Zero Knowledge Proofs

Polynomial Commitments based on Pairing and Discrete Logarithm

Instructors: Dan Boneh, Shafi Goldwasser, Dawn Song, Justin Thaler, Yupeng Zhang
Recall: how to build an efficient SNARK?

- A polynomial commitment scheme
- A polynomial interactive oracle proof (IOP)

SNARK for general circuits
Recall: Plonk

Univariate polynomial commitment

Plonk Polynomial IOP

SNARK for general circuits
Recall: interactive proofs

Multivariate polynomial commitment

Sumcheck protocol

SNARK for general circuits
What is a polynomial commitment

Prover

\[ f(x) \in \mathcal{F} \]

Verifier

\[ \text{commit}(f) \rightarrow \text{com}_f \]

choose a family of polynomials \( \mathcal{F} \)

\[ u \]

\[ \nu, \text{ proof } \pi \]

proof that: \( f(u) = \nu \) and \( f \in \mathcal{F} \)
Definitions of polynomial commitments

- $\text{keygen}(\lambda, \mathcal{F}) \rightarrow gp$
- $\text{commit}(gp, f) \rightarrow com_f$
- $\text{eval}(gp, f, u) \rightarrow v, \pi$
- $\text{verify}(gp, com_f, u, v, \pi) \rightarrow \text{accept or reject}$

**Knowledge sound:** for every poly. time adversary $A = (A_0, A_1)$ such that $keygen(\lambda, \mathcal{F}) \rightarrow gp, A_0(gp) \rightarrow com_f, A_1(gp, u) \rightarrow v, \pi$:

$$\Pr[ V(vp, x, \pi) = \text{accept } ] = 1$$

there is an efficient extractor $E$ (that uses $A$) s.t.

$keygen(\lambda, \mathcal{F}) \rightarrow gp, A_0(gp) \rightarrow com_f, E(gp, com_f) \rightarrow f$:

$$\Pr[f(u)=v \text{ and } f(x) \in \mathcal{F}] > 1 - \epsilon \text{ (for a negligible } \epsilon)$$
Plan of this lecture

- Background
- KZG polynomial commitment and its variants
- Bulletproofs and other schemes based on discrete-log
Background
A set $\mathbb{G}$ and an operation $\ast$

1. **Closure**: For all $a, b \in \mathbb{G}$, $a \ast b \in \mathbb{G}$

2. **Associativity**: For all $a, b, c \in \mathbb{G}$, $(a \ast b) \ast c = a \ast (b \ast c)$

3. **Identity**: There exists a unique element $e \in \mathbb{G}$ s.t. for every $a \in \mathbb{G}$, $e \ast a = a \ast e = a$.

4. **Inverse**: For each $a \in \mathbb{G}$, there exists $b \in \mathbb{G}$ s.t. $a \ast b = b \ast a = e$

E.g.: integers $\{ \ldots, -2, -1, 0, 1, 2, \ldots \}$ under add $+$

positive integers mod prime $p$: $\{1, 2, \ldots, p - 1\}$ under mult $\times$

elliptic curves
Generator of a group

- An element $g$ that generates all elements in the group by taking all powers of $g$

Examples: $\mathbb{Z}_7^* = \{1, 2, 3, 4, 5, 6\}$

- $3^1 = 3; \quad 3^2 = 2; \quad 3^3 = 6; \quad 3^4 = 4; \quad 3^5 = 5; \quad 3^6 = 1; \quad \text{mod } 7$
Discrete logarithm assumption

- A group $\mathbb{G}$ has an alternative representation as the powers of the generator $g$: $\{g, g^2, g^3, \ldots, g^{p-1}\}$
- Discrete logarithm problem: given $y \in \mathbb{G}$, find $x$ s.t. $g^x = y$
- Example: Find $x$ such that $3^x = 4 \mod 7$
- Discrete-log assumption: discrete-log problem is computationally hard
Diffie-Hellman assumption

- Computational DH assumption:
  Given $\mathbb{G}$, $g$, $g^x$, $g^y$, cannot compute $g^{xy}$
Bilinear pairing

- \((p, \mathbb{G}, g, \mathbb{G}_T, e)\)
  - \(\mathbb{G}\) and \(\mathbb{G}_T\) are both multiplicative cyclic group of order \(p\), \(g\) is the generator of \(\mathbb{G}\).
    \(\mathbb{G}:\text{base group, } \mathbb{G}_T:\text{target group}\)
  - Pairing: \(e(P^x, Q^y) = e(P, Q)^{xy} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T\)
  - Example: \(e(g^x, g^y) = e(g, g)^{xy} = e(g^{xy}, g)\)

Given \(g^x\) and \(g^y\), a pairing can check that some element \(h = g^{xy}\) without knowing \(x\) and \(y\)
Example: BLS signature [Boneh–Lynn–Shacham’2001]

- Keygen: $p, \mathbb{G}, g, \mathbb{G}_T, e$
  - private key $x$, public key $g^x$

- Sign$(sk, m)$: $H(m)^x$, where $H$ is a cryptographic hash that maps the message space to $\mathbb{G}$

- Verify$(\sigma, m)$: $e(H(m), g^x) = e(\sigma, g)$
KZG polynomial commitment
The KZG poly-commit scheme  (Kate-Zaverucha-Goldberg’2010)
Polynomial commitment

Prover

\[ f(x) \in \mathcal{F} \]

Verifier

\[ \text{keygen}(\lambda, \mathcal{F}) \togp \]

\[ \text{commit}(f) \to \text{com}_f \]

\[ u \]

\[ \text{eval}(gp, f, u) \to v, \pi \]
The KZG poly-commit scheme  (Kate-Zaverucha-Goldberg’2010)

Bilinear Group $p, G, g, G_T, e$

Univariate polynomials $\mathcal{F} = \mathbb{F}_p^{(\leq d)}[X]$

$\text{keygen}(\lambda, \mathcal{F}) \rightarrow gp$:  
- Sample random $\tau \in \mathbb{F}_p$
- $gp = (g, g^\tau, g^{\tau^2}, \ldots, g^{\tau^d})$
- delete $\tau$ !! (trusted setup)
The KZG poly-commit scheme  

(Kate-Zaverucha-Goldberg’2010)

\[ gp = (g, g^\tau, g^{\tau^2}, \ldots, g^{\tau^d}) \]

\[ commit(gp, f) \rightarrow com_f : \]

- \[ f(x) = f_0 + f_1x + f_2x^2 + \cdots + f_dx^d \]
- \[ com_f = g^{f(\tau)} \]
  \[ = g^{f_0 + f_1\tau + f_2\tau^2 + \cdots + f_d\tau^d} \]
  \[ = (g)^{f_0} \cdot (g^\tau)^{f_1} \cdot (g^{\tau^2})^{f_2} \cdot \ldots \cdot (g^{\tau^d})^{f_d} \]
The KZG poly-commit scheme  

(Kate-Zaverucha-Goldberg’2010)

gp = (g, g^τ, g^{τ^2}, ..., g^{τ^d})

eval(gp, f, u) \rightarrow v, \pi :
- f(x) - f(u) = (x - u)q(x), as \( u \) is a root of \( f(x) - f(u) \)
- Compute \( q(x) \) and \( \pi = g^{q(τ)} \), using \( gp \)
The KZG poly-commit scheme  (Kate-Zaverucha-Goldberg’2010)

\[ f(x) - f(u) = (x - u)q(x) \]

Honest prover: \( \text{com}_f = g^{f(\tau)}, \pi = g^{q(\tau)}, v = f(u) \)

\[
\text{verify}(gp, \text{com}_f, u, v, \pi) :
\]

- Idea: check the equation at point \( \tau \):
  \[ g^{f(\tau) - f(u)} = g^{(\tau-u)q(\tau)} \]
- Challenge: only know \( g^{\tau-u} \) and \( g^{q(\tau)} \)
- Solution: pairing!
  \[ e(\text{com}_f / g^v, g) = e(g^{\tau-u}, \pi) \]
  \[ e(g, g)^{f(\tau) - f(u)} = e(g, g)^{(\tau-u)q(\tau)} \]
KZG polynomial commitment

Univariate polynomials of degree $\leq d$

$gp = (g, g^\tau, g^{\tau^2}, ..., g^{\tau^d})$

Prover

$\mathbf{f}(x)$

$f(x) - f(u) = (x - u)q(x)$

Verifier

$\text{com}_f = g^{f(\tau)}$

$\nu, \text{ proof } \pi = g^{q(\tau)}$

$e(\text{com}_f / g^\nu, g) = e(g^{\tau - u}, \pi)$
Soundness of the KZG scheme

q-Strong Bilinear Diffie-Hellman (q-SBDH) assumption:

Given \((p, \mathbb{G}, g, \mathbb{G}_T, e), (g, g^\tau, g^{\tau^2}, \ldots, g^{\tau^d})\), cannot compute \(e(g, g)^{1/(\tau-u)}\) for any \(u\)
Soundness of the KZG scheme

Proof by contradiction: Suppose $v^* \neq f(u)$, $\pi^*$ pass the verification

- $e(\text{com}_f / g^{v^*}, g) = e(g^\tau - u, \pi^*)$
- $e(g^{f(\tau) - v^*}, g) = e(g^\tau - u, \pi^*)$  Knowledge assumption later

$\iff e(g^{f(\tau) - f(u) + f(u) - v^*}, g) = e(g^\tau - u, \pi^*)$, define $\delta = f(u) - v^*$

$\iff e(g^{(\tau - u)q(\tau) + \delta}, g) = e(g^\tau - u, \pi^*)$

$\iff e(g, g)^{(\tau - u)q(\tau) + \delta} = e(g, \pi^*)^{\tau - u}$

$\iff e(g, g)^{\delta} = \left( e(g, \pi^*) / e(g, g)^{q(\tau)} \right)^{\tau - u}$

$\iff e(g, g)^{\delta}_{\tau - u} = e(g, \pi^*) / e(g, g)^{q(\tau)}$  breaks q-SBDH assumption!
Knowledge soundness and KoE assumption

- Why the prover knows $f$ s.t. $\text{com}_f = g^{f(\tau)}$

Knowledge of exponent assumption:

- $g, g^{\tau}, g^{\tau^2}, ..., g^{\tau^d}$
- Sample random $\alpha$, compute $g^\alpha, g^{\alpha \tau}, g^{\alpha \tau^2}, ..., g^{\alpha \tau^d}$
- $\text{com}_f = g^{f(\tau)}, \text{com}'_f = g^{\alpha f(\tau)}$
- If $e(\text{com}_f, g^\alpha) = e(\text{com}'_f, g)$, there exists an extractor $E$ that extracts $f$ s.t. $\text{com}_f = g^{f(\tau)}$
KZG with knowledge soundness

- Keygen: gp includes both $g, g^\tau, g^{\tau^2}, ..., g^{\tau^d}$ and $g^\alpha, g^{\alpha\tau}, g^{\alpha\tau^2}, ..., g^{\alpha\tau^d}$
- Commit: $com_f = g^{f(\tau)}, \quad com'_f = g^{\alpha f(\tau)}$
- Verify: additionally checks $e(com_f, g^\alpha) = e(com'_f, g)$

- Knowledge soundness proof: extract $f$ in the first step by the KoE assumption
Generic group model (GGM) [Shoup’97, Maurer’05]

- (Informal) Adversary is only given an oracle to compute the group operation.
  
  E.g., given $g, g^\tau, g^{\tau^2}, \ldots, g^{\tau^d}$, Adv can only compute their linear combinations.

- GGM can replace the KoE assumption and reduce the commitment size in KZG.

KZG polynomial commitment

Univariate polynomials of degree $\leq d$

$gp = (g, g^\tau, g^{\tau^2}, \ldots, g^{\tau^d})$

Prover

$\forall x \in \mathbb{R}$, $\mathbf{f}(x) = \mathbf{g}(\mathbf{f}(\tau))$

$u$

$\nu$, proof $\pi = g^{q(\tau)}$

Verifier

$\text{com}_f = \mathbf{g}^{\mathbf{f}(\tau)}$

$\text{com}_f$

$e(\text{com}_f / g^\nu, g) = e(g^{\tau u}, \pi)$
Properties of the KZG poly-commit

- Keygen: trusted setup!
- Commit: $O(d)$ group exponentiations, $O(1)$ commitment size
- Eval: $O(d)$ group exponentiations
  
  $q(x)$ can be computed efficiently in linear time!
- Proof size: $O(1)$, 1 group element
- Verifier time: $O(1)$, 1 pairing
A distributed generation of \( gp \) s.t. no one can reconstruct the trapdoor if at least one of the participants is honest and discards their secrets

- \( gp = \left( g^\tau, g^{\tau^2}, ..., g^{\tau^d} \right) = (g_1, g_2, ..., g_d) \)
- Sample random \( s \), update \( gp' = (g'_1, g'_2, ..., g'_d) = (g^s_1, g^{s^2}_2, ..., g^{s^d}_d) \)
  \[ = \left( g^{s\tau}, g^{(s\tau)^2}, ..., g^{(s\tau)^d} \right) \] with secret \( \tau \cdot s \)!
- Check the correctness of \( gp' \)
  1. The contributor knows \( s \) s.t. \( g'_1 = (g_1)^s \)
  2. \( gp' \) consists of consecutive powers \( e(g'_i, g'_1) = e(g'_{i+1}, g) \), and \( g'_1 \neq 1 \)

See [Nikolaenko-Ragsdale-Bonneau-Boneh’22]
Variants of KZG polynomial commitment
Multivariate poly-commit [Papamanthou-Shi-Tamassia’13]

E.g., \( f(x_1, \ldots, x_k) = x_1 x_3 + x_1 x_4 x_5 + x_7 \)

Key idea: \( f(x_1, \ldots, x_k) - f(u_1, \ldots, u_k) = \sum_{i=1}^{k} (x_i - u_i) q_i(\vec{x}) \)

- **Keygen:** sample \( \tau_1, \tau_2, \ldots, \tau_k \), compute \( gp \) as \( g \) raised to all possible monomials of \( \tau_1, \tau_2, \ldots, \tau_k \) e.g., \( 2^k \) monomials for multilinear polynomial
- **Commit:** \( com_f = g^{f(\tau_1, \tau_2, \ldots, \tau_k)} \)
- **Eval:** compute \( \pi_i = g^{q_i(\vec{\tau})} \)
- **Verify:** \( e(\text{com}_f / g^v, g) = \prod_{i=1}^{k} e(g^{\tau_i - u_i}, \pi_i) \)

\( O(\log N) \) proof size and verifier time.
Achieving zero-knowledge [ZGKPP’2018]

- See lecture 1 for the formal definition
- Plain KZG is not ZK. E.g., $com_f = g^{f(\tau)}$ is deterministic

Solution: masking with randomizers

- Commit: $com_f = g^{f(\tau)+r\eta}$
- Eval: $f(x) + ry - f(u) = (x - u)(q(x) + r'y) + y(r - r'(x - u))$
  
  $$\pi = g^{q(\tau)+r'y}, g^{r-r'(\tau-u)}$$
Prover wants to prove $f$ at $u_1, \ldots, u_m$ for $m < d$

Key idea:
- Extrapolate $f(u_1), \ldots, f(u_m)$ to get $h(x)$
- $f(x) - h(x) = \prod_{i=1}^{m}(x - u_i) q(x)$
- $\pi = g^{q(\tau)}$
- $e\left(\text{com}_f / g^{h(\tau)}, g \right) = e\left(g^{\prod_{i=1}^{m}(\tau - u_i)}, \pi \right)$
Batch opening: multiple polynomials

Prover wants to prove $f_i(u_{i,j}) = v_{i,j}$ for $i \in [n], j \in [m]$

Key idea:
- Extrapolate $f_i(u_1), ..., f(u_m)$ to get $h_i(x)$ for $i \in [n]$
- $f_i(x) - h_i(x) = \prod_{i=1}^{m} (x - u_m) q_i(x)$
- Combine all $q_i(x)$ via a random linear combination
Plonk [Gabizon-Williamson-Ciobotaru’20]

Univariate KZG

Plonk Polynomial IOP

SNARK for general circuits
vSQL [ZGKPP’17], Libra [XZZPS’19]

- Multivariate KZG
- Sumcheck protocol / GKR protocol
- SNARK for general circuits
Polynomial commitments based on discrete-log
Recall: Pros and Cons of the KZG poly-commit

✓ Commitment and proof size: $O(1)$, 1 group element
✓ Verifier time: $O(1)$ pairing

✗ Keygen: trusted setup
Bulletproofs [BCCGP’16, BBBPWM’18]

Transparent setup: sample random \( gp = (g_0, g_1, g_2, \ldots, g_d) \) in \( \mathbb{G} \)

Commit: 
\[
\begin{align*}
    f(x) &= f_0 + f_1x + f_2x^2 + \cdots + f_dx^d \\
    \text{com}_f &= g_0^{f_0} g_1^{f_1} g_2^{f_2} \cdots g_d^{f_d}
\end{align*}
\]

Pedersen vector commitment
High-level idea

\[ \text{com}_f = g_0 f_0 g_1 f_1 g_2 f_2 g_3 f_3 \]

\[ v = f_0 + f_1 u + f_2 u^2 + f_3 u^3 \]

\[ \text{com}_{f'} = g_0' f_0' g_1' f_1' \]

\[ v' = f_0' + f_1' u \]
Poly-commitment based on Bulletproofs

**Prover**

\[ gp = (g_0, g_1, g_2, g_3) \]

\[ v = f_0 + f_1 u + f_2 u^2 + f_3 u^3 \]

\[ L = g_2^f_0 g_3^f_1 \]

\[ R = g_0^f_2 g_1^f_3 \]

\[ v_L = f_0 + f_1 u \]

\[ v_R = f_2 + f_3 u \]

\[ r f_0 + f_2, r f_1 + f_3 \]

**Verifier**

\[ \text{com}_f = g_0^f_0 g_1^f_1 g_2^f_2 g_3^f_3 \]

\[ v = v_L + v_R u^2 \]

\[ \text{com}' = L^r \cdot \text{com}_f \cdot R^{r^{-1}} \]

\[ gp' = (g_0^{r^{-1}} g_2, g_1^{r^{-1}} g_3) \]

\[ v' = r v_L + v_R \]
Poly-commitment based on Bulletproofs

\[\text{com}_f = g_0^{f_0} g_1^{f_1} g_2^{f_2} g_3^{f_3}, \quad L = g_2^{f_0} g_3^{f_1} \quad R = g_0^{f_2} g_1^{f_3}\]

\[\text{com}' = L^r \cdot \text{com}_f \cdot R^{r^{-1}} = g_0^{f_0+r^{-1}f_2} g_2^{r f_0+f_2} \cdot g_1^{f_1+r^{-1}f_3} g_3^{r f_1+f_3}\]

\[= (g_0^{r^{-1}} g_2)^{r f_0+f_2} \cdot (g_1^{r^{-1}} g_3)^{r f_1+f_3}\]

\[gp' = (g_0^{r^{-1}}, g_2, g_1^{r^{-1}}, g_3)\]
Poly-commitment based on Bulletproofs

**Prover**

- \( f_0, f_1, f_2, f_3 \)
- \( rf_0 + f_2, rf_1 + f_3 \)

**Verifier**

- \( gp = (g_0, g_1, g_2, g_3) \)
- \( v = f_0 + f_1u + f_2u^2 + f_3u^3 \)
- \( L = g_2^f_0 g_3^f_1 \)
- \( R = g_0^f_2 g_1^f_3 \)
- \( v_L = f_0 + f_1u \)
- \( v_R = f_2 + f_3u \)
- \( v = v_L + v_Ru^2 \)
- \( com_f = \frac{g_0^f_0 g_1^f_1 g_2^f_2 g_3^f_3}{g_0^r g_1^r g_2^r g_3^r} \)
- \( com' = L^r \cdot com_f \cdot R^{r^{-1}} \)
- \( gp' = (g_0^{r^{-1}} g_2, g_1^{r^{-1}} g_3) \)
- \( v' = rv_L + v_R \)
Poly-commitment based on Bulletproofs

- **Eval**
  2. Receive $r$ from verifier, reduce $f$ to $f'$ of degree $\frac{d}{2}$
  3. Update the bases $gp'$

- **Verify**
  1. Check $v = v_L + v_R u^{d/2}$
  2. Generate $r$ randomly
  3. Update $com' = L^r \cdot com_f \cdot R^{r^{-1}}, gp', v' = rv_L + v_R$

Recurse log $d$ times
Properties of Bulletproofs

- Keygen: $O(d)$, transparent setup!
- Commit: $O(d)$ group exponentiations, $O(1)$ commitment size
- Eval: $O(d)$ group exponentiations
  (non-interactive via Fiat Shamir)
- Proof size: $O(\log d)$
- Verifier time: $O(d)$
Hyrax [Wahby-Tzialla-shelat-Thaler-Walfish’18]

- Improves the verifier time to $O(\sqrt{d})$ by representing the coefficients as a 2-D matrix
- Proof size: $O(\sqrt{d})$
Dory [Lee’2021]

- Improving verifier time to $O(\log d)$

- Key idea: delegating the structured verifier computation to the prover using inner pairing product arguments [BMMTV’2021]

- Also improves the prover time to $O(\sqrt{d})$ exponentiations plus $O(d)$ field operations
Dark [Bünz-Fisch-Szepieniec’20]

- Achieves $O(\log d)$ proof size and verifier time
- Group of unknown order
## Summary

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Prover</th>
<th>Proof size</th>
<th>Verifier</th>
<th>Trusted Setup</th>
<th>Crypto primitive</th>
</tr>
</thead>
<tbody>
<tr>
<td>KZG</td>
<td>$O(d)$</td>
<td>$O(1)$</td>
<td>$O(1)$</td>
<td>✓</td>
<td>pairing</td>
</tr>
<tr>
<td>Bullet-proofs</td>
<td>$O(d)$</td>
<td>$O(\log d)$</td>
<td>$O(d)$</td>
<td>✗</td>
<td>discrete-log</td>
</tr>
<tr>
<td>Hyrax</td>
<td>$O(d)$</td>
<td>$O(\sqrt{d})$</td>
<td>$O(\sqrt{d})$</td>
<td>✗</td>
<td>discrete-log</td>
</tr>
<tr>
<td>Dory</td>
<td>$O(d)$</td>
<td>$O(\log d)$</td>
<td>$O(\log d)$</td>
<td>✗</td>
<td>pairing</td>
</tr>
<tr>
<td>Dark</td>
<td>$O(d)$</td>
<td>$O(\log d)$</td>
<td>$O(\log d)$</td>
<td>✗</td>
<td>unknown order group</td>
</tr>
</tbody>
</table>
End of Lecture