COMMITMENTS and the RANDOM ORACLE MODEL

Outline:

- Commitments
  - Motivation
  - Definition
  - Example: Pedersen Commitments
- Random Oracle Model
  - Simple Commitment Scheme from Hash Function
  - Intuition
  - Formalization
  - Example: Simple PRF

Commitment Scheme Motivating Example:

- Alice and Bob want to "flip a coin" over the phone

Attempt #1:

Alice		Bob
1. Alice flips coin and gets b		
	2. Alice sends b to Bob $\longrightarrow$	
Alice outputs b		Bob outputs b

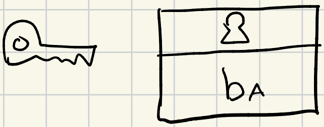
Problem: What if Bob doesn't trust Alice?

Attempt #2:

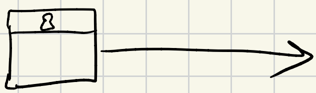
Alice | Bob

1. Alice flips coin and gets b_A

2. Alice locks b_A in box



3. Alice sends locked box to Bob

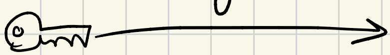


4. Bob flips coin and gets b_B

5. Bob sends b_B to Alice

$\longleftarrow$

6. Alice sends key to Bob



Alice outputs $b_A \oplus b_B$ | Bob outputs $b_A \oplus b_B$

Claim: $b_A \oplus b_B$ is random if at least 1 party is honest

Why?

1) If Alice honest, b_A is uniformly random. Bob knows nothing about b_A when choosing $b_B \Rightarrow b_A$ and b_B are independent $\Rightarrow b_A \oplus b_B$ is also uniformly random

2) If Bob honest, b_B is uniformly random. Alice can't change $b_A \Rightarrow b_A$ and b_B are independent $\Rightarrow b_A \oplus b_B$ uniformly random

Commitment Schemes

Intuition: cryptographic opaque locked box

Formally: a pair of algorithms $\Pi = (\text{Setup}, \text{Commit})$

- $\text{Setup}(1^\lambda) \to pp$
  takes security parameter as input and outputs public parameters
- $\text{Commit}(pp, m, r) \to c$
  takes public parameters as input, a message m from the message space $\mathcal{M}$, and randomness r from randomness space $\mathcal{R}$ and outputs a commitment in commitment space $\mathcal{C}$

Two Key Properties:

Hiding: commitments hide the message
"you can't see through the box"

$\forall m, m' \in \mathcal{M}$

$$\{\text{Commit}(m,r) : r \xleftarrow{\$} \mathcal{R}\} \approx \{\text{Commit}(m',r) : r \xleftarrow{\$} \mathcal{R}\}$$

- perfect: distributions same
- statistical: statistical distance negligible
- computational: no efficient distinguisher

Binding: no efficient adversary can produce m, m', r, r' such that $\text{Commit}(m,r) = \text{Commit}(m',r')$
"only one value locked in the box"

$\forall$ PPT $\mathcal{A}$

$$\Pr\left[\begin{array}{l}\text{Commit}(m,r) = \text{Commit}(m',r') \\ \wedge\ (m,r) \neq (m',r')\end{array} : (m,r,m',r') \leftarrow \mathcal{A}(pp)\right] \leq \text{negl}(\lambda)$$

computational

Pedersen Commitments

Construction

G: a group of prime order p generated by g

We assume the discrete log problem is hard in G:
↳ given (G, p, g, g^x) for a uniformly random $x \xleftarrow{\$} \mathbb{Z}_p$, it is infeasible to compute x

Discrete-Log Security Game (formal)

Adversary A		Challenger
		$x \xleftarrow{\$} \mathbb{Z}_p$
		$h \leftarrow g^x$
	$\xleftarrow{\quad g, h \quad}$	
	$\xrightarrow{\quad x' \quad}$	

↳ A wins when $x = x'$

D-log assumption: all PPT adversaries win w/ only negligible probability

$$\mathcal{M} = \mathbb{Z}_p, \quad \mathcal{R} = \mathbb{Z}_p, \quad \mathcal{C} = G$$

Commitment Scheme:

Setup(1^λ):
- sample $h \xleftarrow{\$} G$
- output (G, p, g, h)

Commit$((G, p, g, h), m)$:
- $r \xleftarrow{\$} \mathbb{Z}_p$
- output $g^m h^r$

Analysis

1) Perfectly Hiding

Pf. For any $m \in \mathcal{M}$, consider the distribution

$$\{\text{Commit}(m,r) : r \xleftarrow{\$} R\} = \{g^m h^r : r \xleftarrow{\$} R\}$$

r is unif. rand $\Rightarrow h^r$ is unif rand $\Rightarrow g^m h^r$ is unif. rand.

$\Rightarrow$ distribution is independent of m

2) Computationally Binding

To prove binding, we show d-log hardness $\Rightarrow$ binding.

Pf. We assume we have an adversary $\mathcal{A}$ that can break binding of Pedersen w/ non-negl probability ρ. We then show that we can use $\mathcal{A}$ to build adversary $\mathcal{B}$ that wins D-log game. $\mathcal{B}$ simultaneously plays the role of adversary in the D-log game and challenger in the Pedersen binding game

Life Advice: to break d-log, get two different representations of a group element

For example:

$$g^m h^r = g^{m'} h^{r'} \Rightarrow g^m (g^x)^r = g^{m'} (g^x)^{r'} \Rightarrow g^{m+xr} = g^{m'+xr'}$$

$$\Rightarrow m + xr = m' + xr' \Rightarrow x = \frac{m-m'}{r'-r}$$

Pedersen Binding Security Game Pf.

$\mathcal{A}$ (Binding adversary) | $\mathcal{B}$ | D-log Challenger

D-log Challenger: $x \xleftarrow{\$} \mathbb{Z}_q$, $h \leftarrow g^x$

D-log Challenger $\to \mathcal{B}$: g, h

$\mathcal{B} \to \mathcal{A}$: G, p, g, h

Magic!

$\mathcal{A} \to \mathcal{B}$: m, m', r, r'

$\mathcal{B}$: $g^m h^r = g^{m'} h^{r'}$, $m \neq m'$

use life advice!

$$x' = \frac{m-m'}{r'-r}$$

$\mathcal{B} \to$ D-log Challenger: x'

$\Pr[x' = x] = \rho$ which is not negligible!

Pedersen commitments are homomorphic

$$\begin{aligned} \mathrm{Commit}(m,r)\cdot\mathrm{Commit}(m',r') &= g^m h^r g^{m'} h^{r'} \\ &= g^{m+m'} h^{r+r'} \\ &= \mathrm{Commit}(m+m', r+r') \end{aligned}$$

... but are there simpler commitment schemes??

YES!!

Hash-based commitments!

... but how to prove security?

We need the RANDOM ORACLE MODEL

Random Oracle Model *controversial

Reasoning about security of hash functions is hard ⇒ instead of reasoning about hash directly, treat hash function H as a random function

$H: \mathcal{X} \to \mathcal{Y}$ defined by $H(x) \mapsto$ a random element of $\mathcal{Y}$

→ agrees w/ common intuition for hash functions

→ pervasive in cryptographic implementations

Consider simple commitment with hash H modeled as random function:

$\text{Commit}(m, r) := H(m, r)$

- hiding: H's output is uniformly random
- binding: breaking binding requires finding $(m, r) \neq (m', r')$ such that $H(m, r) = H(m', r')$, a collision, which is hard for a random function

Q: Is $H(m)$ a commitment? Or g^m?

NO! m may have insufficient entropy

Elegant but some questions:

* why a "model" not an assumption?
* how can we formalize this?

Why a model?

Cryptography is (epistemologically) part of mathematics. We model the world and prove theorems within the model.

↳ Our proofs so far have been in the standard model
- weak assumptions

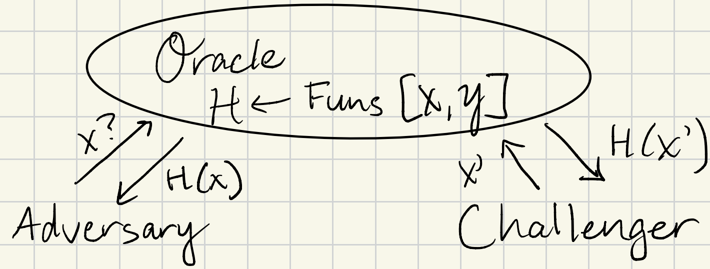
↳ Now we'll see the random oracle model
- a stronger assumption: "all parties have oracle access to H, a random function, sampled at start-up"

Weakness: in our implementations, we do not sample H ⇒ the model is a <u>LIE</u>?

"all models are wrong; some are useful"

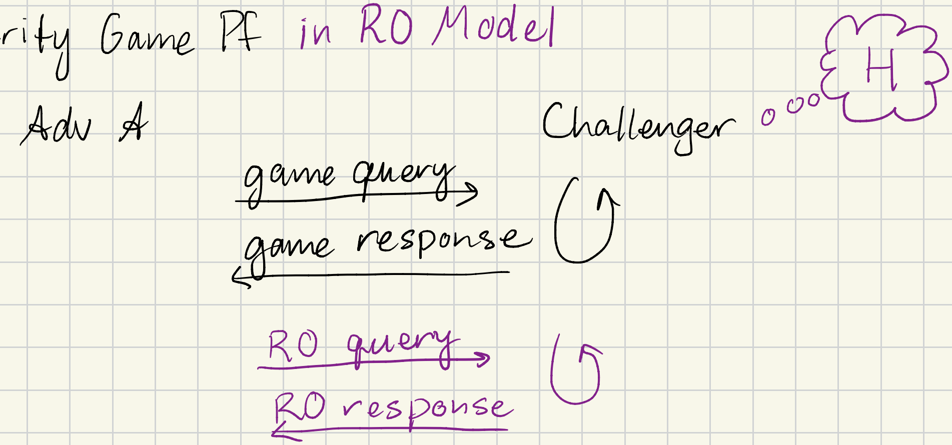
How to formalize?

- let $H: \mathcal{X} \to \mathcal{Y}$ be a function (i.e. the random oracle)
- all parties have access to an oracle that samples H



↳ in security game proof, adversary sends RO queries to challenger

Security Game Pf in RO Model



↳ challenger's responses must be ≈ to a random function

↳ Ex: for each adversary query $H(m)$, C sets $H(m) \xleftarrow{\$} \mathcal{Y}$ and remembers previous answers

PRF Proof in RO Model

PRF Security Game for $F: \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$ that uses RO H

Adversary $\mathcal{A}$ — Challenger(b)

Challenger: if $b=0$, $k \xleftarrow{\$} \mathcal{K}$, $f \leftarrow F(k,\cdot)$; else $f \xleftarrow{\$} \text{Funs}[\mathcal{M}, \mathcal{Y}]$

$\mathcal{A} \to$ Challenger: $f @ m?$

Challenger $\to \mathcal{A}$: $f(m)$

$\mathcal{A} \to$ Challenger: $H @ m?$

Challenger $\to \mathcal{A}$: $H(m)$

$\mathcal{A} \to$ Challenger: b'

Let W_b be the event that $\mathcal{A}$ outputs 1 when $b=0,1$. $\mathcal{A}$'s advantage with respect to F is $\text{PRFadv}[\mathcal{A}, F] := |\Pr[W_0] - \Pr[W_1]|$. A PRF F is secure if for all efficient adversaries, $\text{PRFadv}[\mathcal{A}, F]$ is negligible

Now let's use this definition to prove security of a specific hash-based PRF!

The PRF: $f(k,x) = H(x)^k$, $H: \mathcal{X} \to G$

Claim: Secure in RO model assuming DDH

Decisional Diffie-Helman (DDH) Assumption:

for group of order q with generator g:

$$\{(g^x, g^y, g^{xy}) : x,y \xleftarrow{\$} \mathbb{Z}_q\} \approx_c \{(g^x, g^y, g^z) : x,y,z \xleftarrow{\$} \mathbb{Z}\}$$

"DDH triple" "random triple"

DDH Security Game

Adversary $\mathcal{B}$ | **Challenger (b)**

Challenger:
$x, y, z \xleftarrow{\$} \mathbb{Z}_q$
if $b=0$, $X = g^x$, $Y = g^y$, $Z = g^{xy}$
else $X = g^x$, $Y = g^y$, $Z = g^z$

Challenger $\xrightarrow{X, Y, Z}$ Adversary $\mathcal{B}$

Adversary $\mathcal{B}$ $\xrightarrow{\hat{b}}$ Challenger

Let W_b be the event $\mathcal{B}$ outputs 1 for $b = 0, 1$. $\mathcal{B}$'s advantage in solving DDH for G is $\text{DDHadv}[\mathcal{B}, G] := |\Pr[W_0] - \Pr[W_1]|$

↳ we say the DDH assumption holds for G if for all efficient adversaries $\mathcal{B}$ the quantity $\text{DDHadv}[\mathcal{B}, G]$ is negligible

We will prove that if there exists an adversary $\mathcal{A}$ who breaks our PRF, then we can build an adversary $\mathcal{B}$ who uses $\mathcal{A}$ to break DDH. Just as before, $\mathcal{B}$ simultaneously plays the role of challenger in the PRF security game and adversary in the DDH game.

$\mathcal{A}$ (PRF adversary) | **$\mathcal{B}^{RO}$** | **DDH Challenger (b)**

DDH Challenger:
$x, y, z \xleftarrow{\$} \mathbb{Z}_q$
if $b=0$, $X = g^x$, $Y = g^y$, $Z = g^{xy}$
else: $X = g^x$, $Y = g^y$, $Z = g^z$

DDH Challenger $\xrightarrow{X, Y, Z}$ $\mathcal{B}$

$\mathcal{A}$ $\xrightarrow{f \,@\, m?}$ $\mathcal{B}$

$\mathcal{B}$ $\xrightarrow{f(m)}$ $\mathcal{A}$

$\mathcal{A}$ $\xrightarrow{H \,@\, m?}$ $\mathcal{B}$

$\mathcal{B}$ $\xrightarrow{H(m)}$ $\mathcal{A}$

Magically determine if interacting with PRF or random function

$\mathcal{A}$ $\xrightarrow{b'}$ $\mathcal{B}$ $\xrightarrow{b'}$ DDH Challenger

* B has to convince A it is interacting with a "real" PRF challenger ⟶ its query answers must be indistinguishable from answers from "real" PRF challenger.

Trick: B doesn't use RO - it just pretends to!

DDH Adversary $B(pp, X, Y, Z)$

↳ B maintains map H to keep track of simulated RO values for consistency

↳ To answer H @ m queries:
- If $m \notin H$:
  - $\alpha \xleftarrow{\$} \mathbb{Z}_q$
  - Set $H(m) \leftarrow (X^\alpha, \alpha)$ ← indistinguishable from random
- Send $H(m)[0]$

↳ To answer f @ m queries:
- If $m \notin H$:
  - $\alpha \xleftarrow{\$} \mathbb{Z}_q$
  - Set $H(m) \leftarrow (X^\alpha, \alpha)$
- $\alpha \leftarrow H(m)[1]$
- Send Z^α ← if b=1, then this is PRF where y is secret key!

Note:

if $b=0$, $Z^\alpha = g^{xy\alpha} = g^{x\alpha y} = X^{\alpha y} = H(m)^y$ ← the PRF!

if $b=1$, $Z^\alpha = g^{z\alpha} = (g^\alpha)^z$ ← indistinguishable from random!

Ergo, guessing PRF vs. random is equivalent to guessing DDH triple vs. random triple!

⇒ B and A have the same advantage!

⇒ if A can break PRF security, then B can break DDH assumption

Thoughts and Comments on RO Model

- a heuristic model that seems to work well in reality and gives simpler/faster schemes than we have in the standard model

- in applications, we replace RO with a specific hash function → pretty pervasive in implemented crypto

- there are some (contrived) schemes that are secure in RO model but insecure in standard model no matter what hash function is used

- some people especially don't like this "dirty trick" of programming an RO - how is it connected to reality?!

- we (at Stanford) tend to be RO-friendly :)

On Instantiation

- do not use Merkle-Damgard hash like SHA256
- SHA3 (sponge-based) or
- SHA 2, carefully padded