Polynomial Commitments

Lecture 19

Discrete Log-based Approaches

Recall

Polynomial Commitment

- Prover wants to (succinctly) commit to a polynomial oand 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.

HVZK Proof of Knowledge

- Proof of Knowledge: If an adversary can give valid proofs (with significant probability), then there is an efficient way to extract a witness from that adversary
- A ZK Proof of knowledge of discrete log of $Y=g^y$
  - $P \rightarrow V$: $R := g^r$
    $V \rightarrow P$: x
    $P \rightarrow V$: $s := xy + r$ (modulo order of the group, p)
    V checks: $g^s = Y^x R$

  Will require p to be a prime.
  Operations in the exponents are in $\mathbb{F}_p$

  - Proof of Knowledge:
    - Firstly, $g^s = Y^x R \Rightarrow s = xy+r$, where $R = g^r$
    - If after sending R, P could respond to two different challenges x_1 and x_2 as $s_1 = x_1y + r$ and $s_2 = x_2y + r$, then can solve for y (in $\mathbb{F}_p$)
  - HVZK: simulation picks s, x first and sets $R = g^s/Y^x$

HVZK and Special Soundness

- HVZK: Simulation for honest (passively corrupt) verifier
  - e.g. in PoK of discrete log, simulator picks (x,s) first and computes R (without knowing r). Relies on verifier to pick x independent of R.
- Special soundness: If given (R,x,s) and (R,x′,s′) s.t. x≠x′ and both accepted by verifier, then can derive a valid witness
  - e.g. solve y from s=xy+r and s′=x′y+r (given x,s,x′,s′)
  - Implies soundness: for each R s.t. prover has significant probability of being able to convince, can extract y from the prover with comparable probability (using "rewinding", in a stand-alone setting)

Honest-Verifier ZK Proofs

- ZK PoK to prove equality of discrete logs for ((g,Y),(h,Z)), i.e., $Y = g^y$ and $Z = h^y$ [Chaum-Pederson]
  - P→V: $(R,W) := (g^r, h^r)$
    V→P: x
    P→V: $s := xy + r$ (modulo order of the group, p)
    V checks: $g^s = Y^x R$ and $h^s = Z^x W$

  Two parallel executions of the previous proof, with same x and s (forcing same r, y)

- Special Soundness:
  - $g^s = Y^x R$ and $h^s = Z^x W \Rightarrow s = xy+r = xy'+r'$ where $R=g^r$, $Y=g^y$ and $W=h^{r'}$, $Z=h^{y'}$
  - If two accepting transcripts (R,W,x_1,s_1) and (R,W,x_2,s_2) $(x_1 \neq x_2)$, then $s_1 = x_1y + r = x_1y' + r'$ and $s_2 = x_2y + r = x_2y' + r'$. Then can find $y = y' = (s_1-s_2)/(x_1-x_2)$ (in $\mathbb{F}_p$).
- HVZK: simulation picks x, s first and sets $R=g^s/Y^x$, $W=h^s/Z^x$

A Commitment Scheme

- Pedersen commitment: public parameters of the scheme encode a prime-order group from a family where discrete log is assumed to be hard
  - Commit(x;r) = $h^r g^x$ where g,h are generators of the group, which are also included in the public parameters
    - Hiding is information-theoretic: Writing $g=h^a$, r+ax is uniformly distributed when r is uniformly random
    - Binding is based on discrete log: Giving (x,r), (x′,r′) s.t. x≠x′ and r+ax = r′+ax′ allows for solving a=(r-r′)/(x′-x). Breaks the discrete log assumption

Not needed for (non-ZK) SNARKs. Can take r=0 (i.e., omit h)

- Vector variant: to commit to a vector $\mathbf{x} = (x_1,\ldots,x_n) \in \mathbb{F}_p^n$
  - Public params: generators $g_1,\ldots,g_n,h$ (and the group parameters)
  - Commit($x_1,..,x_n$;r) = $h^r \cdot \Pi_i g_i^{x_i}$. Hiding as before. Binding by a similar reduction but it guesses i s.t. $x_i \neq x_i'$.

A Commitment Scheme

- Pedersen commitment is homomorphic
  - Given commitments Commit(x;r) = $h^r g^x$ and Commit(x′;r′) = $h^{r'} g^{x'}$ can compute Commit(x+x′;r″) as $(h^r g^x)\ (h^{r'} g^{x'}) = h^{r+r'} g^{x+x'}$, where r″=r+r′
- In the vector variant as well
  - From Commit($x_1,..,x_n$;r)=$h^r \cdot \Pi_i\ g_i^{x_i}$ and Commit($x'_1,..,x'_n$;r′)=$h^{r'} \cdot \Pi_i\ g_i^{x'_i}$ can compute Commit($x_1+x'_1,..,x_n+x'_n$;r+r′)

Polynomial Commitment

- Will support committing to a vector $\mathbf{x} = (x_1,\ldots,x_n) \in \mathbb{F}_p^n$ and showing that for another known vector $\mathbf{y} = (y_1,\ldots,y_n) \in \mathbb{F}_p^n$, $\langle \mathbf{x},\mathbf{y} \rangle = u$
  - Enough for polynomial commitment: $P(\alpha) = \langle \mathbf{x},\mathbf{y} \rangle$, where $\mathbf{x}$ are the coefficients of the polynomial P in an appropriate basis, and $\mathbf{y}$ has the corresponding basis polynomials evaluated at α
    - For univariate polynomials in standard basis: $\mathbf{y} = (1, \alpha, \alpha^2, \ldots, \alpha^{n-1})$
- Will commit to $\mathbf{x}$ using Pedersen vector commitment. Also commit to an auxiliary vector $\mathbf{d} \in \mathbb{F}_p^n$
- To evaluate $\langle \mathbf{x},\mathbf{y} \rangle$, send $z = \langle \mathbf{x},\mathbf{y} \rangle$ and $s = \langle \mathbf{d},\mathbf{y} \rangle$
- Verifier sends $\beta \leftarrow \mathbb{F}_p$. Also computes Commit($\beta\mathbf{x}+\mathbf{d}$)
- Prover opens this commitment to $\mathbf{w}$. Verifier checks the opening and that $\langle \mathbf{w},\mathbf{y} \rangle = \beta z + s$

Short commitment

Long proof

Bulletproofs

- Goal: To reduce the proof size
  - Proof verification will still be linear time
- High level idea: Divide, combine and conquer
  - Divide: Split the vector into two vectors of half the length
  - Combine: A single problem obtained by merging the two sub-problems with a random weight
  - Conquer: Recurse

Bulletproofs

Proof of Knowledge of Commitment

- Writing $g_i = g^{G_i}$, and $\mathbf{G}=(G_1,\ldots,G_n)$, $\text{Commit}(\mathbf{x};0) = g^{\langle \mathbf{x},\mathbf{G}\rangle}$
- A short proof for knowledge of $\mathbf{x}$, given $\mathbf{G}$ and $g^{\langle \mathbf{x},\mathbf{G}\rangle}$
- Let $\mathbf{x} = \mathbf{x}_L \,||\, \mathbf{x}_R$ where $\mathbf{x}_L, \mathbf{x}_R \in \mathbb{F}_p^{n/2}$. Similarly $\mathbf{G} = \mathbf{G}_L \,||\, \mathbf{G}_R$

  Then $\langle \mathbf{x},\mathbf{G}\rangle = \langle \mathbf{x}_L,\mathbf{G}_L\rangle + \langle \mathbf{x}_R,\mathbf{G}_R\rangle$
- Idea: come up with randomly combined vectors $\mathbf{x}'$, $\mathbf{G}'$ such that verifying knowledge of $\mathbf{x}'$ given $g^{\langle \mathbf{x}',\mathbf{G}'\rangle}$ is enough to verify knowledge of $\mathbf{x}$
  - Try $\mathbf{x}' = \alpha\mathbf{x}_L + \beta\mathbf{x}_R$, $\mathbf{G}' = \beta\mathbf{G}_L + \alpha\mathbf{G}_R$ so that

    $\langle \mathbf{x}',\mathbf{G}'\rangle = \alpha\beta \langle \mathbf{x},\mathbf{G}\rangle + \alpha^2 \langle \mathbf{x}_L,\mathbf{G}_R\rangle + \beta^2 \langle \mathbf{x}_R,\mathbf{G}_L\rangle$
  - Will take $\beta = \alpha^{-1}$: $\langle \mathbf{x}',\mathbf{G}'\rangle = \langle \mathbf{x},\mathbf{G}\rangle + \alpha^2 \langle \mathbf{x}_L,\mathbf{G}_R\rangle + \alpha^{-2} \langle \mathbf{x}_R,\mathbf{G}_L\rangle$
  - Prover will send $g^{\langle \mathbf{x}_L,\mathbf{G}_R\rangle}$, $g^{\langle \mathbf{x}_R,\mathbf{G}_L\rangle}$ (before seeing α). After seeing α, they recurse on $g^{\langle \mathbf{x},\mathbf{G}\rangle}[g^{\langle \mathbf{x}_L,\mathbf{G}_R\rangle}]^{\alpha^2}\,[g^{\langle \mathbf{x}_R,\mathbf{G}_L\rangle}]^{\alpha^{-2}}$ for $g^{\langle \mathbf{x}',\mathbf{G}'\rangle}$.
  - Base case: send $\mathbf{x}$
- Special soundness: From $\mathbf{x}'$ for two values of α, can compute $\mathbf{x}_L$ and $\mathbf{x}_R$
- Turns out, extraction works recursively

Bulletproofs

Polynomial Commitment

- Recall: To support committing to a vector $\mathbf{x} = (x_1,\ldots,x_n) \in \mathbb{F}_p^n$ and showing that for another known vector $\mathbf{y} = (y_1,\ldots,y_n) \in \mathbb{F}_p^n$, $\langle \mathbf{x},\mathbf{y} \rangle = u$
- Idea: In addition to proving knowledge of $\mathbf{x}$ given $g^{\langle \mathbf{x},\mathbf{G} \rangle}$, also need to show $\langle \mathbf{x},\mathbf{y} \rangle = u$. Two parallel executions for $\langle \mathbf{x},\mathbf{G} \rangle$ and $\langle \mathbf{x},\mathbf{y} \rangle$.
  - Verifier already has $g^{\langle \mathbf{x},\mathbf{G} \rangle}$ and $\langle \mathbf{x},\mathbf{y} \rangle$
  - Prover sends $g^{\langle \mathbf{x}_L,\mathbf{G}_R \rangle}$, $g^{\langle \mathbf{x}_R,\mathbf{G}_L \rangle}$, $\langle \mathbf{x}_L,\mathbf{y}_L \rangle$, $\langle \mathbf{x}_R,\mathbf{y}_R \rangle$. Verifier sends $\alpha \leftarrow \mathbb{F}_p$
    - $\langle \mathbf{x}',\mathbf{G}' \rangle = \langle \mathbf{x},\mathbf{G} \rangle + \alpha^2 \langle \mathbf{x}_L,\mathbf{G}_L \rangle + \alpha^{-2} \langle \mathbf{x}_R,\mathbf{G}_R \rangle$

      $\langle \mathbf{x}',\mathbf{y}' \rangle = \langle \mathbf{x},\mathbf{y} \rangle + \alpha^2 \langle \mathbf{x}_L,\mathbf{y}_L \rangle + \alpha^{-2} \langle \mathbf{x}_R,\mathbf{y}_R \rangle$
    - They recurse on $g^{\langle \mathbf{x}',\mathbf{G}' \rangle}$ and $\langle \mathbf{x}',\mathbf{y}' \rangle$
  - Base case: Send $\mathbf{x}$. Verifier checks $g^{\langle \mathbf{x},\mathbf{G} \rangle}$ and $\langle \mathbf{x},\mathbf{y} \rangle$
- Note: This is not hiding, but can be upgraded to be so
- Is public coin: Can apply Fiat-Shamir