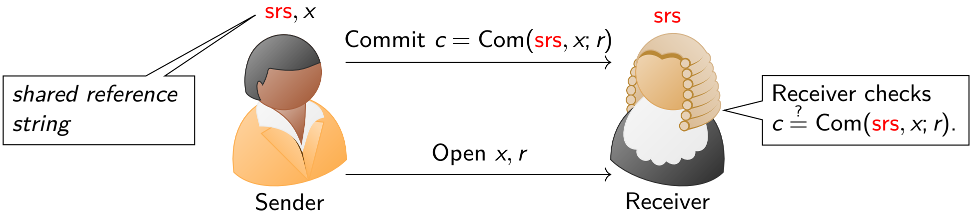
CS171: Cryptography

Lecture 19

Sanjam Garg

Commitment Schemes

- Bind to a secret value that cannot be later explained with an alternate value.



- Correctness: A sender should be able to convince an honest receiver of the correct opening with *overwhelming* probability. (Easy to see)
- Binding: No PPT cheating sender can find two openings for the same commitment. That is, $\forall$ PPT $\mathcal{A}$ we have that

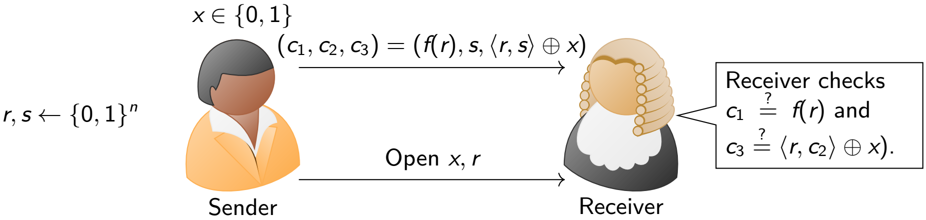
$$\Pr[(x, r, x', r') \leftarrow \mathcal{A}(1^\lambda, \mathsf{srs}) \text{ such that } x \neq x' \text{ and } \mathsf{Com}(\mathsf{srs}, x, r) = \mathsf{Com}(\mathsf{srs}, x', r')] = \mathsf{neg}(\lambda)$$

- Hiding: The commitment doesn't leak any information about the committed value x. That is, $\forall$ PPT $\mathcal{A}, x, x'$ we have that

$$\left|\Pr[\mathcal{A}(1^\lambda, \mathsf{srs}, \mathsf{Com}(\mathsf{srs}, x; r)) = 1] - \Pr[\mathcal{A}(1^\lambda, \mathsf{srs}, \mathsf{Com}(\mathsf{srs}, x'; r')) = 1]\right| \leq \frac{1}{2} + \mathsf{neg}(\lambda)$$

Commitment Scheme From Hardness Concentration

$f\colon \{0,1\}^n \to \{0,1\}^n$ be a one-way permutation



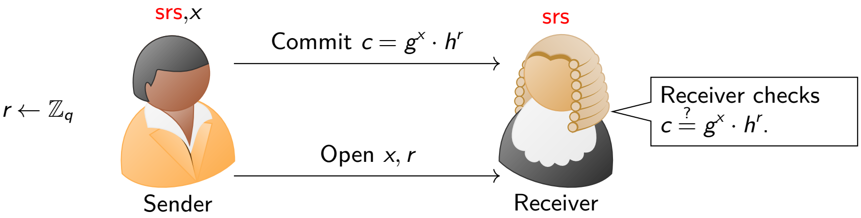
- **Binding:** Because f is a permutation, given c there is a unique value of r, x such that $c_1 = f(r)$ and $c_3 = \langle r, c_2\rangle \oplus x)$.
- **Hiding:** Follows from the hardness concentration property.

Can we use any encryption algorithm to get a commitment scheme?

- Given $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ let sender execute $\mathsf{Com}(x; r)$ as follows. Use randomness r to execute Gen and then encrypt x using Enc and the obtained key k.
- No!
- While this commitment offers hiding, it doesn't give binding.
- Shouldn't binding come from the correctness of encryption?
- The encrypter may not choose their random coins honestly.

Pederson Commitment Schemes

$\mathsf{srs} = (G, g, q, h)$



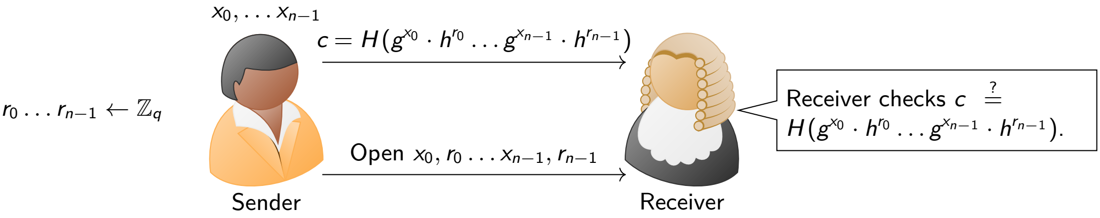
- **Binding:** Given x, x', r, r' such that $g^x \cdot h^r = c = g^{x'} \cdot h^{r'}$ we can compute $dlog_g(h)$.
- **Hiding:** For every $c = g^x h^r$ and x' there exists $r' = r + \frac{x' - x}{dlog_g(h)}$.

Commitment to a vector $\mathbf{x} = (\mathbf{x_0}, \ldots \mathbf{x_{n-1}})$

Send $c_i = \mathrm{Com}(x_i; r_i)$ for each i.

Can we do it succinctly?

Merkle Commitment Schemes



- Hashing in More Detail ($n = 2^\ell$): For every $i \in \{0, n-1\}$, $c_i^0 = g^{x_i} h^{r_i}$. For all $j \in \{0, \ldots \ell - 1\}, i \in \{0 \ldots 2^j - 1\}$ set $c_{i/2}^{j+1} = H(c_i^j || c_{i+1}^j)$. Finally, $c = c_0^\ell$.
- Binding: An attacker that outputs distinct $x_0, r_0, \ldots x_{n-1}, r_{n-1}$ and $x'_0, r'_0, \ldots x'_n, r'_n$ such that $\exists i$ with $x_i \neq x'_i$ and the receiver checks pass on both can be used to break either (i) CRHF, or (ii) compute $dlog_g(h)$.
- Hiding: For every $c_i^0 = g^{x_i} h^{r_i}$ that is hashed and x'_i there exists $r'_i = r_i + \frac{x'_i - x_i}{dlog_g(h)}$.
- Partial Opening (Location k): Opening c_k^0, x_k, r_k and $\forall j \in \{0, \ell\}$ send $c^j_{\frac{k}{2^j}}$ and $c^j_{\frac{k}{2^j}+1}$.

Commitment to a Polynomial $f(x)$ of degree $n-1$

Succinctly

Polynomial Interpolation

Problem: Given $a_0 \ldots a_{n-1}$ **(evaluation representation) find <u>the</u> degree-**$n-1$ **polynomial** $f(x) = b_0 + b_1 x + \ldots b_{n-1} x^{n-1}$ **(coefficient representation), i.e.** $b_0, b_1 \ldots b_{n-1}$**, such that for all** $i \in H = \{0, \ldots n-1\}$ **we have** $f(i) = a_i$**.**

- Let $L_i(x)$ be the degree-$n-1$ polynomial such that $L_i(i) = 1$ and for all $j \in H \backslash \{i\}$ $L_i(j) = 0$

$$L_i(x) = \frac{\prod_{j \in H \backslash \{i\}} (x - j)}{\prod_{j \in H \backslash \{i\}} (i - j)}.$$

- Next, we have

$$f(x) = \sum_{i \in H} a_i \cdot L_i(x)$$

- L_is can be cached for efficiency. DIY: Prove that the constructed polynomials are correct and unique.

KZG Polynomial Commitment/Pairing Curve BLS12-381

- Gives groups $G_1 = \langle g_1 \rangle, G_2 = \langle g_2 \rangle$ and G_T (of the same prime order p) along with a bilinear pairing operation e.
- For every $\alpha, \beta \in \mathbb{Z}_p^*$, we have that $e(g_1^{\alpha}, g_2^{\beta}) = e(g_1, g_2)^{\alpha\beta}$.
- **Setup:** srs generation that supports committing to degree $d-1$ polynomials:
  - Sample $\tau \leftarrow \mathbb{Z}_p^*$.
  - srs $= (h_0 = g_1, h_1 = g_1^{\tau}, g_1^{\tau^2}, \ldots h_d = g_1^{\tau^{d-1}}, g_2, h' = g_2^{\tau})$
- **Commitment:** Given srs and a polynomial $f(x) = c_0 + c_1 x + \ldots c_{d-1} x^{d-1}$ of degree $d-1$, we can compute Com(f) as:

$$F = \mathrm{Com}(f) = g_1^{f(\tau)} = \prod_{i=0}^{d-1} h_i^{c_i}$$

- **Opening:** Show that $f(z) = s$. In this case, $g(x) = f(x) - s$ is such that $g(z) = 0$. Or, $x - z$ divides $f(x) - s$.
- Sender computes $T(x) = \frac{f(x)-f(z)}{x-z}$ and sends $W = \mathrm{Com}(T)$.
- **Receiver Accepts if:** $e\left(\frac{F}{g_1^s}, g_2\right) = e\left(W, \frac{h'}{g_2^z}\right)$.

Optimizing Opening by Batching — Warmup

Often we want to check multiple pairing equations:

$$e(F_0, g_2) = e(W_0, h_2)$$

$$e(F_1, g_2) = e(W_1, h_2)$$

$$e(F_2, g_2) = e(W_2, h_2)$$

A faster way to check? The receiver samples a random γ and checks:

$$e\left(\prod_{i=0}^{2} F_i^{\gamma^i}, g_2\right) = e\left(\prod_{i=0}^{2} W_i^{\gamma^i}, h_2\right)$$

Need only 2 pairings instead of 6.

Optimizing Opening by Batching

- **Problem:** Consider the setting where sender commits to polynomials $f_1 \dots f_t$ as $F_1 \dots F_t$ and wants to show that for all i we have that $f_i(z) = s_i$.
- **Opening:** Receiver sends random γ. Sender computes $T(x) = \sum_{i=1}^{t} \gamma^{i-1} \cdot \frac{f_i(x) - f_i(z)}{x-z}$ and sends $W = \mathrm{Com}(T)$.
- **Receiver Accepts if:** $e\left(\prod_{i=1}^{t} \left(\frac{F_i}{g_1^{s_i}}\right)^{\gamma^{i-1}}, g_2\right) = e\left(W, \frac{h'}{g_2^z}\right)$. (only two pairings)

KZG Commitment is Homomorphic

- Given commitments c_1, c_2 to polynomials $f_1(x)$ and $f_2(x)$ find a commitment to the polynomial $g(x) = f_1(x) + f_2(x)$?
- Output Commitment as $c_1 \cdot c_2$.