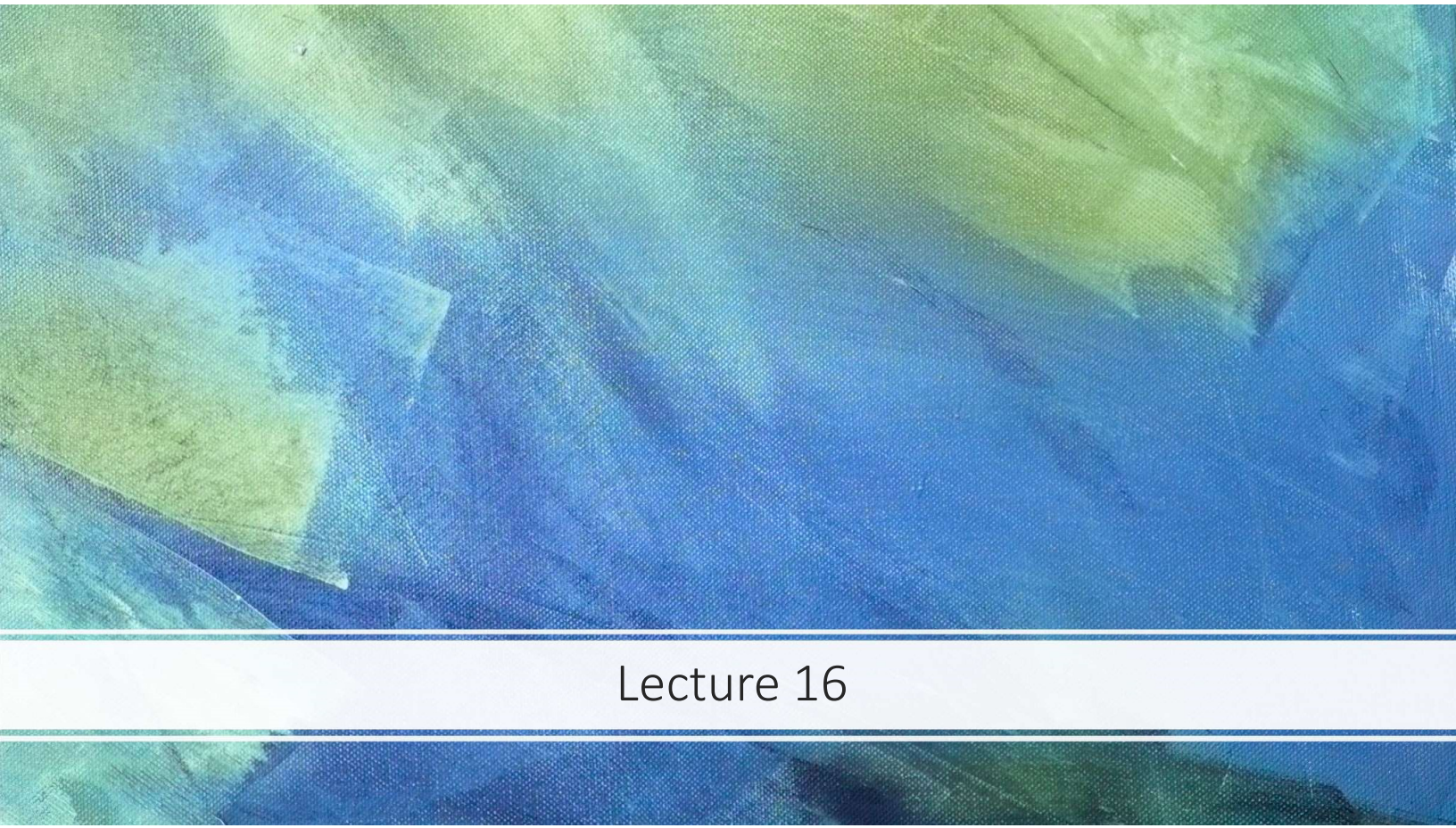
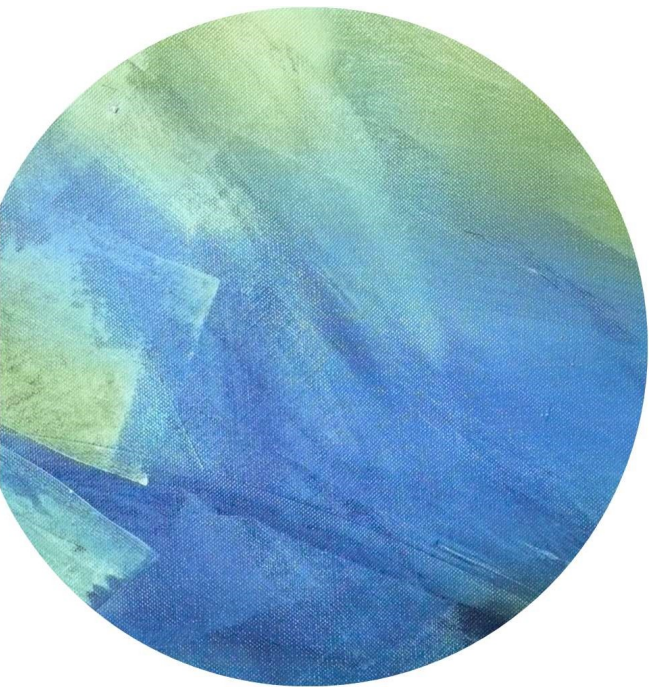

Lecture 16



Outline

Proofs on
Encrypted Data

Range Proofs

Proofs on Encrypted Data

Chaum-Pedersen

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

- Proof of a given triple being of the following form

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

  (g, u) pick b. compute $v = g^b$, $w = u^b$. DH: (g,u,v,w)

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

Chaum-Pedersen

Prover **Verifier**

(u, v, w)

$d \leftarrow Z_p$, $(v' = g^d, w' = g^{ad})$ $= u^d$

(v', w') →

← c

$c \leftarrow Z_p$

$e = d + b.c$

e →

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

(g, u, g^b, u^b)

(u, v, w, p, q)

$= (g^a, g^{ab}, g^b, g^c, g^{bc})$

Check_1: $g^e = v'.(v)^c$ $= g^d.(g^b)^c$

Check_2: $u^e = w'.(w)^c$

$= u^d.(u^b)^c$

Chaum-Pedersen

$f(x) = g^x$.

$f(x_1) \cdot f(x_2) = g^{x_1} \cdot g^{x_2} = f(x_1 + x_2)$

$(f(x))^\alpha = g^{x\alpha} = f(x\alpha)$

Prover **Verifier**

(u, v, w)

$d \leftarrow Z_p$, $v' = f(d)$, $w' = f(ad)$

(v', w') →

← c

$c \leftarrow Z_p$

$e = d + b.c$

e →

(u, v, w) = (f(a), f(b), f(ab))

g^a g^b g^{ab}

Check_1: $f(e) = v'.(v)^c = f(d).(f(b))^c$ $(\Leftrightarrow e = d + bc)$

Check_2: $f(ae) = w'.(w)^c = f(ad).(f(ab))^c$

$ae \Leftrightarrow ad + abc$

$(f(e))^a \Leftrightarrow w' . f(ab)^c$

$f(ab) \hookrightarrow w$

Chaum-Pedersen: Zero-Knowledge

Simulator **Verifier**

(u, v, w), guess c

e ← Z_p, (v' = g^e/v^c, w' = u^e/w^c)

(v', w')

c

e

DDH : (g, g^a, g^b, g^{ab})

$\approx (g, g^a, g^b, g^c)$

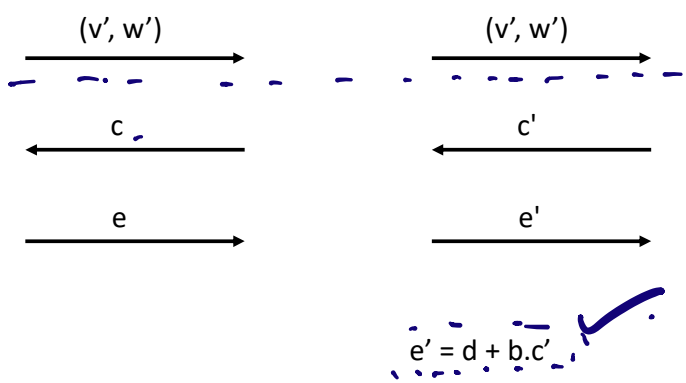
Chaum-Pedersen: Soundness

Prover

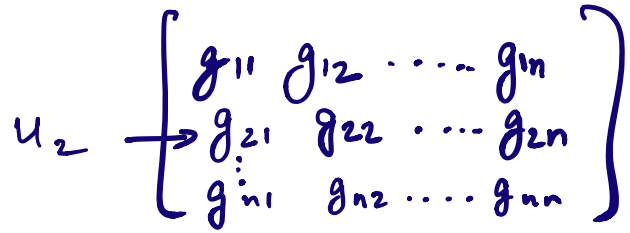
(u, v, w)

$d \leftarrow Z_p$, $(v' = g^d, w' = g^{ad})$

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



$$b = \frac{e - e'}{c' - c}$$



General Linear Relations on Exponents

Prover **Verifier**

NP relation $\mathcal{R}$ = { P, $(u_1, u_2, \ldots u_n)$: ∃ $(a_1, a_2, \ldots a_n)$ s.t. $u_i = \prod_j g_{ij}^{a_j}$, and $P(a_1, a_2, \ldots a_n)$ = true}

P, $(u_1, u_2, \ldots u_n)$

P, $(u_1, u_2, \ldots u_n)$

General Linear Relations on Exponents

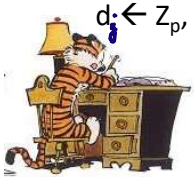
Prover **Verifier**

NP relation $\mathcal{R}$ = { P, $(u_1, u_2, \ldots u_n)$ $\exists (a_1, a_2, \ldots a_n)$ s.t: $u_i = \prod_{j=1}^{n} g_{ij}^{a_j}$, and $P(a_1, a_2, \ldots a_n)$ = true}

$d_1, \ldots, d_n$

P, $(u_1, u_2, \ldots u_n)$ P, $(u_1, u_2, \ldots u_n)$

$d_j \leftarrow Z_p$, $(u'_i = \prod_{j=1}^{n} g_{ij}^{d_j})$



$(u'_1, u'_2, \ldots u'_n)$

c

each u'_i is basically u_i with d_j in exp. instead of a_j

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

$g_1 = g$

$g_2 = u = g^a$

$\exists b$ s.t. $v = g_1^b$, $w = g_2^b$

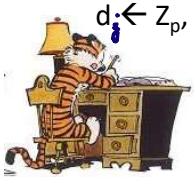
General Linear Relations on Exponents

Prover **Verifier**

NP relation $\mathcal{R} = \{ P, (u_1, u_2, \ldots u_n) : \exists (a_1, a_2, \ldots a_n)$ s.t. $u_i = \prod g_{ij}^{a_j}$, and $P(a_1, a_2, \ldots a_n) = true\}$

$P, (u_1, u_2, \ldots u_n)$ $P, (u_1, u_2, \ldots u_n)$

$d_j \leftarrow Z_p$, $(u'_i = \prod g_{ij}^{d_j})$



$(u'_1, u'_2, \ldots u'_n)$

c

$c \leftarrow Z_p$

$\forall j \in [n], e_j = a_j \cdot c + d_j$ $e_1, \ldots e_n$

General Linear Relations on Exponents

Prover **Verifier**

NP relation $\mathcal{R}$ = { P, $(u_1, u_2, \ldots u_n)$: ∃ $(a_1, a_2, \ldots a_n)$ s.t. $u_i = \prod_j g_{ij}^{a_j}$, and $P(a_1, a_2, \ldots a_n)$ = true }

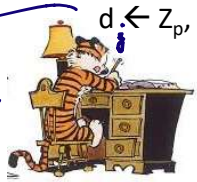
$\Rightarrow P'(g^{a_1} \ldots g^{a_n})$ = true

P, $(u_1, u_2, \ldots u_n)$

P, $(u_1, u_2, \ldots u_n)$

subject to satisfying predicate

$d_j \leftarrow Z_p$, $(u'_i = \prod_j g_{ij}^{d_j})$



$(u'_1, u'_2, \ldots u'_n)$

c

$c \leftarrow Z_p$

$e_j = d_j + a_j.c$

$e_1, e_2, \ldots, e_n$

$\forall i \in [n], \prod_j g_{ij}^{e_j} \stackrel{?}{=} u'_i \cdot u_i^c$

$= \prod_j g_{ij}^{d_j} \prod_j g_{ij}^{a_j.c}$

$= \prod_j g_{ij}^{d_j + a_j.c}$

Equality of Ciphertexts

Recall: El-Gamal Encryption

$PK = (g, h)$, $SK = a$ s.t. $g^a = h$, $Enc_{PK}(m; r) = g^r, h^r.m$

Equality of Ciphertexts

Recall: El-Gamal Encryption

$PK = (g, h)$, $SK = a$ s.t. $g^a = h$, $Enc_{PK}(m; r) = g^r, h^r.m$

$PK1 = (g, h_1)$, $PK2 = (g, h_2)$

$ct_1 = Enc_{PK1}(m; r_1) = (g^{r1}, h_1^{r1}.m)$
$ct_2 = Enc_{PK2}(m; r_2) = (g^{r2}, h_2^{r2}.m)$

ct_1 , ct_2 and a proof that
both encrypt the same message

Equality of Ciphertexts

Recall: El-Gamal Encryption

$PK = (g, h)$, $SK = a$ s.t. $g^a = h$, $Enc_{PK}(m; r) = g^r, h^r.m$

$PK1 = (g, h_1)$, $PK2 = (g, h_2)$

$ct_1 = Enc_{PK1}(m; r_1) = (g^{r1}, h_1^{r1}.m) = (p_1, q_1)$

$ct_2 = Enc_{PK2}(m; r_2) = (g^{r2}, h_2^{r2}.m) = (p_2, q_2)$

ct_1 , ct_2 and a proof that both encrypt the same message

There exist r1, r2 such that:
$p_1 = g^{r1}$, $p_2 = g^{r2}$, $q_1/q_2 = h_2^{r2}/h_1^{r1}$

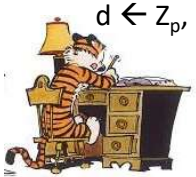
General Linear Relations on Exponents

Prover **Verifier**

NP relation $\mathcal{R}$ = { P, $(u_1, u_2, \ldots u_n)$: ∃ $(a_1, a_2, \ldots a_n)$ s.t. $u_i = \Pi g^{a}$, and $P(a_1, a_2, \ldots a_n)$ = true}

P, $(u_1, u_2, \ldots u_n)$

d ← Z_p, $(u'_i = \Pi g$)



$(u'_1, u'_2, \ldots u'_n)$

c

$e_1, e_2, \ldots, e_n$

$e_j = d + a_j.c$

P, $(u_1, u_2, \ldots u_n)$

c ← Z_p

Range Proofs

Encrypting bits

Prover **Verifier**

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

$b \bmod |G| \in \{0,1\}$

$Enc(b; r)$ w.r.t. $pk = (g, h)$

$= (g, h, g^r, h^r \cdot g^b) = (g, h, v, w)$

✓ If $b = 0$: (g,h,v,w) is a DH tuple.

$= (g, h, g^r, h^r)$

✓ If $b = 1$: $(g, h, v, \frac{w}{g})$ is a DH tuple

$\hookrightarrow (g, h, g^r, h^r)$

OR Composition

Prover **Verifier**

(g, h, v, w) is DH tuple

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

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?

(g, h, v, w)

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

true false

c_0

What about letting the prover simulate exactly one of them? guess c_1

Make it so that P controls exactly one out of c_0 & c_1

Verifier picks c

Set c_0, c_1 s.t. $c_0 \oplus c_1 = c$.

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

Sim → (a_1, c_1, z_1) real an : a_1

c

$c_0 = c \oplus c_1$

(c_0, z_0) : c_1, z_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