Polynomial Commitments

Lecture 20

Bilinear Pairing-based Approaches

Recall

Polynomial Commitment

- Prover wants to (succinctly) commit to a polynomial and later let the verifier (interactively) evaluate it on points of its choice
  - Generally, a multi-variate polynomial with a known number of variables and known degree
    - e.g., a multi-linear polynomial in GKR. In some other applications, univariate polynomial of a known degree
- Trivial solution: send the coefficients of the polynomial
  - But not succinct and evaluating the polynomial is expensive
  - Want verifier's computation/communication to be sub-linear in the size of the polynomial
- Non-trivial solutions: Using Merkle hashes and low-degree tests; from hardness of discrete logarithm; from bilinear pairings; using "IOPs"; ...

Polynomial Commitment

- Today: Discrete Log based approaches
  - Based on homomorphic commitment
- First scheme: short commitments, long proofs
- Second scheme: Bulletproofs: short commitments and proofs, but verification time is still linear
  - Using bilinear pairings (later), can reduce the verification time as well
- Tools: homomorphic commitments and Sigma protocols (3-message, public-coin, honest verifier ZK proofs with "special soundness")

Not important for (non-ZK) SNARKs.

Bilinear Pairings

- Groups $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_t$, of prime order p
- e: $\mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_t$, such that for generators g_1, g_2 of $\mathbb{G}_1$, $\mathbb{G}_2$,

  $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$
- $e(g_1^a, \Pi_i\, g_2^{x_i}) = e(g_1, g_2)^{a(\Sigma_i\, x_i)} = \Pi_i\, e(g_1^a, g_2^{x_i})$
- When $\mathbb{G}_1 = \mathbb{G}_2$, DDH cannot hold in that group
  - But otherwise it could hold in both: SXDH (Symmetric External Diffie-Hellman) assumption

KZG scheme

- Recall: $P(\alpha)=v \Leftrightarrow (X-\alpha)$ divides $P(X)-v$
  - i.e., $\exists$ polynomial Q (of degree one less) s.t. $(X-\alpha)Q(X) = P(X)-v$
  - Plan: Prover commits to $Q(\beta)$ (as $g^{Q(\beta)}$) . Verifier would homomorphically check the equation at $X=\beta$ for a secret $\beta \leftarrow \mathbb{F}$
    - Prover needs to commit to $Q(\beta)$ without knowing β. (A public coin Verifier also cannot know β.)
    - Idea: Have a trusted party provide commitments of β^i
    - Problem: Need commitment to allow homomorphic multiplication of two committed values, namely $Q(\beta)$ and $\beta-\alpha$
    - Possible using pairings. Will use $\mathbb{G}_1 = \mathbb{G}_2$

KZG scheme

- To check $(X-\alpha)Q(X) = P(X)-v$
- A trusted setup: prime order group G and generator g, and for a random $\beta \leftarrow \mathbb{F}$, the group elements $g^{\beta}, g^{\beta^2}, \dots, g^{\beta^d}$
- Prover commits to $P(\beta)$ where $P(X) = \Sigma_{i=0}^{d} c_i X^i$: $z = g^{P(\beta)} = \Pi_{i=0}^{d} [g^{\beta^i}]^{c_i}$
- Verifier sends $\alpha \leftarrow \mathbb{F}$. Prover sends $w = g^{Q(\beta)}$ where $Q(X) = \dfrac{P(x) - v}{X - \alpha}$
- Verifier checks $e(z, g^{-v}) = e(w, g^{\beta} \cdot g^{-\alpha})$
- If the prover can open $P(\beta)$ to two distinct values v_1, v_2, then can break "strong Diffie-Hellman assumption" (SDH)
  - SDH: Given $g^{\beta}, g^{\beta^2}, \dots, g^{\beta^d}$ it is infeasible to output $(\alpha, g^{1/(\beta-\alpha)})$
  - If w_1, w_2 s.t. $e(z, g^{-v_j}) = e(w_j, g^{\beta} \cdot g^{-\alpha})$ for both j=1,2 then $(w_1 \cdot w_2^{-1})^{1/(v_2-v_1)} = g^{1/(\beta-\alpha)}$
- Under SDH, Prover can open $P(\beta)$ to at most one value, but not guaranteed that P is a polynomial. In the "Generic Group Model" becomes an extractable polynomial commitment scheme.

KZG scheme

Alternate Version

- To avoid the heuristic Generic Group Model
- But will rely on a "knowledge" assumption called "Power Knowledge of Exponent" assumption
  - Idea: Given (g,g^γ), $(g^\beta,g^{\gamma\beta})$, $(g^{\beta^2},g^{\gamma\beta^2})$,..., $(g^{\beta^d},g^{\gamma\beta^d})$, the only way to find a pair (h,h^γ) is to set $h = \prod_{i=0}^{d} [g^{\beta^i}]^{c_i}$ and $h^\gamma = \prod_{i=0}^{d} [g^{\gamma\beta^i}]^{c_i}$
    - Only way: From any adversary which can do this, can extract $c_0,...,c_d$ which satisfy $h = \prod_{i=0}^{d} [g^{\beta^i}]^{c_i}$
  - Generalises earlier "knowledge" assumptions
    - KEA1: Given (g,g^γ) to output (h,h^γ) must know c s.t. $h=g^c$
    - KEA3: Given (g,g^γ),$(g^\beta,g^{\gamma\beta})$, to output (h,h^γ) must know c_0, c_1 s.t. $h=g^{c_0} (g^\beta)^{c_1}$

KZG scheme

Alternate Version

- Power Knowledge of Exponent assumption: Given (g, g^γ), $(g^\beta, g^{\gamma\beta})$, $(g^{\beta^2}, g^{\gamma\beta^2})$,..., $(g^{\beta^d}, g^{\gamma\beta^d})$, from any adversary which can find a pair (h, h^γ) with significant probability, one can extract $c_0,\ldots,c_d$ which satisfy $h = \Pi_{i=0}^{d} [g^{\beta^i}]^{c_i}$

- Trusted setup has G and (g, g^γ), $(g^\beta, g^{\gamma\beta})$, $(g^{\beta^2}, g^{\gamma\beta^2})$,..., $(g^{\beta^d}, g^{\gamma\beta^d})$
- Prover sends $z = g^{P(\beta)} = \Pi_{i=0}^{d} [g^{\beta^i}]^{c_i}$ and $z' = z^\gamma = \Pi_{i=0}^{d} [g^{\gamma\beta^i}]^{c_i}$
- Verifier sends $\alpha \leftarrow \mathbb{F}$. Prover sends $w = g^{Q(\beta)}$ where $Q(X) = \dfrac{P(x) - v}{X - \alpha}$
- Verifier checks $e(z, g^{-v}) = e(w, g^\beta \cdot g^{-\alpha})$ and that $e(z, g^\gamma) = e(z', g)$
  - The second check ensures that $z = g^{P(\beta)}$, and the prover knows P; hence it can complete the proof with $v=P(\beta)$. The first check, as before, ensures that it can do this only for one value of v, without breaking SDH (given $g^{\gamma\beta^i}$ in addition, for a random γ; but in the SDH experiment adversary can get them from g^{β^i})

Dory

- Recall Bulletproofs:
  - A proof of knowledge of $\mathbf{x}\in\mathbb{F}_p^n$, given $\mathbf{G}\in\mathbb{F}_p^n$ and $g^{\langle \mathbf{x},\mathbf{G}\rangle}$, in parallel with a proof that $\langle \mathbf{x},\mathbf{y}\rangle = v$ for a given $\mathbf{y}\in\mathbb{F}_p^n$ (using same challenges)
  - Prover sends $g^{\langle \mathbf{x}_L,\mathbf{G}_R\rangle}$, $g^{\langle \mathbf{x}_R,\mathbf{G}_L\rangle}$, $\langle \mathbf{x}_L,\mathbf{y}_R\rangle$, $\langle \mathbf{x}_R,\mathbf{y}_L\rangle$. Verifier sends $\alpha\leftarrow\mathbb{F}_p$
    - Recurse on $g^{\langle \mathbf{x}',\mathbf{G}'\rangle}$ and $\langle \mathbf{x}',\mathbf{y}'\rangle$ computed using values sent by Prover
      - $\langle \mathbf{x}',\mathbf{G}'\rangle = \langle \mathbf{x},\mathbf{G}\rangle + \alpha^2 \langle \mathbf{x}_L,\mathbf{G}_R\rangle + \alpha^{-2} \langle \mathbf{x}_R,\mathbf{G}_L\rangle$

        $\langle \mathbf{x}',\mathbf{y}'\rangle = \langle \mathbf{x},\mathbf{y}\rangle + \alpha^2 \langle \mathbf{x}_L,\mathbf{y}_R\rangle + \alpha^{-2} \langle \mathbf{x}_R,\mathbf{y}_L\rangle$

        [$\mathbf{x}' = \alpha\mathbf{x}_L + \alpha^{-1}\mathbf{x}_R$, $\mathbf{G}' = \alpha^{-1}\mathbf{G}_L + \alpha\mathbf{G}_R$, $\mathbf{y}' = \alpha^{-1}\mathbf{y}_L + \alpha\mathbf{y}_R$]
    - Base case when n=1: prover sends $\mathbf{x}$
  - To compute $g^{\langle \mathbf{x}',\mathbf{G}'\rangle}$ and $\langle \mathbf{x}',\mathbf{y}'\rangle$ verifier takes linear time (in the first iterations as well as over all)
  - Idea to reduce verification time: Prover carries out the computation, and proves to the verifier that it is consistent with a publicly pre-computed succinct commitment of g_i, i=1 to n (and with $\mathbf{y}\in\mathbb{F}_p^n$)

Dory

- Vector Commitment of Group Elements
  - Public parameters: $h \in \mathbb{G}_1$, and for i=1 to n, $g_i = g^{G_i} \leftarrow \mathbb{G}_2$
  - For $\mathbf{m} \in \mathbb{G}_1^n$ and $\rho \leftarrow \mathbb{G}_2$, $Com_{(h,\mathbf{g})}(\mathbf{m};\rho) = e(h,\rho)\ \Pi_i\ e(m_i,g_i) = e(h,g)^{R+\langle M,G\rangle}$,
    where $m_i = h^{M_i}$
  - Information-theoretically hiding (can use R=0 if hiding not required)
  - Binding from an analog of Discrete Log assumption, in turn implied by DDH in $\mathbb{G}_2$ [Exercise]
  - **Notation change**: Will use additive notation for the groups (exponentiation replaced with multiplication by elements in $\mathbb{F}_p$).
    For $\mathbf{a} \in \mathbb{G}_1^n$ and $\mathbf{b} \in \mathbb{G}_2^n$ let $\langle \mathbf{a},\mathbf{b} \rangle = \Pi_i\ e(a_i,b_i) \in \mathbb{G}_t$

Dory

- To verify knowledge of $\mathbf{x} \in \mathbb{G}_1^n$ s.t. $a = \langle \mathbf{x},\mathbf{g} \rangle$, $b = \langle \mathbf{x},\mathbf{h} \rangle$, given $c = \langle \mathbf{z},\mathbf{g} \rangle$ and $d = \langle \mathbf{z},\mathbf{h} \rangle$, where **z**,**h** are setup vectors, and **g** is a dynamically determined vector (initially part of the setup)
- Plan: Reduce to proof of knowledge of $\mathbf{x}^* \in \mathbb{G}_1^{n/2}$ s.t. $a^* = \langle \mathbf{x}^*,\mathbf{g}^* \rangle$ where a^* and $\mathbf{g}^*$ are defined by random choices of the verifier
    - Base case: When n=1, **h**,**z** in the setup. Get **x**,**g** and verify a,b,c.
    - At each level of recursion, there will be fresh setup vectors **z**,**h**
    - At each level $d = \langle \mathbf{z},\mathbf{h} \rangle$ made available as a pre-processed value
    - Also pre-processed values linking the setup vectors at adjacent levels: $Z_L = \langle \mathbf{z}_L,\mathbf{h}^* \rangle$, $Z_R = \langle \mathbf{z}_L,\mathbf{h}^* \rangle$, $H_L = \langle \mathbf{z}^*,\mathbf{h}_L \rangle$, $H_R = \langle \mathbf{z}^*,\mathbf{h}_R \rangle$
        - (Z_L,Z_R) work as a commitment of **z** w.r.t. $\mathbf{h}^*$. Similarly H_L,H_R
        - To change any, need to change all. But at the lowest level (n=1) z,h will be given in the clear

Dory

- Verifier holding $a = \langle x, g \rangle$, $b = \langle x, h \rangle$, $c = \langle z, g \rangle$. Also pre-processed values: $d = \langle z, h \rangle$, $Z_L = \langle z_L, h^* \rangle$, $Z_R = \langle z_L, h^* \rangle$, $H_L = \langle z^*, h_L \rangle$, $H_R = \langle z^*, h_R \rangle$
- To reduce verifying knowledge of $x \in \mathbb{G}_1^n$ to knowledge of $x^* \in \mathbb{G}_1^{n/2}$
- Prover sends $u_L = \langle x_L, h^* \rangle$, $u_R = \langle x_R, h^* \rangle$, $v_L = \langle z^*, g_L \rangle$, $v_R = \langle z^*, g_R \rangle$
- Verifier sends $\beta \leftarrow \mathbb{F}_p$. Let $x' = x + \beta z$, and $g' = g + \beta^{-1} h$
- Prover sends $w_L = \langle x'_L, g'_R \rangle$ and $w_R = \langle x'_R, g'_L \rangle$
- Verifier sends $\alpha \leftarrow \mathbb{F}_p$. Let $x^* = \alpha x'_L + \alpha^{-1} x'_R$, and $g^* = \alpha^{-1} g'_L + \alpha g'_R$
- Verifier computes $a^* = \langle x^*, g^* \rangle$, $b^* = \langle x^*, h^* \rangle$, $c^* = \langle z^*, g^* \rangle$ as:
  - $a^* = \langle x', g' \rangle + \alpha^2 w_L + \alpha^{-2} w_R$

    $= \langle x, g \rangle + \beta^{-1} \langle x, h \rangle + \beta \langle z, g \rangle + \langle z, h \rangle + \alpha^2 w_L + \alpha^{-2} w_R$

    $= a + \beta^{-1} b + \beta c + d + \alpha^2 w_L + \alpha^{-2} w_R$
  - $b^* = \alpha \langle x_L + \beta z_L, h^* \rangle + \alpha^{-1} \langle x_R + \beta z_R, h^* \rangle = \alpha u_L + \alpha \beta Z_L + \alpha^{-1} u_R + \alpha^{-1} \beta Z_R$
  - $c^* = \alpha \langle z^*, g_L + \beta^{-1} h_L \rangle + \alpha^{-1} \langle z^*, g_R + \beta^{-1} h_R \rangle = \alpha v_L + \alpha \beta^{-1} H_L + \alpha^{-1} v_R + \alpha^{-1} \beta^{-1} H_R$

Dory

- Can be extended to a proof of knowledge of $\mathbf{x} \in \mathbb{G}_1^n$ s.t. $a = \langle \mathbf{x}, \mathbf{g} \rangle$ and $u = \langle \mathbf{x}, \mathbf{y} \rangle$, as in the case of Bulletproofs
  - Where $\mathbf{y} = (1, r, r^2, \ldots, r^{n-1})$ for some $r \in \mathbb{F}_p$
  - In the base case of the recursion, the verifier needs to verify $\langle \mathbf{x}^{(\log n)}, \mathbf{y}^{(\log n)} \rangle$ where $\mathbf{y}^{(i+1)} = \alpha_i \mathbf{y}_L^{(i)} + \alpha_i^{-1} \mathbf{y}_R^{(i)}$ (with $\mathbf{y}^{(0)} = \mathbf{y}$)
  - Note: $y^{(0)}_j = r^j$. $y^{(i+1)}_j = \alpha_i\ y^{(i)}_j + \alpha_i^{-1}\ y^{(i)}_{j+n/2^\wedge(i+1)}$
  - Inductively for k>0, $y^{(k)}_j = r^j\ \Pi_{i=0 \text{ to } k-1}\ (\alpha_i + \alpha_i^{-1}\ r^{n/2^\wedge(i+1)})$
    - Base case: k=1: $y^{(1)}_j = \alpha_0\ r^j + \alpha_0^{-1}\ r^{j+n/2} = (\alpha_0 + \alpha_0^{-1}\ r^{n/2})\ r^j$
    - $y^{(k+1)}_j = \alpha_k\ y^{(k)}_j + \alpha_k^{-1}\ y^{(k)}_{j+n/2^\wedge(k+1)} = (\alpha_k + \alpha_k^{-1}\ r^{n/2^\wedge(k+1)})\ \ y^{(k)}_j$ ✔
  - So, $\mathbf{y}^{(\log n)}_0 = \Pi_{i=0 \text{ to } \log n-1}\ (\alpha_i + \alpha_i^{-1}\ r^{n/2^\wedge(i+1)})$, which can be computed in O(log n) time, keeping the overall verification time O(log n)