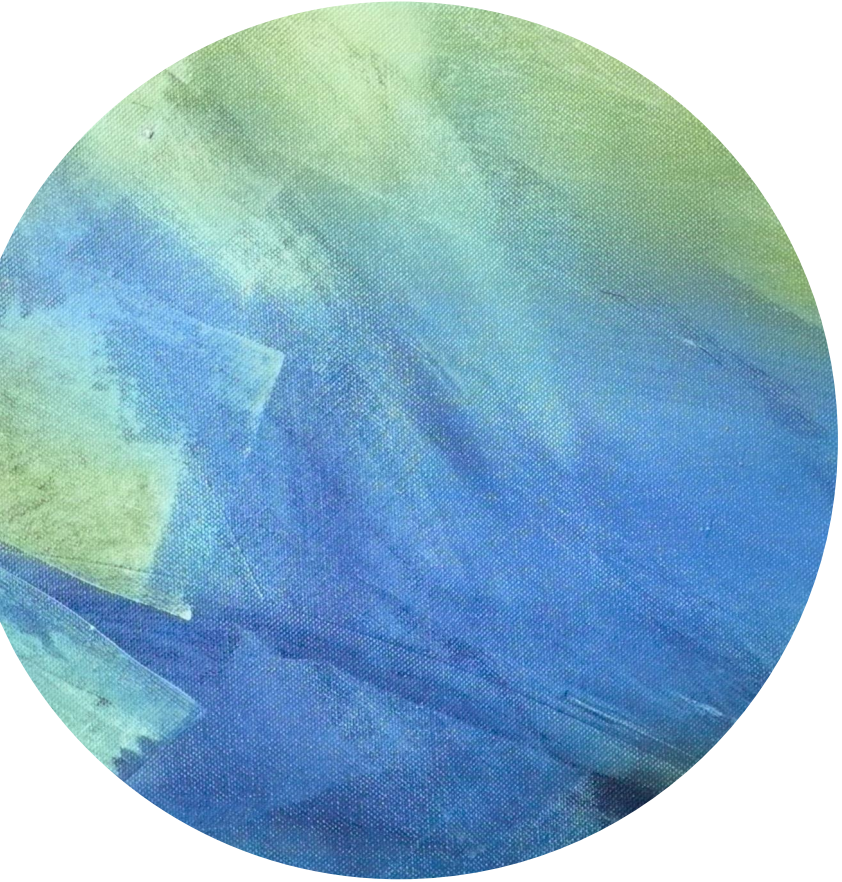
Lecture 22

Rubin



Outline

OR Composition, continued

Pairing-based cryptography

OR Composition, Continued

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, v, w

$= g^a$

g^b

$g^{ab} \cdot g^0$

OR

$g^{ab} \cdot g^1$

(g, h, v, w) is a DDH tuple:

Chaum-Pedersen

$(g, h, v, \frac{w}{g})$ is a DDH tuple

OR Composition

Prover **Verifier**

Goal: Prove that 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?

$ch \leftarrow \mathbb{Z}_p$ in Chaum-Pedersen

OR Composition

Prover | TRUE | FALSE **Verifier**

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

Π_1

First P msg of "$\exists\, w \mid (x_0, w) \in R_0$" $\longrightarrow$

P msg of "$\exists\, w \mid (x_1, w) \in R_1$" $\longrightarrow$ ch, Π_2.

$\longleftarrow$ ch

$|ch| = |ch_0| = |ch_1|$.

ch_0, $\longrightarrow$

ch_1 $\longrightarrow$

check: $ch_0 \oplus ch_1 = ch$

complete protocol$_1$ $\longrightarrow$

complete protocol$_2$ $\longrightarrow$

OR Composition : Soundness.

Prover **Verifier**

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

Before rewinding.

FIRST MESSAGE

ch

ch_0 ch_1

REWIND and send

ch'

ch_0' ch_1'

$ch' \neq ch$

$\Downarrow$

either $ch_0' \neq ch_0$

or $ch_1' \neq ch_1$

OR Composition : Zero - Knowledge.

Prover **Verifier**

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

Guess ch in advance

$\Rightarrow$ Sample random ch_0, ch_1 s.t. $ch_0 \oplus ch_1 = ch$.

Simulate Π_1 given ch_0	Simulate Π_2 given ch_1

OR Composition : EXAMPLE.

Prover **Verifier**

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

FALSE.

$\boxed{g, h, v, w \equiv \text{DDH}}$

$\tilde{P}$ $\boxed{g, h, v, \frac{w}{g} \equiv \text{DDH}}$

$R, T \longrightarrow$

$\tilde{R}, \tilde{T} \longrightarrow$

$\longleftarrow \cancel{P}$

$P' \longleftarrow$

$\longleftarrow \cancel{\tilde{P}}$

$P \longrightarrow$

$\tilde{P} \longrightarrow$

Application: Encrypting bits

Prover **Verifier**

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

Non-Interactive Zero-Knowledge

Prover **Verifier**

"commit" →

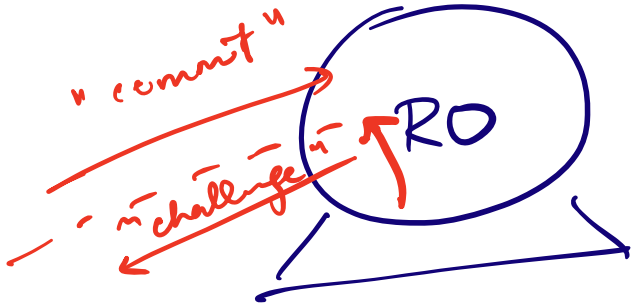
← "challenge" [n bit]

"response" →

SOUNDNESS.

Non-Interactive Zero-Knowledge: Fiat-Shamir

Prover **Verifier**



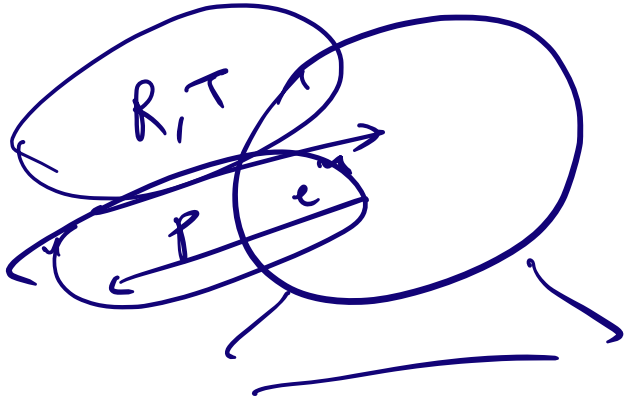
"commit" → "challenge" = H("commit") → "response" →

Non-Interactive Zero-Knowledge: Fiat-Shamir

Prover **Verifier**

Sim fixes "challenge" first, randomly

Then computes "commit" message.



R, T

H(R, T)

S, r + p·b

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

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

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

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 $G_0 = G_1$,