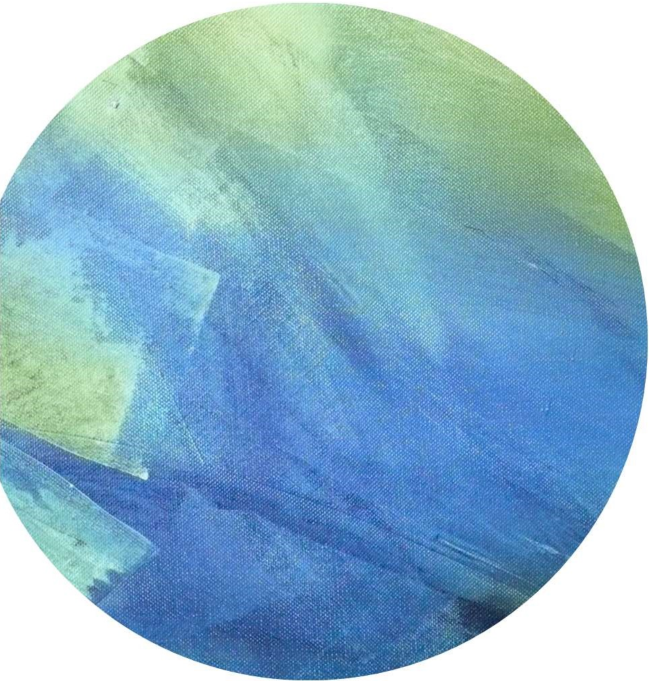
Lecture 17



Outline

OR Composition, continued

Pairing-based cryptography

OR Composition, Continued

Encrypting bits

$\checkmark\ (g, h, ct)$ is a DH tuple

OR $(g, h, ct = (c_1, c_2))$ when modified $\left(g, h, c_1, \frac{c_2}{g}\right)$ is a DH tuple

$c_1 = g^c$, $c_2 = h^c \cdot g$

Prover **Verifier**

NP language = { (g,h,v,w) : ∃ (b,c) such that b \in {0,1} and v = g^c, w = $h^c.g^b$}

Enc : $pp = (g, h)$ $h = g^\alpha$

$ct = Enc(b, (g, h), c) \rightarrow \underline{(h^c \cdot g^{b}, g^c)}$

p.t. $ct = Enc(0, \ldots)$ or $Enc(1, \ldots)$

$g, h, (h^c \cdot g^0 = h^c) = \underline{(g, g^\alpha, g^c, g^{\alpha c + 1})}$

OR Composition

Prover **Verifier**

Goal: Prove that $\exists$ w such that (x_0,w) in R_0 or (x_1,w) in R_1

Suppose we have a protocol (P_0, V_0) for R_0, and a protocol (P_1, V_1) for R_1

Can we combine them to obtain a protocol for R_0 OR R_1?

What about letting the prover simulate exactly one of them?

OR Composition

Prover **Verifier**

Goal: Prove that $\exists$ w such that (x_0, w) in R_0 or $(x_1, \cancel{w})$ in R_1

TRUE

FALSE

statement 0

statement 1

Prove: $\boxed{}$

Sim: guess b_1

output $\boxed{}$

b_0

b_1

c

V checks each proof

also checks that $c = b_0 \oplus b_1$

OR Composition

~~**Prover**~~ Simulator **Verifier**

Goal: Prove that w such that (x_0,w) in R_0 or (x_1,w) in R_1

Guess (b_0, b_1)

Sim(b_0) | Sim(b_1)

c

If $c = b_0 \oplus b_1$

Application: Encrypting bits

Prover **Verifier**

NP language = { (g,h,v,w) : ∃ (b,c) such that b \in {0,1} and v = g^c, w = $h^c.g^b$}

(g, h, u, v) is a DH tuple OR $(g, h, u, \frac{v}{g})$ is a DH tuple

OR $(g, h, u, \frac{v}{g^2})$ is DH tuple

Non-Interactive Zero-Knowledge

Prover

Verifier

$\boxed{com_1}$

ch_1

$open_1$

$\boxed{com_2}$

ch_2

$\vdots$

n

Non-Interactive Zero-Knowledge: Fiat-Shamir

Prover

Verifier

RO

$com_1, com_2 \ldots com_n \longrightarrow$

$\longleftarrow ch_1, ch_2 \ldots ch_n$

$com_1, com_2 \ldots com_n$

$ch_1 \ldots ch_n$

$open_1, open_2 \ldots open_n$

Non-Interactive Zero-Knowledge: Fiat-Shamir

Prover **Verifier**

~~com_1~~

~~ch_1~~

~~com_2~~

~~ch_2~~

RO

$Sim(b) = com$

$ch = b$

if $ch_1 =$

$RO(\ )$

$Sim(b), b,$ opening

Pairings

Pairing-based cryptography

- So far, we've looked at hard problems like discrete log, CDH, HDH, DDH in groups
- Certain groups have an additional structure

- Let G_0, G_1, G_T be 3 cyclic groups of prime order

  where $g_0 \in G_0$ and $g_1 \in G_1$ are generators

  base groups

  pairing maps

- A pairing is an efficiently computable function e: $G_0 \times G_1 \rightarrow G_T$ such that:

  target group

  1. $g_T = e(g_0, g_1)$ is a generator of G_T
  2. For all $(u, u') \in G_0$ and $(v, v') \in G_1$,

     $e(u.u', v) = e(u,v).e(u',v)$ and $e(u, v.v') = e(u,v).e(u,v')$

$$e(u,v) \cdot e(u',v) = e(u \cdot u', v)$$
$$e(u,v) \cdot e(u,v') = e(u, v.v')$$

Pairing-based cryptography

- A pairing is an efficiently computable function e: $G_0 \times G_1 \rightarrow G_T$ such that:

  1. $g_T = e(g_0, g_1)$ is a generator of G_T
  2. For all $(u, u') \in G_0$ and $(v, v') \in G_1$,

     $e(u.u', v) = e(u,v).e(u',v)$ and $e(u, v.v') = e(u,v).e(u,v')$

- Consequences: $e(g_0^a, g_1^b) = (e(g_0, g_1))^{ab} = e(g_0^b, g_1^a)$

$$e(g_0^2, g_1^b) = e(g_0, g_1^b) \cdot e(g_0, g_1^b)$$

Generalized: $e(g_0^a, g_1^b) = (e(g_0, g_1^b))^a$

Similarly, $= (e(g_0, g_1))^{ab}$

NORMAL GROUPS: $g^a \; g^b$: Hard to compute g^{ab}.

PAIRING GROUPS: $g_0^a \; g_1^b$: Easy to compute g_T^{ab}

Pairing-based cryptography

- A pairing is an efficiently computable function e: $G_0 \times G_1 \rightarrow G_T$ such that:
  1. $g_T = e(g_0, g_1)$ is a generator of G_T
  2. For all $(u, u') \in G_0$ and $(v, v') \in G_1$,

     $e(u.u', v) = e(u,v).e(u',v)$ and $e(u, v.v') = e(u,v).e(u,v')$
- Consequences: when $G_0 = G_1$, then DDH in G_0 is easy.

$\Rightarrow g_0 = g_1$. $(u, v, w) = (g_0^a, g_0^b, g_0^{ab})$

DDH assumption: $(u, v, w) \approx (g_0^a, g_0^b, g_0^c)$ for random c.

Recall: $\boxed{e(g_0^a, g_0^b)} = (e(g_0, g_0))^{ab}$. $\boxed{e(g_0^c, g_0)} = (e(g_0, g_0))^c$

$e: \underline{G_0} \times \underline{G_1} \longrightarrow \underline{G_T}$

A Useful Hardness Assumption

$\boxed{\text{Old CDH: Given } g_0^a, g_0^b, \text{ hard to compute } g_0^{ab} \quad \text{new co-CDH}: (g_1^a)}$

- Co-CDH assumption:
  - Sample random (a, b) in Z_q
  - $u_0 = g_0^a$, $u_1 = g_1^a$, $v_0 = g_0^b$, $\cancel{z_0 = g_0^{ab}}$
  - Send (u_0, u_1, v_0) to $\mathcal{A}$
  - $\mathcal{A}$ outputs $z' \in G_0$
  - $\mathcal{A}$ wins if $z' = \cancel{z_0}\ g_0^{ab}$

$g_0^a, g_0^b \cdot \underline{g_T^{ab}}$: easy

$\underline{g_0^{ab}}$: hard

$pk = g_1^a$,

G_0 G_1 G_T

Warmup: BLS Signatures

$sk = a$. pk g_1^a

Sign $(H(m))^a$

- Constructed as:
  1. KeyGen (1^k): Sample random a, output (sk = a), (pk = g_1^a). $\in \mathbb{Z}_q$
  2. Sign (sk = a, m): $\sigma = (H(m))^a \in G_0$
  3. Verify (pk, m, σ): Output 1 iff e (H(m), pk) = e (σ, g_1). $\overset{\text{iff}}{\Longleftarrow\Rightarrow}$ $\sigma = (H(m))^a$

$$\boxed{e(H(m), pk)} = e(H(m), g_1^a) = (e(H(m), g_1))^a$$

$\in G_0$

$$= e((H(m))^a, g_1) = \boxed{e(\sigma, g_1)}$$

Warmup: BLS Signatures

1. KeyGen (1^k): Sample random a, output (sk = a), (pk = $g_1{}^a$).
2. Sign (sk = a, m): $\sigma = (H(m))^a \in G_0$
3. Verify (pk, m, σ): Output 1 iff e (H(m), pk) = e (σ, g_1).

BLS Signatures - Aggregation

- Constructed as:

1. KeyGen (1^k): Sample random a, output (sk = a), (pk = $g_1^a \in G_1$)
2. Sign (sk = a, m): $\sigma = (H(m))^a \in G_0$
3. Verify (pk, m, σ): Output 1 iff e (H(m), pk) = e (σ, g_1).

4. SignAgg (pk_1, …pk_n, σ_1,… σ_n): $\sigma_{agg} = \sigma_1 . \sigma_2 . \ldots \sigma_n \in G_0$
5. VerifyAgg (pk_1, …pk_n, m_1, …m_n, σ_1,… σ_n):

BLS Signatures - Aggregation

1. SignAgg (pk_1, …pk_n, σ_1,… σ_n): $\sigma_{agg} = \sigma_1 . \sigma_2 . \ldots \sigma_n \in G_0$
2. VerifyAgg (pk_1, …pk_n, m_1, …m_n, σ_1,… σ_n):

Next time:
Zero-Knowledge from Pairings