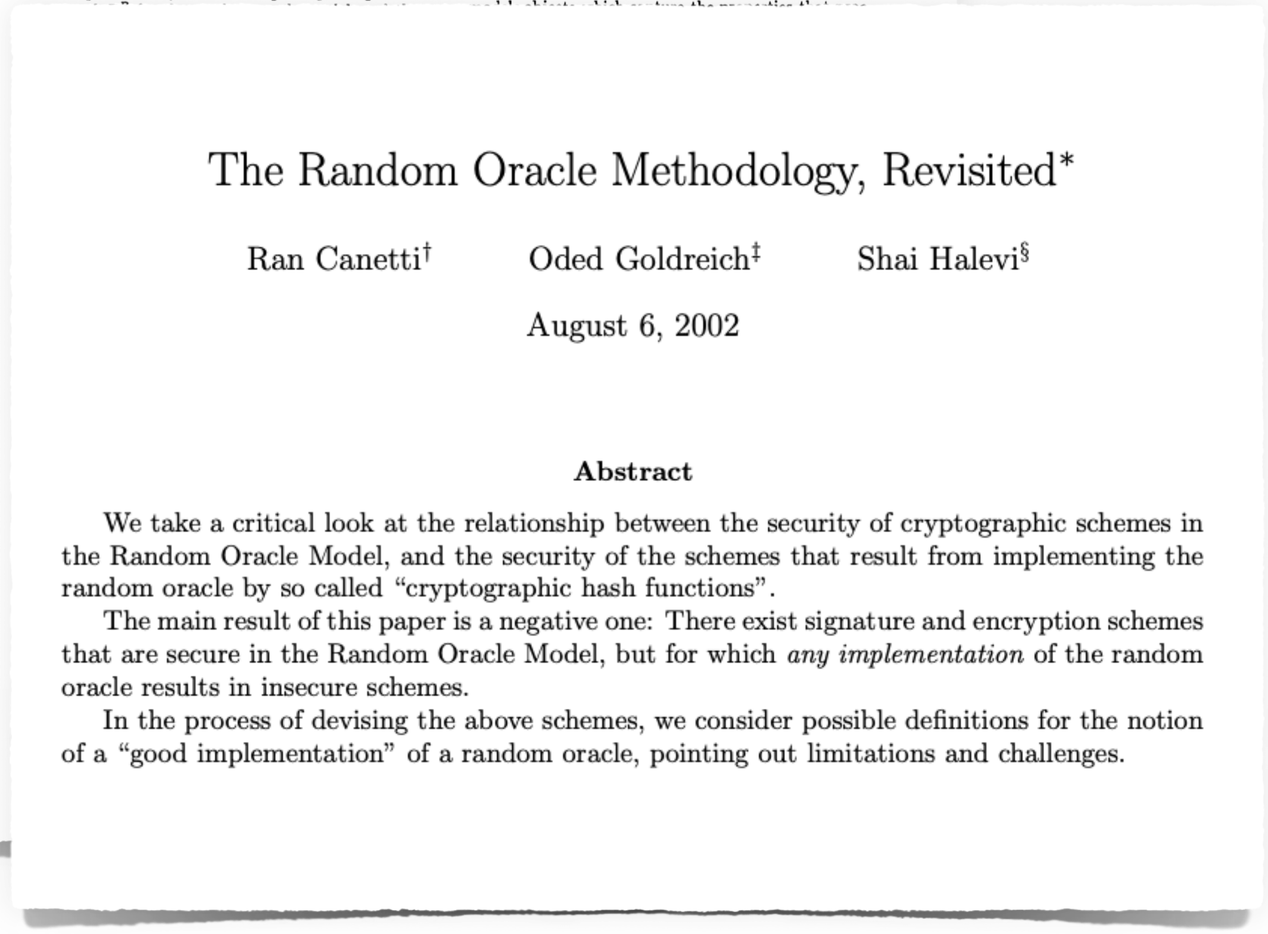
Mihir Bellare* Phillip Rogaway†

Abstract

We argue that the random oracle model —where all parties have access to a public random oracle— provides a bridge between cryptographic theory and cryptographic practice. In the paradigm we suggest, a practical protocol P is produced by first devising and proving correct

ties of practical primitives which satisfy not only the strongest kinds of assumptions they like to make, but even have strengths which have not been defined or formalized.

In order to bring to practice some of the benefits of provable security, it makes sense to incorporate into our



The Random Oracle Methodology, Revisited*

Ran Canetti† Oded Goldreich‡ Shai Halevi§

August 6, 2002

Abstract

We take a critical look at the relationship between the security of cryptographic schemes in the Random Oracle Model, and the security of the schemes that result from implementing the random oracle by so called "cryptographic hash functions".

The main result of this paper is a negative one: There exist signature and encryption schemes that are secure in the Random Oracle Model, but for which *any implementation* of the random oracle results in insecure schemes.

In the process of devising the above schemes, we consider possible definitions for the notion of a "good implementation" of a random oracle, pointing out limitations and challenges.

There is no such thing as a random oracle!

But we still use them all the time

How does this relate to provable security?

Alice **Bob**

Modern Cryptographic approach to provable security

0) Define assumptions

1) Define Security

2) Specify a system

3) Prove that if assumptions hold, system is secure

But random oracle “assumption” is always false, for any fixed function H

Alice **Bob**

Modern Cryptographic approach to provable security

0) Define assumptions

1) Define Security

2) Specify a system

3) Prove that if assumptions hold, system is secure

But random oracle “assumption” is always false, for any fixed function H

Doesn’t this mean we can prove anything we want?

Random Oracle Model

Alice **Bob**

There’s a public function black-box function H, and on any input H returns a random output

Random Oracle Model

Alice **Bob**

There's a public function black-box function H, and on any input H returns a random output

This is a bit like changing the parties' model of computation

before, parties were simply poly-bounded algorithms — we were in the **standard model**

In ROM, parties are poly-bounded computations *with access to global service **H***

Random Oracle Model

Alice

Random Oracle Heuristic

Bob

0) Define assumptions

1) Define Security

2) Specify a system where parties have access to RO

3) Prove that if assumptions hold, system is secure

Random Oracle Model

Alice

Random Oracle Heuristic

Bob

0) Define assumptions

1) Define Security

2) Specify a system where parties have access to RO

3) Prove that if assumptions hold, system is secure

4) Replace all instances of RO by concrete function H, and pray

Practitioner

Theoretician

“Almost all known attacks on the RO heuristic are contrived. A proof in the ROM is almost as good as a proof in the standard model!”

Practitioner

“Almost all known attacks on the RO heuristic are contrived. A proof in the ROM is almost as good as a proof in the standard model!”

Theoretician

“A proof in the ROM does not demonstrate security, but it might help understand what a proof in the standard model would look like.”

Hash Functions — SHA256

Hash Functions — SHA256

One point: SHA has fixed block size, 512 bits

Domain extension via Merkle-Damgard

Some Applications of Hash Functions

- MACs
- Password security
- Version control
- Outsourced Data Storage
- Blockchains
- …

Message Authentication Codes

A MAC scheme is a pair of algorithms tag such that:

Real

```
k ← K

get(m):
  return tag(k, m)

check(m, t):
  return tag(k, m) = t
```

$\overset{c}{\approx}$

Fake

```
k K
S ← empty-set

get(m):
  t ← tag(k, m)
  S ← S ∪ {(m, t)}
  return t

check(m, t):
  return (m, t) ∈ S
```

Message Authentication Codes

We saw that for PRF F with n-bit input, MAC on n-bit messages are trivial

But for longer messages, we had to be more clever

Let F be a PRF with in = out = λ. ECBC-MAC refers to the following scheme:

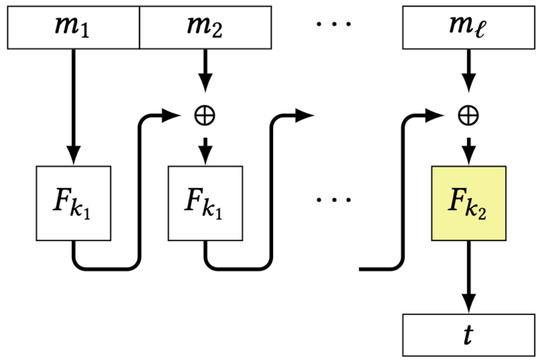
$\text{ECBCMAC}^F\Big((k_1, k_2), m_1 \cdots m_\ell\Big)$:

$t := 0^\lambda$

for $i = 1$ to $\ell - 1$:

  $t := F(k_1, m_i \oplus t)$

return $F(k_2, m_\ell \oplus t)$



Message Authentication Codes

We saw that for PRF F with n-bit input, MAC on n-bit messages are trivial

But for longer messages, we had to be more clever

Let F be a PRF with in = out = λ. ECBC-MAC refers to the following scheme:

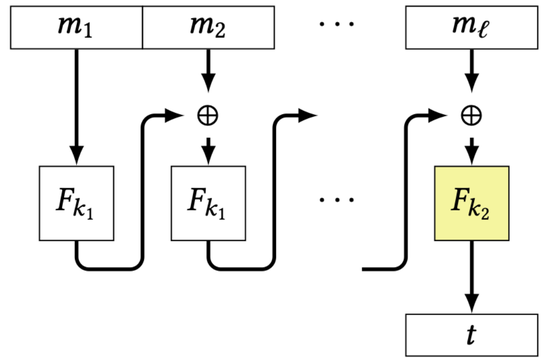
$\text{ECBCMAC}^F\left((k_1, k_2), m_1 \cdots m_\ell\right)$:

$t := 0^\lambda$

for $i = 1$ to $\ell - 1$:

  $t := F(k_1, m_i \oplus t)$

return $F(k_2, m_\ell \oplus t)$



HMAC:

```
tag(K, m):
  return H(K || m)
```

Application: Password Databases

Username	Password
milo-dog-2016	password1
riley-cat-2020	milo-is-cool
…	…
davida	super-secure-pwd

Application: Password Databases

Username	Password
milo-dog-2016	H(password1)
riley-cat-2020	H(milo-is-cool)
…	…
davida	H(super-secure-pwd)

Why is it better to store hashes than cleartext passwords?

Application: Password Databases

Username	Salt	Password
milo-dog-2016	s0	H(s0, password1)
riley-cat-2020	s1	H(s1, milo-is-cool)
…	…	…
davida	s2	H(s2, super-secure-pwd)

Why is it better to store salted hashes than regular hashes?

Today's objectives

Introduce cryptographic hash function

Understand the differences between hash functions and ciphers

Define the random oracle model

See some application of hash functions to MACs