The Quantum Random Oracle Model

Ted Eaton

University of Waterloo

August 2, 2016

Theorem

Cryptographers kind of like proofs

Theorem

Cryptographers kind of like proofs

Proof.

Mathematicians like proofs

Theorem

Cryptographers kind of like proofs

Proof.

Mathematicians like proofs
Cryptographers are kind of mathematicians

Theorem

Cryptographers kind of like proofs

Proof.

Mathematicians like proofs

Cryptographers are kind of mathematicians

$\therefore$ Cryptographers kind of like proofs $\square$

Fundamental Question: How can we prove security?

Fundamental Question: How can we prove security?

- Define Security

Fundamental Question: How can we prove security?

- Define Security
- Define Protocol

Signature Scheme:

Signature Scheme:

- Two parties, Alice and Bob

Signature Scheme:

- Two parties, Alice and Bob
- Alice wants to send Bob a message

Signature Scheme:

- Two parties, Alice and Bob
- Alice wants to send Bob a message
- Bob wants to be 100% certain that message really came from Alice

Signature Scheme:

- Two parties, Alice and Bob
- Alice wants to send Bob a message
- Bob wants to be 100% certain that message really came from Alice
- Alice 'signs' the message in a way only she can, and sends the message, along with the signature, to Bob

Definition

A Signature scheme is a triple of algorithms -

Definition

A Signature scheme is a triple of algorithms -

- KEYGEN - Outputs a verification key vk and a signing key sk

Definition

A Signature scheme is a triple of algorithms -

- KEYGEN - Outputs a verification key vk and a signing key sk
- SIGN - Takes in a message M and a signing key sk and out puts a signature σ

Definition

A Signature scheme is a triple of algorithms -

- KEYGEN - Outputs a verification key vk and a signing key sk
- SIGN - Takes in a message M and a signing key sk and out puts a signature σ
- VRFY - Takes in a message M, a signature σ, and a verification key vk and outputs either ACCEPT or REJECT

Definition

A Signature scheme is a triple of algorithms -

- KEYGEN - Outputs a verification key vk and a signing key sk
- SIGN - Takes in a message M and a signing key sk and out puts a signature σ
- VRFY - Takes in a message M, a signature σ, and a verification key vk and outputs either ACCEPT or REJECT

Correctness property: If $(vk, sk) \leftarrow KEYGEN$ and $\sigma \leftarrow SIGN(M, sk)$ then $VRFY(M, \sigma, vk) \rightarrow ACCEPT$

Ted's super awesome signature scheme

Ted's super awesome signature scheme

- KEYGEN: Pick two large primes p and q. Let $N = p \times q$. $vk = N$, $sk = (p, q)$

Ted's super awesome signature scheme

- KEYGEN: Pick two large primes p and q. Let $N = p \times q$. $vk = N$, $sk = (p, q)$
- SIGN: Interpret M as an integer. Let $a = M \times p$ and $b = q$. Output $\sigma = (a, b)$

Ted's super awesome signature scheme

- KEYGEN: Pick two large primes p and q. Let $N = p \times q$. $vk = N$, $sk = (p, q)$
- SIGN: Interpret M as an integer. Let $a = M \times p$ and $b = q$. Output $\sigma = (a, b)$
- VRFY: Parse σ as a, b. Output ACCEPT if and only if $a \times b = vk \times M$

Security definition needs to answer:

Security definition needs to answer:

- What we are trying to prevent happen

Security definition needs to answer:

- What we are trying to prevent happen
- What computational resources the adversary has

Security definition needs to answer:

- What we are trying to prevent happen
- What computational resources the adversary has
- What information they have access to

Security definition needs to answer:

- What we are trying to prevent happen
- What computational resources the adversary has
- What information they have access to
- How they access that information

Definition

Existential Unforgeability under chosen message attack (eu-cma)

Definition

Existential Unforgeability under chosen message attack (eu-cma)

Consider the following game

Definition

Existential Unforgeability under chosen message attack (eu-cma)

Consider the following game

- We run KEYGEN and send the verification key vk to the forger

Definition

Existential Unforgeability under chosen message attack (eu-cma)

Consider the following game

- We run KEYGEN and send the verification key vk to the forger
- The forger sends us a message M_1

Definition

Existential Unforgeability under chosen message attack (eu-cma)

Consider the following game

- We run KEYGEN and send the verification key vk to the forger
- The forger sends us a message M_1
- We must send the forger a signature σ_1, for which $\text{VRFY}(M, \sigma, vk) \rightarrow \text{ACCEPT}$

Definition

Existential Unforgeability under chosen message attack (eu-cma)

Consider the following game

- We run KEYGEN and send the verification key vk to the forger
- The forger sends us a message M_1
- We must send the forger a signature σ_1, for which $\text{VRFY}(M, \sigma, vk) \rightarrow \text{ACCEPT}$
- Repeat this process as much as the forger likes, as long as we are still poly time

Definition

Existential Unforgeability under chosen message attack (eu-cma)

Consider the following game

- We run KEYGEN and send the verification key vk to the forger
- The forger sends us a message M_1
- We must send the forger a signature σ_1, for which $\text{VRFY}(M, \sigma, vk) \rightarrow \text{ACCEPT}$
- Repeat this process as much as the forger likes, as long as we are still poly time
- The forger submits a forgery, (M^*, σ^*), with $M^* \neq M_i$ for any i

Definition

Existential Unforgeability under chosen message attack (eu-cma)

Consider the following game

- We run KEYGEN and send the verification key vk to the forger
- The forger sends us a message M_1
- We must send the forger a signature σ_1, for which $\text{VRFY}(M, \sigma, vk) \rightarrow \text{ACCEPT}$
- Repeat this process as much as the forger likes, as long as we are still poly time
- The forger submits a forgery, (M^*, σ^*), with $M^* \neq M_i$ for any i

The forger is said to have 'won' the game if $\text{VRFY}(M^*, \sigma^*, vk) \rightarrow \text{ACCEPT}$.

Definition

Existential Unforgeability under chosen message attack (eu-cma)

Consider the following game

- We run KEYGEN and send the verification key vk to the forger
- The forger sends us a message M_1
- We must send the forger a signature σ_1, for which $\text{VRFY}(M, \sigma, vk) \rightarrow \text{ACCEPT}$
- Repeat this process as much as the forger likes, as long as we are still poly time
- The forger submits a forgery, (M^*, σ^*), with $M^* \neq M_i$ for any i

The forger is said to have 'won' the game if $\text{VRFY}(M^*, \sigma^*, vk) \rightarrow \text{ACCEPT}$.

If the probability of the forger winning the game is negligible, the signature scheme is said to be existentially unforgeable under chosen message attack.

$$f : \mathcal{X} \to \mathcal{X}$$

$f : \mathcal{X} \to \mathcal{X}$

Inverting f problem: Given $t \in \mathcal{X}$, find $w \in \mathcal{X}$ such that $f(w) = t$.

Assumption: the inverting f problem can't be solved in polynomial time.

$f : \mathcal{X} \to \mathcal{X}$

Inverting f problem: Given $t \in \mathcal{X}$, find $w \in \mathcal{X}$ such that $f(w) = t$.

Assumption: the inverting f problem can't be solved in polynomial time.

There is some trapdoor information td and an algorithm f^{-1} such that if $w \leftarrow f^{-1}(t, td)$ then $f(w) = t$.

Trapdoor signatures - V1.0

- KEYGEN: Output $vk = f$, $sk = td$

Trapdoor signatures - V1.0

- KEYGEN: Output $vk = f$, $sk = td$
- SIGN: Interpret M as an element of $\mathcal{X}$. Using td, find $\sigma \in \mathcal{X}$ such that $f(\sigma) = M$. Output σ.

Trapdoor signatures - V1.0

- KEYGEN: Output $vk = f$, $sk = td$
- SIGN: Interpret M as an element of $\mathcal{X}$. Using td, find $\sigma \in \mathcal{X}$ such that $f(\sigma) = M$. Output σ.
- VRFY: Output ACCEPT if and only if $f(\sigma) = M$

Trapdoor signatures - V1.0

- KEYGEN: Output $vk = f$, $sk = td$
- SIGN: Interpret M as an element of $\mathcal{X}$. Using td, find $\sigma \in \mathcal{X}$ such that $f(\sigma) = M$. Output σ.
- VRFY: Output ACCEPT if and only if $f(\sigma) = M$

To break V1.0: choose $\sigma^* \in \mathcal{X}$. Let $M^* = f(\sigma^*)$. Output (M^*, σ^*) as a forgery.

Hash Functions

A Hash $H : \mathcal{M} \to \mathcal{X}$ function is a multi-purpose cryptographic function that satisfies several important properties.

Hash Functions

A Hash $H : \mathcal{M} \to \mathcal{X}$ function is a multi-purpose cryptographic function that satisfies several important properties.

- Preimage resistance: given $r \in \mathcal{X}$, hard to find $M \in \mathcal{M}$ such that $H(M) = r$

Hash Functions

A Hash $H : \mathcal{M} \to \mathcal{X}$ function is a multi-purpose cryptographic function that satisfies several important properties.

- Preimage resistance: given $r \in \mathcal{X}$, hard to find $M \in \mathcal{M}$ such that $H(M) = r$
- Collision resistance: Hard to find $M, M' \in \mathcal{M}$ such that $H(M) = H(M')$.

Hash Functions

A Hash $H : \mathcal{M} \to \mathcal{X}$ function is a multi-purpose cryptographic function that satisfies several important properties.

- Preimage resistance: given $r \in \mathcal{X}$, hard to find $M \in \mathcal{M}$ such that $H(M) = r$
- Collision resistance: Hard to find $M, M' \in \mathcal{M}$ such that $H(M) = H(M')$.
- Second Preimage resistance: given $M \in \mathcal{M}$, hard to find M' such that $H(M') = H(M)$.

Trapdoor signatures - V2.0

- KEYGEN: Output $vk = f$, $sk = td$
- SIGN: Interpret $H(M)$ as an element of $\mathcal{X}$. Using td, find $\sigma \in \mathcal{X}$ such that $f(\sigma) = H(M)$. Output σ.
- VRFY: Output ACCEPT if and only if $f(\sigma) = H(M)$

Trapdoor signatures - V2.0

- KEYGEN: Output $vk = f$, $sk = td$
- SIGN: Interpret $H(M)$ as an element of $\mathcal{X}$. Using td, find $\sigma \in \mathcal{X}$ such that $f(\sigma) = H(M)$. Output σ.
- VRFY: Output ACCEPT if and only if $f(\sigma) = H(M)$

Previous break no longer works - can't invert H to find the corresponding message M for a chosen signature σ.

- Assume that V2.0 is not existentially unforgeable

- Assume that V2.0 is not existentially unforgeable
- Show that this means there is a way to invert f, without the trapdoor information

- Assume that V2.0 is not existentially unforgeable
- Show that this means there is a way to invert f, without the trapdoor information

Task: Play a game of existential unforgeability with a forger. Use the forger's win to invert f without the trapdoor information

- Assume that V2.0 is not existentially unforgeable
- Show that this means there is a way to invert f, without the trapdoor information

Task: Play a game of existential unforgeability with a forger. Use the forger's win to invert f without the trapdoor information
Neccesary implication: We won't have the trapdoor information.

- Assume that V2.0 is not existentially unforgeable
- Show that this means there is a way to invert f, without the trapdoor information

Task: Play a game of existential unforgeability with a forger. Use the forger's win to invert f without the trapdoor information
Neccesary implication: We won't have the trapdoor information.
Question: How to sign messages without trapdoor information?

Random Oracle Model

- Hash function are meant to replace truly random functions

Random Oracle Model

- Hash function are meant to replace truly random functions
- What if we replaced the hash function back with a truly random function ('oracle')

Random Oracle Model

- Hash function are meant to replace truly random functions
- What if we replaced the hash function back with a truly random function ('oracle')
- This would establish security at least in some abstract sense

Random Oracle Model

- Hash function are meant to replace truly random functions
- What if we replaced the hash function back with a truly random function ('oracle')
- This would establish security at least in some abstract sense
- Controversial amongst cryptographers

Random Oracle Model

- Forger must query random oracle on message M and get back $H(M)$

Random Oracle Model

- Forger must query random oracle on message M and get back $H(M)$
- Give ourselves control over the random oracle

Random Oracle Model

- Forger must query random oracle on message M and get back $H(M)$
- Give ourselves control over the random oracle
- We can 'run' the random oracle however we want

Random Oracle Model

- Forger must query random oracle on message M and get back $H(M)$
- Give ourselves control over the random oracle
- We can 'run' the random oracle however we want
- Important rule: No matter what we do, the output of the random oracle must still be indistinguishable from truly random

Random Oracle Model

- Forger must query random oracle on message M and get back $H(M)$
- Give ourselves control over the random oracle
- We can 'run' the random oracle however we want
- Important rule: No matter what we do, the output of the random oracle must still be indistinguishable from truly random
- This additional power will allow us to sign messages without trapdoor information

Turning a forger into an inverter

Acting as a random oracle:

Turning a forger into an inverter

Acting as a random oracle:

- When the forger queries a message M, asking for $H(M)$:

Turning a forger into an inverter

Acting as a random oracle:

- When the forger queries a message M, asking for $H(M)$:
- Choose a random $\sigma \in \mathcal{X}$

Turning a forger into an inverter

Acting as a random oracle:

- When the forger queries a message M, asking for $H(M)$:
- Choose a random $\sigma \in \mathcal{X}$
- Return $f(\sigma)$ as $H(M)$

Turning a forger into an inverter

Acting as a random oracle:

- When the forger queries a message M, asking for $H(M)$:
- Choose a random $\sigma \in \mathcal{X}$
- Return $f(\sigma)$ as $H(M)$
- Store $(M, \sigma, f(\sigma))$ in a table for future answers

Turning a forger into an inverter

Acting as a signing oracle without trapdoor info:

Turning a forger into an inverter

Acting as a signing oracle without trapdoor info:

- The forger asks for a signature on a message M

Turning a forger into an inverter

Acting as a signing oracle without trapdoor info:

- The forger asks for a signature on a message M
- Look in the random oracle table for M

Turning a forger into an inverter

Acting as a signing oracle without trapdoor info:

- The forger asks for a signature on a message M
- Look in the random oracle table for M
  - If it's not there, query the random oracle yourself so it is

Turning a forger into an inverter

Acting as a signing oracle without trapdoor info:

- The forger asks for a signature on a message M
- Look in the random oracle table for M
  - If it's not there, query the random oracle yourself so it is
  - Get σ from the table

Turning a forger into an inverter

Acting as a signing oracle without trapdoor info:

- The forger asks for a signature on a message M
- Look in the random oracle table for M
  - If it's not there, query the random oracle yourself so it is
  - Get σ from the table
- Return σ to the forger. By construction of the random oracle, $f(\sigma) = H(M)$

Turning a forger into an inverter

Now capable of signing mesages

Turning a forger into an inverter

Now capable of signing mesages
Need to turn the eventual forgery (M^*, σ^*) that the forger creates into a preimage of a target, $t \in \mathcal{X}$

Turning a forger into an inverter

Now capable of signing mesages
Need to turn the eventual forgery (M^*, σ^*) that the forger creates into a preimage of a target, $t \in \mathcal{X}$
Idea: Imbed the element t into the random oracle

Turning a forger into an inverter

- Assume that q queries are made to the random oracle

Turning a forger into an inverter

- Assume that q queries are made to the random oracle
- Choose a random query that the adversary makes, and rather than responding in the usual way, just respond with t

Turning a forger into an inverter

- Assume that q queries are made to the random oracle
- Choose a random query that the adversary makes, and rather than responding in the usual way, just respond with t
- Hope that this query is the message that the adversary creates a forged signature on

Turning a forger into an inverter

- Assume that q queries are made to the random oracle
- Choose a random query that the adversary makes, and rather than responding in the usual way, just respond with t
- Hope that this query is the message that the adversary creates a forged signature on
- If we guessed correctly, (M^*, σ^*) is the forgery, and $t = H(M^*) = f(\sigma^*)$, so we have inverted f on our target t

Turning a forger into an inverter

- Assume that q queries are made to the random oracle
- Choose a random query that the adversary makes, and rather than responding in the usual way, just respond with t
- Hope that this query is the message that the adversary creates a forged signature on
- If we guessed correctly, (M^*, σ^*) is the forgery, and $t = H(M^*) = f(\sigma^*)$, so we have inverted f on our target t
- Probability of quessing correctly: $1/q$

Time for quantum

Quantum computers - Capable of solving certain problems much more quickly than classical computers

Time for quantum

Quantum computers - Capable of solving certain problems much more quickly than classical computers

- Factoring numbers
- Solve discrete logarithm problem in generic groups
- Can search unsorted information in square-root time

Time for quantum

Quantum computers - Capable of solving certain problems much more quickly than classical computers

- Factoring numbers
- Solve discrete logarithm problem in generic groups
- Can search unsorted information in square-root time

Not all problems solved quickly

Time for quantum

Quantum computers - Capable of solving certain problems much more quickly than classical computers

- Factoring numbers
- Solve discrete logarithm problem in generic groups
- Can search unsorted information in square-root time

Not all problems solved quickly

- Some lattice problems (LWE, SVP, SIVP)
- Isogenies on elliptic curves
- Multivariate polynomials

Time for quantum

Quantum computers - Capable of solving certain problems much more quickly than classical computers

- Factoring numbers
- Solve discrete logarithm problem in generic groups
- Can search unsorted information in square-root time

Not all problems solved quickly

- Some lattice problems (LWE, SVP, SIVP)
- Isogenies on elliptic curves
- Multivariate polynomials

Assume that we still have a trapdoor permutation that is hard for a quantum computer to invert - is our signature scheme still secure?

Quantum Information Processing 101

	Classical	Quantum
Unit	bit - 0 or 1	qubit - $\alpha_0\|0\rangle + \alpha_1\|1\rangle$
Data	bitstring - $\{0,1\}^N$	superposition - $\sum_{x\in\{0,1\}^N} \alpha_x\|x\rangle$
Function	$H : x \mapsto H(x)$	$H : \sum \alpha_x\|x\rangle\|y\rangle \mapsto \sum \alpha_x\|x\rangle\|y \oplus H(x)\rangle$

	Classical	Quantum
Function	$H : x \mapsto H(x)$	$H : \sum \alpha_x \lvert x\rangle \lvert y\rangle \mapsto \sum \alpha_x \lvert x\rangle \lvert y \oplus H(x)\rangle$

	Classical	Quantum
Function	$H : x \mapsto H(x)$	$H : \sum \alpha_x \lvert x\rangle \lvert y\rangle \mapsto \sum \alpha_x \lvert x\rangle \lvert y \oplus H(x)\rangle$

Problem: a forger with a quantum computer could reasonably want to use the hash function in the quantum way, but our way of handling the random oracle assumes the classical way

Add a second random oracle: $O : \{0,1\}^N \to \mathcal{X}$

Add a second random oracle: $O : \{0,1\}^N \to \mathcal{X}$
We can make O and f quantum accessible much in the same way H can be quantum accessible
Rather than randomly choosing σ on each oracle query, let $\sigma = O(M)$. Then respond to random oracle queries with $f(O(M))$

Add a second random oracle: $O : \{0,1\}^N \to \mathcal{X}$
We can make O and f quantum accessible much in the same way H can be quantum accessible
Rather than randomly choosing σ on each oracle query, let $\sigma = O(M)$. Then respond to random oracle queries with $f(O(M))$

$$\sum_{M \in \{0,1\}^N} \alpha_M |M\rangle|0\rangle \mapsto \sum_{M \in \{0,1\}^N} \alpha_M |M\rangle|f(O(M))\rangle$$

To respond to a signing query M, simply respond with $O(M)$. Then by construction $f(O(M)) = H(M)$.

Choose a random subset $\mathcal{M} \subset \{0,1\}^N$, $|\mathcal{M}| = 2^N/q$

Choose a random subset $\mathcal{M} \subset \{0,1\}^N$, $|\mathcal{M}| = 2^N/q$
Construct the quantum random oracle so that responses to any $M \in \mathcal{M}$ is just t, the element of $\mathcal{X}$ we are trying to invert

Choose a random subset $\mathcal{M} \subset \{0,1\}^N$, $|\mathcal{M}| = 2^N/q$
Construct the quantum random oracle so that responses to any $M \in \mathcal{M}$ is just t, the element of $\mathcal{X}$ we are trying to invert

$$\sum_{M \in \{0,1\}^N} \alpha_M |M\rangle|0\rangle \mapsto \sum_{M \in \{0,1\}^N} \alpha_M |M\rangle|H(M)\rangle$$

$$H(M) := \begin{cases} t & \text{if } M \in \mathcal{M} \\ f(O(M)) & \text{otherwise} \end{cases}$$

Choose a random subset $\mathcal{M} \subset \{0,1\}^N$, $|\mathcal{M}| = 2^N/q$
Construct the quantum random oracle so that responses to any $M \in \mathcal{M}$ is just t, the element of $\mathcal{X}$ we are trying to invert

$$\sum_{M \in \{0,1\}^N} \alpha_M |M\rangle|0\rangle \mapsto \sum_{M \in \{0,1\}^N} \alpha_M |M\rangle|H(M)\rangle$$

$$H(M) := \begin{cases} t & \text{if } M \in \mathcal{M} \\ f(O(M)) & \text{otherwise} \end{cases}$$

Again, hope that the forgery the adversary submits is on an M^* such that $H(M^*) = t$.

Questions?