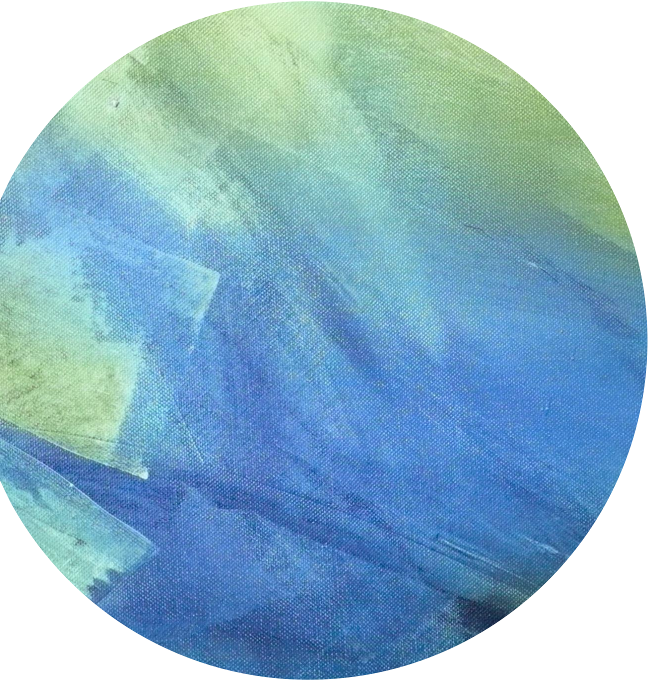
Lecture 13



Outline

Schnorr Signatures

Commitments

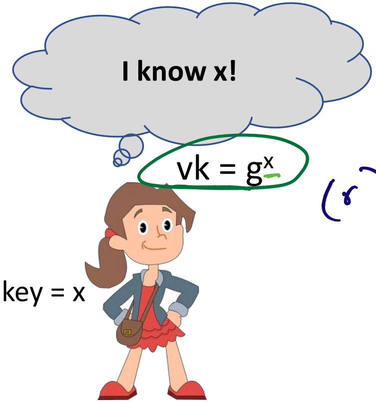
Schnorr Signatures

Schnorr Signatures

- Signatures from groups
  - Gen outputs (vk = g^x, sign key = x)
  - Sign (m, sign key) :
  - Verify (σ, vk, m) :

Schnorr Signatures

$X = g^x, \; R = g^r, \; h$



sign key = x

(r)

$vk = g^x = X$

$R = g^r \longrightarrow$

$\longleftarrow h$

$s \longrightarrow$

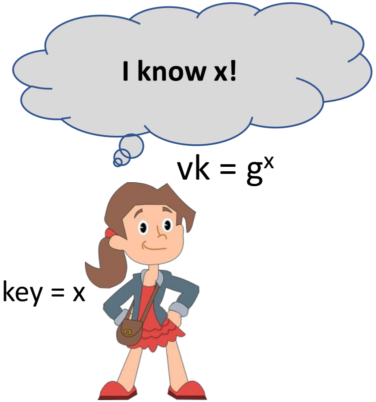
$s = r + hx$

"Alice must know x"

$g^s = (g^r)(g^x)^h = R \cdot X^h$

does not know r.

Schnorr Signatures



vk = g^x

sign key = x

Signer

m

$\mathrm{Sign}(m, \overset{x}{sk}) = ??$

Receiver

$\underline{R} = g^r, \ \underline{h} = H(m \| R), \ \underline{s} = (r + h \cdot x)$

Schnorr Signatures

- Signatures from groups
  - Gen outputs (vk = g^x, sign key = x)

    what about $s = rh + x$?

  - Sign (m, sign key) = $R=g^r$, h=H(m,R), s = r + hx. Output ($h, s, \underline{R}$) .

    Verification:

    $$\boxed{g^s = R \cdot X^h}$$

    where $h = H(m \| R)$

    $$R = g^s \cdot X^{-h}.$$

  - Verify (σ, vk, m) : Check if $\boxed{h = H(m, g^s X^{-h})}$
- Is this secure?

Schnorr Signatures

- Signatures from groups
  - Gen outputs (vk = g^x, sign key = x)
  - Sign (m, sign key) = $R=g^r$, $h=H(m,R)$, $s = r + hx$. Output (R,s)
  - Verify (σ, vk, m) : Check if $g^s = RX^h$ for $h = H(m,R)$
- Is this secure?

A forger can be used to get distinct signatures (h1,s1), (h2,s2) with same (m,R) (different h, by programming the RO), and that lets us solve for x

Schnorr Signatures

"Programmability":

RO

(m, R)

h

vk = g^x

m ,

(h, s, R)

1) $h = H(m, R)$

s.t. verification passes.

Game 1

m, R

h_1

A

(h_1, s_1, R)

Game 2

m, R

h_2

A

(h_2, s_2, R)

$s_1 = r + h_1 x$

$s_2 = r + h_2 x$

$(s_1, s_2, h_1, h_2$ are known)

Schnorr Signatures

$vk = g^x$

m

Multi-Signatures

- Multiple signers signing the same message
- Each signer has an (SK,VK) pair
- Resulting signature must be “compact”: size independent of #signers

Multi-Signatures

- Multiple signers signing the same message

x_i g^{x_i}

- Each signer has an (SK,VK) pair

- Resulting signature must be “compact”: size independent of #signers

- Security requirement: Unforgeability (chosen message security)

- Adversary can collude with all but one signer

Multi-Signatures

- Schnorr: $sk = x$, $vk = X = g^x$
  - Sign (m, x) : $R=g^r$, $h=H(m,R)$, $s = r + hx$. Output (R,s)
  - Verify (σ, X, m) : Check if $g^s = RX^h$ for $h = H(m,R)$

Multi-Signatures

- Schnorr: $sk = x$, $vk = X = g^x$
  - Sign (m, x) : $R=g^r$, $h=H(m,R)$, $s = r + hx$. Output (R,s)
  - Verify (σ, X, m) : Check if $\boxed{g^s = RX^h}$ for $h = H(m,R)$
- Multi-signatures:
  - Multiple signers with signing keys $x_1,\ldots,x_n$ and verification keys $\boxed{X_1,\ldots,X_n}$ can create "aggregated" signature (R,s) such that $g^s = R.X_1^{h\_1}\ldots X_n^{h\_n}$

(s, R)

$g^s = R \cdot X_1^{h_1} X_2^{h_2} \cdots X_n^{h_n}$

Multi-Signatures

- Schnorr: $sk = x$, $vk = X = g^x$
  - Sign (m, x) : $R=g^r$, $h=H(m,R)$, $s = r + hx$. Output (R,s)
  - Verify (σ, X, m) : Check if $\boxed{g^s = RX^h}$ for $h = H(m,R)$

$s = r_1 + (x_1 \cdot h_1) + (r_2 + x_2 \cdot h_2) \ldots$

- Multi-signatures:
  - Multiple signers with signing keys $x_1,\ldots,x_n$ and verification keys $X_1,\ldots,X_n$ can create "aggregated" signature (R,s) such that $\boxed{g^s = R.X_1^{h\_1}\ldots X_n^{h\_n}}$
  - Each party picks r_i and publishes g^{r_i} (R_i). Set R = $\boxed{g^{r\_1+\ldots+r\_n}}$. $= R_1 R_2 \ldots R_n$.
  - Set $h\_i = H(m,R,X_i,L)$, where $L = (X_1,\ldots,X_n)$

$h_i = \cancel{H(m, R)}$

$H(m, R, X_i, X_1, \ldots X_n)$

Multi-Signatures

Final sign: (s_n, R)

$R_1 R_2 \ldots R_n = g^{r_1} g^{r_2} \ldots g^{r_n}$

- Schnorr: sk = x, vk = $X = g^x$
  - Sign (m, x) : $R=g^r$, h=H(m,R), s = r + hx. Output (R,s)
  - Verify (σ, X, m) : Check if $g^s = RX^h$ for h = H(m,R)

- Multi-signatures:
  - Multiple signers with signing keys $x_1,\ldots,x_n$ and verification keys $(X_1,\ldots.,X_n)$ can create "aggregated" signature (R,s) such that $g^s = R.X_1^{h\_1}\ldots X_n^{h\_n}$
  - Each party picks r_i and publishes g^{r_i}. Set R = $g^{r\_1+\ldots+r\_n}$.
  - Set h_i = $H(m,R,X_i,L)$, where L = $(X_1,\ldots,X_n)$
  - Then, sequentially $s_i = s_{i-1} + r_i + h_i x_i$ (starting with $s_0 = 0$).
  - So that final signature $s_n = r + h_1x_1 + \ldots + h_nx_n$ where R= g^r.

$s_0 = 0$

$s_1 = s_0 + r_1 + h_1 x_1$

$s_2 = s_1 + r_2 + h_2 . x_2$

$\vdots$

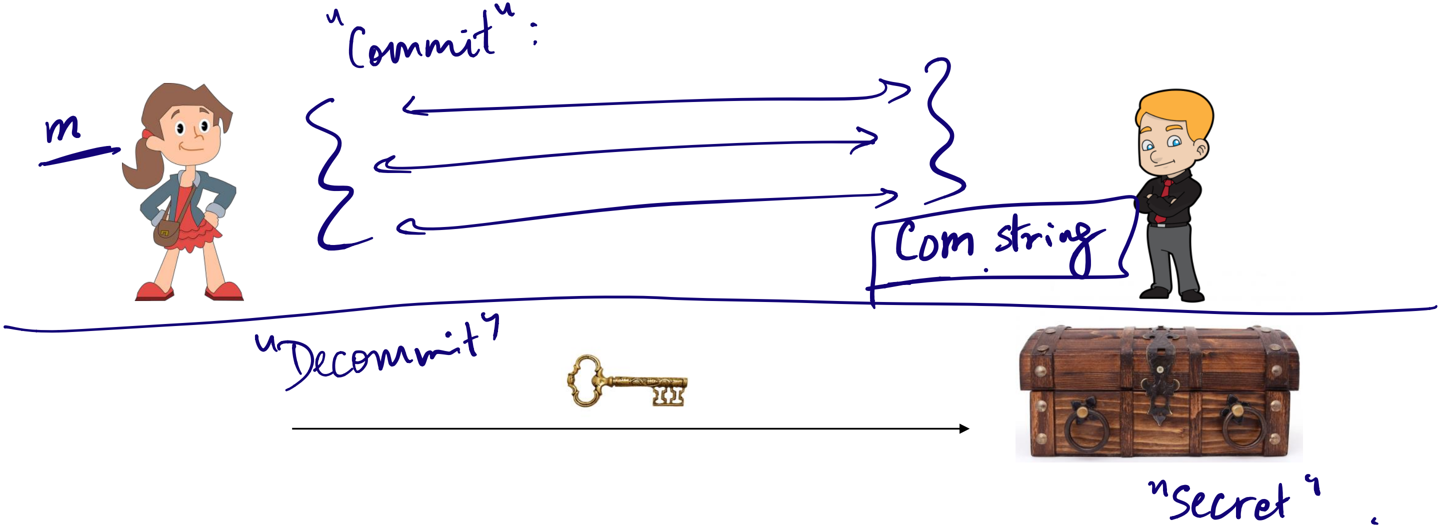
$s_n = \Sigma s_i + \Sigma r_i + \Sigma h_i x_i$

Commitments

Commitments

Commitments

Commitments



Commitments

- Hiding $\forall m_0, m_1$ $com(m_0; r) \approx_c com(m_1; r)$
- Binding $\forall$ string $c \in \{0,1\}^*$,

$Pr\left[\exists \begin{pmatrix} k_1, m_1, \\ k_2, m_2 \end{pmatrix} \text{ s.t. } \left(Decommit(c, k_1, m_1) = 1\right) \wedge \left(Decommit(c, k_2, m_2) = 1\right)\right]$

$= negl$

$Pr\left(\begin{matrix} PPT \\ Adv \end{matrix} \text{ can find } (k_1, m_1, k_2, m_2) \text{ s.t. } \left(Decom(c, k_1, m_1) = 1\right) \wedge \left(Decom(c, k_2, m_2) = 1\right)\right)$

$= negl.$

Examples

- ~~If~~ Is (g, g^x) a commitment to x?

Binding: only if x is small enough ($\leq p$).

NOT. Hiding? Any attacker that knows x, can check if $(g, h) = (g, g^x)$

- Ct = E (k, m) for a symmetric key encryption E

NOT commitment because not binding

(Recall: one-time pad $E(k, m) = k \oplus m$).

C		R
$m, y = (k \oplus m)$	$y = E(k, m)$ $\longrightarrow$	$\boxed{y}$
Decommit:	m', k' $\longrightarrow$	$m' = m \oplus 1, \; k' = k \oplus 1.$

Examples

To com to M. pick r. Send $H(r \| M)$

In practice, we use:

- To commit to message M, choose random, fixed-length r, send H(r || M)
- To open commitment, send r, M
- Receiver cannot fully recover M.

- Sender cannot find another M' to open.

to com to M, pick large r. $Com(M;r) =$ SHA $(M \| r)$ MD5

(computational binding).

Pedersen Commitments

- Public parameters: (p,g,h)
  - p: large prime (1024 bit)
  - g: generator
  - h: g^a for hidden a
- Protocol
  - To commit to x, C chooses random r and sends ($g^x h^r$) to R.
  - To open, C sends x and r to R.
- Benefits:
  - One can prove many things about the committed value without opening it

Pedersen Commitments

- Unconditionally hiding
  - Given a commitment c, every value x is equally likely to be the value committed in c.
  - For example, given x,r, and any x', there exists r' such that $g^x h^r = g^{x'} h^{r'}$, in fact $r = (x-x')a^{-1} + r \bmod q$.

Pedersen Commitments

- Computationally binding
  - Suppose committer sent $g^x h^r$ mod p for some (x, r)
  - Now it finds x' ≠ x and r' such that $c = g^{x'} h^{r'}$.

  - This means that the sender ``knows'' $\log_g(h) = (x'-x)\cdot(r-r')^{-1}$.

  - This means: assuming DL is hard, the sender cannot open the commitment with another value.

Prove Knowledge of Discrete Log without revealing it?

PROVE:
I know x!

vk = g^x

sign key = x

Prove Knowledge of Discrete Log without revealing it?