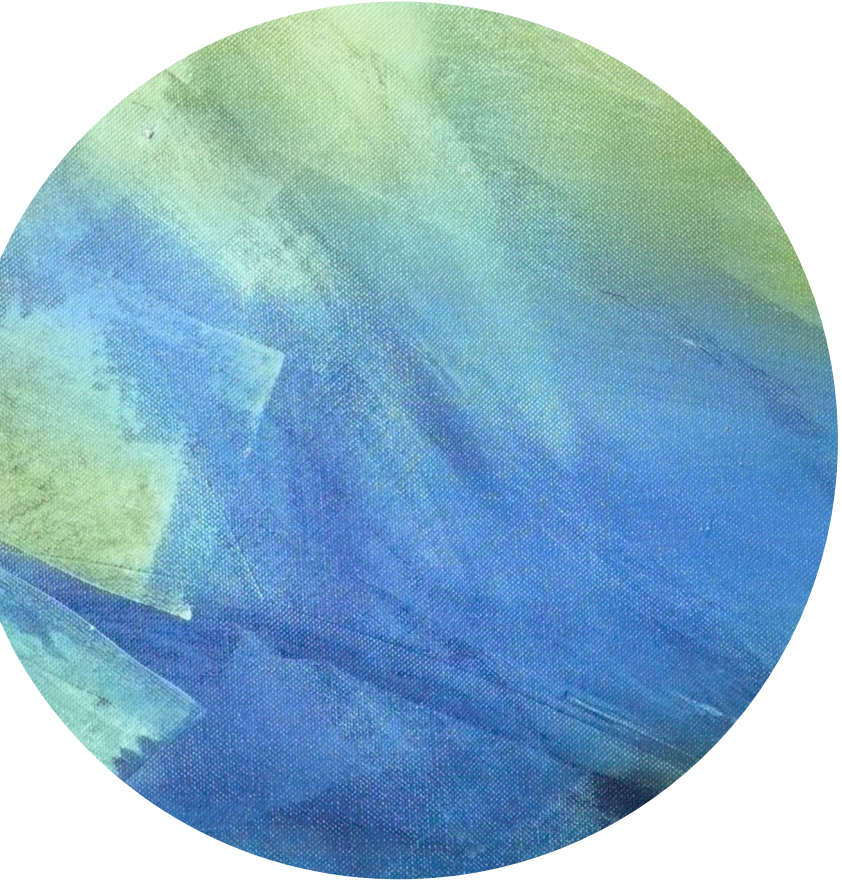
Lecture 20

Scribe : Nathan



Outline

Chaum-Pedersen ZK

OR composition

Proofs on Encrypted Data

Chaum-Pedersen

Recall: Pedersen com.

pp: $g, h(=g^a)$

$com(b) = g^b h^r$

- Public parameters: (p,g,h)
  - p: large prime (1024 bit)
  - g: generator

- Proof of a given triple being of the following form

$(u, v, w) = (g^a, g^b, g^{ab})$

DDH / Decisional Diffie-Helman.

- NP relation $\mathcal{R} = \{ (u, v, w) : \exists (a, b) \text{ s.t. } u = g^a, v = g^b, w = g^{ab} \}$

$\{ (u,v,w) : \exists\, b \text{ s.t. } v = g^b, w = u^b \}$

Chaum-Pedersen: first recall Schnorr

g

Prover

(u, v, w)

$r \leftarrow Z_p$ $(R = g^r)$

$(u, v, w) = (g^a, g^b, g^{ab})$

g

Verifier

(u, v, w)

$R = g^r$ →

$p \leftarrow Z_p$

← p

$e = r + p \cdot b$

e →

check$_1$: $g^r \cdot (v)^p = g^e$

$v = g^b$

Chaum-Pedersen: first recall Schnorr

Prover

g^{r+bit}

(u, v, w)

$r \leftarrow Z_p$ $(R = g^r).g^{bit}$ $T = R^a$

g

$(u, v, w) = (g^a, g^b, g^{ab})$

h h^b

$\xrightarrow{R, T}$

$\xleftarrow{p}$

$\xrightarrow{e}$

e= r + p.b

Verifier

(u, v, w)

$p \leftarrow Z_p$

Check: $g^e = R.(v)^p$

Check: $h^e = T(w)^p$

Chaum-Pedersen

Suppose we have a cheating.

Prover

(u, v, w)

$r \leftarrow Z_p$ $(R = g^r, \mathbf{T = g^{ar}})$

g^r $g^{ar'}$

$(R, \mathbf{T})$ $\longrightarrow$

$\longleftarrow$ p

$e = r + p.b$ $\longrightarrow$ e

Verifier

(u, v, w)

$p \leftarrow Z_p$

$(u, v, w) = (g^a, g^b, g^{ab})$

Check_1: $g^e = R.(v)^p$ $\Rightarrow$ $e = r + bp$ $\quad DLOG_g(R)$

$\mathbf{\text{Check}_2}$: $\mathbf{u^e = T.(w)^p}$ $\Rightarrow$ $e = r' + b'p$

$u^e \overset{?}{=} T(w)^p$

$g^{\cancel{a}e} \overset{?}{=} g^{\cancel{a}r + \cancel{a}bp}$

Chaum-Pedersen

$e = r + bp = r' + b'p$

$r - r' = p(b' - b)$

$\frac{r - r'}{b' - b} = p$

But, $\Pr[\mathcal{A} \text{ guesses } p] = \frac{1}{\exp(n)}$

Prover

(u, v, w)

$r \leftarrow Z_p$ $(R = g^r, T = h^r)$

$(u, v, w) = (h, g^b, w = h^b)$

(R, T) →

← p

$e = r + p.b$

e →

Verifier

(u, v, w)

$p \leftarrow Z_p$

Check_1: $g^e = R.(v)^p$

Check_2: $h^e = T.(w)^p$

Chaum-Pedersen: Soundness

Prover

(u, v, w)

$r \leftarrow Z_p$, $(R = g^r, T = g^{ar})$

$\xrightarrow{(R,\ T)}$

$\xleftarrow{p}$

$e = r + p.b$ $\xrightarrow{e}$

$\xrightarrow{(R,\ T)}$

$\xleftarrow{p'}$

$e' = r + p'.b$ $\xrightarrow{e'}$

$(u, v, w) = (g^a, g^b, g^{ab})$

Chaum-Pedersen: Soundness

Prover

(u, v, w)

$r \leftarrow Z_p$, $(R = g^r, T = g^{ar})$

$\xrightarrow{(R,\ T)}$ $\xrightarrow{(R,\ T)}$

$\xleftarrow{p}$ $\xleftarrow{c'}$

$e = r + p.b$ $\xrightarrow{e}$ $\xrightarrow{e'}$ $e' = r + p'.b$

$(u, v, w) = (g^a, g^b, g^{ab})$

Chaum-Pedersen: Zero-Knowledge

Simulator | **Verifier**

(u, v, w), guess p

$e \leftarrow Z_p$, $(R = g^e/v^p, T = u^e/w^p)$

$\xrightarrow{(R,\ T)}$

$\xleftarrow{p}$

$\xrightarrow{e}$

~~(u, v, w, g, g^ab)~~

Show: $((R, T), p, e)_{sim} = ((R, T), p, e)_{honest\ proof}$

Honest Proof.

$R = g^r \quad T = g^{ar}$

p

$e = r + p \cdot b$

Simulated.

p

$e \leftarrow Z_p$

$R = \frac{g^e}{g^{bp}} = g^{e-bp}$

$T = \frac{u^e}{g^{abp}} = u^{e-bp}$

Proofs of Encrypted Data

Encrypting bits

Prover **Verifier**

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

g^b

$g,$

$pk = g^a$

$b \leftarrow Z_p$

$Enc_{DH}(bit) : g^b, (g^{ab} \cdot g^{bit})$

$Enc_{DH}(0) = g, g^a, g^b, g^{ab}$

$Enc_{DH}(1) = g, g^a, g^b, g^{ab+1}$

(g, h, v, w)

$(g, h, v, w/g)$ is a DDH-tuple.

OR Composition

Prover **Verifier**

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?

AND

~~OR~~ Composition

Prover **Verifier**

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

What about running the two in parallel?

OR Composition

Prover **Verifier**

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

What about letting the prover simulate exactly one of them?

(g, h, v, w) is a DDH tuple (ct encrypts 0)

$g^R \rightarrow$

$\leftarrow P$

$\rightarrow$

$\leftarrow P'$

$(g, h, v, \frac{w}{g})$ is a DDH tuple (ct encrypts 1)

OR Composition

Prover **Verifier**

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

OR Composition

Prover **Verifier**

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

OR Composition

Prover **Verifier**

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