Commitments and the Random Oracle Model (!!)

Outline

- Commitments
  - Definition
  - Pedersen Construction
- The Random Oracle Model ← the most controversial topic in crypto
  - Intuition
  - Simple Applications
  - Formalization
  - A simple (yet useful) PRF

Commitments

A deterministic algorithm:

- $\text{Commit}(m, r) \to c$

  $m \in \mathcal{M}$, $r \in \mathcal{R}$, $c \in \mathcal{C}$

  messages ↑, ↑ randomness, ↖ commitments

Use:

Committer		Committee
$r \xleftarrow{\$} \mathcal{R}$		
$c \leftarrow \text{Commit}(m, r)$		
	$\xrightarrow{\quad c \quad}$	
	(things happen)	
	$\xrightarrow{\quad (m, r) \quad}$	$c \stackrel{?}{=} \text{Commit}(m, r)$

Properties:

Hiding: Commitments do not reveal their message

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

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

↑ perfect, statistical, or computational

Binding: One cannot open a commitment to a different message

No efficient adversary can produce m, m', r, r' such that $\text{Commit}(m, r) = \text{Commit}(m', r')$

Formally: $\forall$ PPT $\mathcal{A}$,

(computational)

$$\Pr\left[\begin{array}{c}(m, r, m', r') \leftarrow \mathcal{A}(g, h); \\ m \neq m' \ \wedge \\ \text{Commit}(m, r) = \text{Commit}(m', r')\end{array}\right] = \text{negl}(\lambda)$$

Not a commitment: AES.Encrypt($k = r$, $m = m$)

(because (r', AES.Decrypt($k = r'$, $c = c$)) is an alternate opening)

Pedersen Commitments

Construction:

G a group of prime order p.

g, h, generators of G whose d-log is unknown.

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

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

Analysis

Perfectly Hiding

Proof: 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 uniform $\Rightarrow$ h^r is uniform $\Rightarrow$ $g^m h^r$ is uniform $\Rightarrow$ distribution is independent of m.

Computationally binding:

Assumes d-log is hard "given $h \in G$, hard to find $x \in \mathbb{Z}_p$ such that $g^x = h$"

d-log security game

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

Adversary wins when $x' = x$.

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

Proof Advice: To break d-log, get two different representations of a group element. For example:

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

Proof that d-log hardness $\Rightarrow$ pedersen binding:

Suppose that A breaks pedersen binding with non-negligible probability, ρ

D-log Challenger		D-log Adversary		A
$x \xleftarrow{\$} \mathbb{Z}_p$				
$h \leftarrow g^x$	$\xrightarrow{g,\, h}$			
			$\xrightarrow{g,\, h}$	
			$\xleftarrow{m,\, r,\, m',\, r'}$	
		$g^m h^r = g^{m'} h^{r'}$, $m \neq m'$		
		use		
	$\xleftarrow{x'}$			

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

Pedersen commitments are Homomorphic:

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

Useful relationship (see more in week 6)

What if homomorphism is not needed? Are there simpler commitments.

Yes... in the **Random Oracle Model!!!**

Random Oracle Model

Treat your hash function H as a random function.

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

agrees with common intuition for hash functions. Pervasive in real cyptographic implementations

Before the details, simple applications:

1. Simple commitments
   $\text{Commit}(m, r) := H(m, r)$
   Hiding because H's output is uniformly random.
   Binding because 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.
2. Simple PRFs.
   $f(k, x) = H(k, x)$
   Secure — since H is random, $H(k, \cdot)$ is random for all k.
   Would be used as a PRF, if hash functions were faster than AES.

Elegant constructions! What's the catch?

Key Questions:

- How can we formalize this?
- Why a "model" not an "assumption"?

Why a "model":

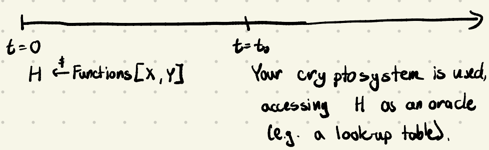
Observation: Cryptography is (epistemologically) part of mathematics. We model the world, and prove theorems within the model.

Our proofs so far have been in the standard model. ← inaccurate, but usually close enough

- weak assumptions, e.g. "programs have private memory" see "whitebox crypto"

Now we'll try out the random oracle model

- a stronger assumption: "all parties have access to H, a random function, sampled at start-up."

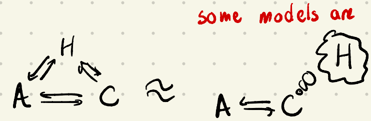


Weakness: In our implementation, we do not sample H. The model is a LIE!

How to formalize?:

"all models are wrong, some models are useful"

- Let $H: X \to Y$ be a function (the random oracle)
- ~~All parties get oracle access to H~~
  - Adversary sends H queries to the challenger!
  - Challenger's responses must be $\approx$ to a random function.
    - one example: for each query $H(m)$, C sets $H(m) \xleftarrow{\$} Y$ (remembering previous answers)



A PRF security proof in the RO model

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

Broadly useful!
key-homomorphic
an "oblivious" PRF
BLS signatures
} see HW

Secure in the RO model, assuming DDH.

Decisional Diffie-Hellman (DDH) Assumption:
for G of order q, with generator g,

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

Assuming and adversary A for our PRF, we'll build an adversary B for DDH.

A (PRF Adv.) | B^{RO} | DDH Challenger (b)

$A \to B$: f @ m

$B \to A$

$A \to B$: H @ m

$B \to A$

B: ?

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

DDH Challenger $\to B$

$b'=1$: random Fn
$b'=0$: PRF

$A \to B$: b'

$B \to$ DDH Challenger: b'

So, what does B^{RO} do? It imitates a random oracle.

B^{RO}

maintains a map for H

f @ m queries
$\alpha \xleftarrow{\$} \mathbb{Z}_q$
set $H(m) \leftarrow X^\alpha$
send Z^α ← if b=1, y is now the secret key

H @ m queries
$\alpha \xleftarrow{\$} \mathbb{Z}_q$.
set $H(m) = X^\alpha$
send X^α ← programming the RO!
- (indistinguishable from random, useful for proof)

Observe:
if $b=1$, $Z^\alpha = g^{xy\alpha} = g^{x\alpha y} = X^{\alpha y} = H(m)^y$ ← the PRF
if $b=0$, $Z^\alpha = g^{z\alpha} = (g^\alpha)^z$ ← uniformly random.
ergo, guessing PRF vs random is equivalent to guessing DDH triple v. random triple.
$\Rightarrow$ B and A have same advantage.

Philosophical Reflections on ROs...

- A model: heuristic but useful
- focusses us on design considerations other than hashing. ← a decision about priorities...
- controversial, but pervasive in implemented crypto
- something that we (Stanford cryptographers) like

On Instantion

- Do not use a Merkle-Damguard hash like SHA256 (length extension)
- SHA 3 (sponge-based) or
- SHA 2, carefully padded