Zero Knowledge Succinct Noninteractive ARguments of Knowledge

Saravanan Vijayakumaran
sarva@ee.iitb.ac.in

Department of Electrical Engineering
Indian Institute of Technology Bombay

October 26, 2018

zkSNARKs

- Arguments
  - ZK proofs where soundness guarantee is required only against PPT provers
- Noninteractive
  - Proof consists of a single message from prover to verifier
- Succinct
  - Proof size is $\mathcal{O}(1)$
  - Requires a trusted setup to generate a common reference string
  - CRS size is linear in size of assertion being proved

Bilinear Pairings

- Let G and G_T be two cyclic groups of prime order q
- In practice, G is an elliptic curve group and G_T is subgroup of $\mathbb{F}_{p^n}$
- Let $G = \langle g \rangle$, i.e. $G = \{g^\alpha \mid \alpha \in \mathbb{Z}_q\}$
- A symmetric **pairing** is an efficient map $e : G \times G \mapsto G_T$ satisfying
  1. **Bilinearity**: $\forall \alpha, \beta \in \mathbb{Z}_q$, we have $e(g^\alpha, g^\beta) = e(g, g)^{\alpha\beta}$
  2. **Non-degeneracy**: $e(g, g)$ is not the identity in G_T
- Finding discrete logs is assumed to be difficult in both groups
- Pairings enable multiplication of secrets
- **Decisional Diffie-Hellman Problem**: Given x, y, z chosen uniformly from $\mathbb{Z}_q$ and g^x, g^y, PPT adversary has to distinguish between g^{xy} and g^z
- DDH problem is easy in G
- Computation DH problem (computing g^{xy} from g^x and g^y) can be difficult

Applications of Pairings

- Three-party Diffie Hellman key agreement
  - Three parties Alice, Bob, Carol have private-public key pairs $(a, g^a), (b, g^b), (c, g^c)$ where $G = \langle g \rangle$
  - Alice sends g^a to the other two
  - Bob sends g^b to the other two
  - Carol sends g^c to the other two
  - Each party can compute common key $K = e(g, g)^{abc} = e(g^b, g^c)^a = e(g^a, g^c)^b = e(g^a, g^b)^c$
- BLS Signature Scheme
  - Suppose $H : \{0, 1\}^* \mapsto G$ is a hash function
  - Let (x, g^x) be a private-public key pair
  - BLS signature on message m is $\sigma = (H(m))^x$
  - Verifier checks that $e(g, \sigma) = e(g^x, H(m))$

Checking Polynomial Evaluation

- Prover knows a polynomial $p(x) \in \mathbb{F}_q[x]$ of degree d
- Verifier wants to check that prover computes $g^{p(s)}$ for some randomly chosen $s \in \mathbb{F}_q$
- Verifier does not care which $p(x)$ is used but cares about the evaluation point s
- Verifier sends $g^{s^i}, i = 0, 1, 2, \ldots, d$ to prover
- If $p(x) = \sum_{i=0}^{d} p_i x^i$, prover can compute $g^{p(s)}$ as

$$g^{p(s)} = \Pi_{i=0}^{d} \left(g^{s^i} \right)^{p_i}$$

- But prover could have computed $g^{p(t)}$ for some $t \neq s$
- Verifier also sends $g^{\alpha s^i}, i = 0, 1, 2, \ldots, d$ for some randomly chosen $\alpha \in \mathbb{F}_q^*$
- Prover can now compute $g^{\alpha p(s)}$
- Anyone can check that $e(g^{\alpha}, g^{p(s)}) = e(g^{\alpha p(s)}, g)$
- But why can't the prover cheat by returning $g^{p(t)}$ and $g^{\alpha p(t)}$?

Knowledge of Exponent Assumptions

- **Knowledge of Exponent Assumption (KEA)**
  - Let G be a cyclic group of prime order p with generator g and let $\alpha \in \mathbb{Z}_p$
  - Given g, g^α, suppose a PPT adversary can output $c, \hat{c}$ such that $\hat{c} = c^\alpha$
  - The only way he can do so is by choosing some $\beta \in \mathbb{Z}_p$ and setting $c = g^\beta$ and $\hat{c} = (g^\alpha)^\beta$
- ***q*-Power Knowledge of Exponent (*q*-PKE) Assumption**
  - Let G be a cyclic group of prime order p with a pairing $e : G \times G \mapsto G_T$
  - Let $G = \langle g \rangle$ and α, s be randomly chosen from $\mathbb{Z}_p^*$
  - Given $g, g^s, g^{s^2}, \ldots, g^{s^q}, g^\alpha, g^{\alpha s}, g^{\alpha s^2}, \ldots, g^{\alpha s^q}$, suppose a PPT adversary can output $c, \hat{c}$ such that $\hat{c} = c^\alpha$
  - The only way he can do so is by choosing some $a_0, a_1, \ldots, a_q \in \mathbb{Z}_p$ and setting $c = \Pi_{i=0}^q \left(g^{s^i}\right)^{a_i}$ and $\hat{c} = \Pi_{i=0}^q \left(g^{\alpha s^i}\right)^{a_i}$
- Under the q-PKE assumption, the polynomial evaluation verifier is convinced of the polynomial evaluation point
- Prover can hide $g^{p(s)}$ by sending $g^{\beta + p(s)}, g^{\alpha(\beta + p(s))}$

Quadratic Arithmetic Programs

- For a field $\mathbb{F}$, an $\mathbb{F}$-arithmetic circuit has inputs and outputs from $\mathbb{F}$
- Gates can perform addition and multiplication

Definition

A QAP Q over a field $\mathbb{F}$ contains three sets of $m+1$ polynomials $\mathcal{V} = \{v_k(x)\}$, $\mathcal{W} = \{w_k(x)\}$, $\mathcal{Y} = \{y_k(x)\}$, for $k \in \{0, 1, \ldots, m\}$, and a target polynomial $t(x)$.

Suppose $F : \mathbb{F}^n \mapsto \mathbb{F}^{n'}$ where $N = n + n'$. We say that Q computes F if:

$(c_1, c_2, \ldots, c_N) \in \mathbb{F}^N$ is a valid assignment of F's inputs and outputs, if and only if there exist coefficients $(c_{N+1}, \ldots, c_m)$ such that $t(x)$ divides $p(x)$ where

$$p(x) = \left(v_0(x) + \sum_{k=1}^{m} c_k v_k(x)\right) \cdot \left(w_0(x) + \sum_{k=1}^{m} c_k w_k(x)\right) - \left(y_0(x) + \sum_{k=1}^{m} c_k y_k(x)\right).$$

So there must exist polynomial $h(x)$ such that $h(x)t(x) = p(x)$.

- Arithmetic circuits can be mapped to QAPs efficiently

Schwartz-Zippel Lemma

Lemma

Let $\mathbb{F}$ be any field. For any nonzero polynomial $f \in \mathbb{F}[x]$ of degree d and any finite subset S of $\mathbb{F}$,

$$\Pr\left[f(s) = 0\right] \leq \frac{d}{|S|}$$

when s is chosen uniformly from S.

- Suppose $\mathbb{F}$ is a finite field of order $\approx 2^{256}$
- If s is chosen uniformly from $\mathbb{F}$, then it is unlikely to be a root of low-degree polynomials
- Equality of polynomials can be checked by evaluating them at the same random point

Outline of zkSNARKs

- Prover wants to show he knows a valid input-output assignment for function F
- A QAP for F is derived
- Prover has to show he knows $(c_1, \ldots, c_m)$ such that $t(x)$ divides $v(x)w(x) - y(x)$
- For a random $s \in \mathbb{F}$, verifier reveals $g^{s^i}, g^{v_k(s)}, g^{w_k(s)}, g^{y_k(s)}, g^{t(s)}$
- Prover calculates $h(x)$ such that $h(x)t(x) = v(x)w(x) - y(x)$
- Prover calculates $g^{v(s)}, g^{w(s)}, g^{y(s)}, g^{h(s)}$
- Verifier checks that
$$\frac{e\left(g^{v(s)}, g^{w(s)}\right)}{e\left(g^{y(s)}, g\right)} = e\left(g^{h(s)}, g^{t(s)}\right)$$
- For zero knowledge, prover picks random $\delta_v, \delta_w, \delta_y$ in $\mathbb{F}$ and reveals $g^{\delta_v t(s)+v(s)}, g^{\delta_w t(s)+w(s)}, g^{\delta_y t(s)+y(s)}$ and an appropriate modification of $g^{h(s)}$
- Proof size is independent of circuit size (a few 100 bytes)
- Verification is of the order of milliseconds

ZCash CRS Generation in Brief

- Involves n parties who need to generate $g^s, g^{s^2}, \ldots, g^{s^d}$
- The value of s should not be made public
- Each party generates a random exponent s_i
- First party publishes $g^{s_1}, g^{s_1^2}, \ldots, g^{s_1^d}$
- Second party publishes $g^{s_1 s_2}, g^{s_1^2 s_2^2}, \ldots, g^{s_1^d s_2^d}$
- Last party publishes $g^{s_1 s_2 \cdots s_n}, \ldots, g^{s_1^d s_2^d \cdots s_n^d}$
- Desired $s = s_1 s_2 \cdots s_n$
- Only one party is required to destroy its secret s_i to keep s secret

References

- Pairing-Based Cryptographic Protocols : A Survey https://eprint.iacr.org/2004/064.pdf
- DDH and CDH Problems https://www.ee.iitb.ac.in/~sarva/courses/EE720/2018/notes/lecture-20.pdf
- Pinocchio: Nearly Practical Verifiable Computation, https://eprint.iacr.org/2013/279.pdf
- ZCash Parameter Generation Protocol https://github.com/zcash/mpc/blob/master/whitepaper.pdf
- Zerocash https://eprint.iacr.org/2014/349.pdf