Random Oracles are Practical: A Paradigm for Designing Efficient Protocols

Mihir Bellare     Phillip Rogaway


presented by: Eun-kyung Kim
Agenda

• Definitions
  ‣ Random Oracle Model
  ‣ Notations
• Encryption
  ‣ Polynomial Security
  ‣ Chosen Cipher-text Security
  ‣ Non-Malleability
• Signatures
• Instantiation
Abstract

• Random Oracle Model (ROM)
  ‣ an ideal mathematical model for a hash function
  ‣ The ROM that they claim more accurately models the real world while simultaneously making proofs easier

• Motivation
  ‣ Large gap between the theoreticians’ and practitioners’ works and views
    ‣ theoretical work gains security at cost of efficiency
    ‣ theorists build PRFs from one-way functions, while in practice, one-way functions are built from PRFs
      ‣ PRF: Pseudo Random Functions
Random Oracle Paradigm

1. Find a formal definition of the problem in the random oracle model

2. Devise a protocol that solves the problem

3. Prove the protocol satisfies definition

4. Replace oracle accesses by computation of a real function (e.g., hash function)
Notations

- \( G: \{0,1\}^* \rightarrow \{0,1\}^\infty \) is a random generator
- \( k \) is the security parameter
- \( H: \{0,1\}^* \rightarrow \{0,1\}^k \) is a random hash function
- \( f \) is a trapdoor permutation with inverse \( f^{-1} \)
- \( G(r) \oplus x \) denotes the bitwise XOR of \( x \) with the first \( |x| \) bits of the output of \( G(r) \)
- \( || \) denotes concatenation
Encryption

• Goal
  ‣ possible but impractical in the standard setting become practical in the random oracle setting

• Scheme
  ‣ extend the notion of public key encryption to the random oracle model

  ‣ PPT generator $G:1^k \rightarrow (E,D)$
    ‣ PPT: Probabilistic, Polynomial Time

  ‣ encryption: $y \leftarrow E^R(x)$
  ‣ decryption: $x \leftarrow D^R(y)$
Polynomial Security

• by Goldwasser, Micali’s notion (1984)

• $B_f$ denotes a hard core predicate for $f$

• $E(x) = f(r_1) \ || \ ... \ || f(r_{|x|})$
  - $r_i$ are randomly chosen such that $B_f(r_i) = x_i$

  ▶ encryption length: $O(k \cdot |x|)$
  ▶ encryption effort: $O(f \cdot |x|)$
  ▶ decryption effort: $O(f^{-1} \cdot |x|)$

  ▶ It is not practical!
Polynomial Security

• in Random Oracle Model

• Given CP-adversary (F,A) chosen plaintext security in the model is:

\[
\Pr\left[ R \leftarrow 2^\infty; \\
(E,D) \leftarrow G(1^k); \\
(m_0,m_1) \leftarrow F^R(E); \\
b \leftarrow \{0,1\}; \\
y \leftarrow E^R(m_b); \\
A^R(E,m_0,m_1,y) = b \right] \leq \frac{1}{2} + k^{-\omega(1)}
\]
Polynomial Security

- \( E(x) = f(r) \ || \ G(r) \oplus x \)
  - \( E \) is the algorithm which on input \( x \) picks \( r \leftarrow d(1^k) \)
- encryption size \( O(|x| + k) \)
Chosen Ciphertext Security

- The scheme of the previous is not secure against RS-attack
  - Given “Rackoff-Simon”-adversary (F,A) chosen ciphertext security in this model is:

  \[
  \Pr[ R \leftarrow 2^x; \\
  (E,D) \leftarrow G(1^k); \\
  (m_0,m_1) \leftarrow F^{R,D}(E); \\
  b \leftarrow \{0,1\}; \\
  y \leftarrow E^R(m_b): \\
  A^{R,D}(E,m_0,m_1,y) = b] \leq \frac{1}{2} + k^{\omega(1)}
  \]

- Encryption by \( E(x) = f(r) \parallel G(r) \oplus x \parallel H(rx) \)
Non-Malleability

• An encryption algorithm is **malleable** if it is possible for an adversary to transform a cipher-text into another cipher-text which decrypts to a related plaintext.

• Non-Malleability is that given the cipher-text it is impossible to generate a different cipher-text so that the respective plain texts are related.
Non-Malleability

- Encryption by $E(x) = f(r) \ || \ G(r) \oplus x \ || \ H(rx)$
  - same as that of the previous

- Given adversary $(F,A)$ security in the sense of malleability is:

$$\begin{align*}
\text{Pr}[ \ R \leftarrow 2^\infty; \\
(E,D) \leftarrow G(1^k); \\
\pi \leftarrow F^R(E); \\
x \leftarrow \pi^R(1^k); \\
y \leftarrow E^R(x); \\
y' \leftarrow A^R(E,\pi, y); \\
\rho^R(x, D^R(y')) &= 1]
\end{align*}$$

is negligible!!
Results

- **Efficient Encryption**
  - $E^G(x) = f(r) \ || \ G(r) \oplus x$
    : achieves polynomial/semantic security
  - $E^{G,H}(x) = f(r) \ || \ G(r) \oplus x \ || \ H(rx)$
    : against chosen ciphertext attack, non-malleable
Signatures

- A digital signature scheme: \((G, S, V)\)
  - \(G\): generator
  - \(S\): signing algorithm
  - \(V\): verifying algorithm

- \(G: 1^k \rightarrow (PK, SK)\)
  - \(PK\): public key
  - \(SK\): secret key

- To sign message \(m\)
  - \(\sigma \leftarrow \text{Sign}^R(SK, m)\)

- To verify \((m, \sigma)\)
  - \(\text{Verify}^R(PK, m, \sigma) \in \{0,1\}\)
Signatures

• in Random Oracle Model

• Given signing adversary $F$, security is:

$$\Pr[R \leftarrow 2^x; \quad (PK, SK) \leftarrow G(1^k); \quad (m, \sigma) \leftarrow F_{R, \text{Sign}^k(SK, \cdot)}(PK); \quad \text{Verify}^R(PK, m, \sigma) = 1]$$

is negligible !!
Instantiation Tips

• Do not instantiate based on the protocol
  ‣ an appropriate instantiation should work for any protocol designed using a black box

• Avoid instantiations revealing internal structure
  ‣ e.g. MD5(x||y||z) can be easily computed given |x|, MD5(x), and z
  ‣ suggestions include:
    ‣ truncating output: h(x) = the first 64 bits of MD5(x)
    ‣ limiting input length: h(x) = MD5(x), where |x| ≤ 400
    ‣ non-standard use: h(x) = MD5(x||x)
Wrap-up

- A random oracle is a mathematical abstraction used in cryptographic proofs
  - In practice, random oracles are typically used to model cryptographic hash functions in schemes where strong randomness assumptions are needed of the hash function's output

- Random Oracle Paradigm
  - The idea is to make use of hash functions that are assumed in the analysis to behave randomly
  - This is a bridge between theory and practice