Zero Knowledge Succinct Noninteractive ARguments of Knowledge

Saravanan Vijayakumaran
sarva@ee.iitb.ac.in

Department of Electrical Engineering
Indian Institute of Technology Bombay

October 15, 2019

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 p
- In practice, G is an elliptic curve group and G_T is subgroup of $\mathbb{F}^*_{r^n}$ where r is a prime
- Let $G = \langle g \rangle$, i.e. $G = \{g^\alpha \mid \alpha \in \mathbb{Z}_p\}$
- A symmetric **pairing** is a efficient map $e : G \times G \mapsto G_T$ satisfying
    1. **Bilinearity**: $\forall \alpha, \beta \in \mathbb{Z}_p$, 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

Computational Diffie-Hellman Problem

- **The CDH experiment $\mathtt{CDH}_{\mathcal{A},\mathcal{G}}(n)$:**
  1. Run $\mathcal{G}(1^n)$ to obtain (G, q, g) where G is a cyclic group of order q (with $\|q\| = n$), and a generator $g \in G$.
  2. Choose a uniform $x_1, x_2 \in \mathbb{Z}_q$ and compute $h_1 = g^{x_1}, h_2 = g^{x_2}$.
  3. $\mathcal{A}$ is given G, q, g, h_1, h_2 and it outputs $h \in \mathbb{Z}_q$.
  4. Experiment output is 1 if $h = g^{x_1 \cdot x_2}$ and 0 otherwise.
- **Definition:** We say that **the CDH problem is hard relative to $\mathcal{G}$** if for every PPT adversary $\mathcal{A}$ there is a negligible function $\mathtt{negl}$ such that

$$\Pr\left[\mathtt{CDH}_{\mathcal{A},\mathcal{G}}(n) = 1\right] \leq \mathtt{negl}(n).$$

Decisional Diffie-Hellman Problem

- **The DDH experiment $\texttt{DDH}_{\mathcal{A},\mathcal{G}}(n)$:**
  1. Run $\mathcal{G}(1^n)$ to obtain (G, q, g) where G is a cyclic group of order q (with $\|q\| = n$), and a generator $g \in G$.
  2. Choose a uniform $x, y, z \in \mathbb{Z}_q$ and compute $u = g^x, v = g^y$
  3. Choose a bit $b \xleftarrow{\$} \{0, 1\}$ and compute $w = g^{bz+(1-b)xy}$
  4. Give the triple u, v, w to the adversary $\mathcal{A}$
  5. $\mathcal{A}$ outputs a bit $b' = \mathcal{A}(G, q, g, u, v, w)$
- **Definition:** We say that **the DDH problem is hard relative to** $\mathcal{G}$ if for all PPT adversaries $\mathcal{A}$ there is a negligible function $\texttt{negl}$ such that

$$\left|\Pr\left[\mathcal{A}\left(G, q, g, g^x, g^y, g^z\right) = 1\right] - \Pr\left[\mathcal{A}\left(G, q, g, g^x, g^y, g^{xy}\right) = 1\right]\right| \leq \texttt{negl}(n)$$

- If G has a pairing, then DDH problem is easy in G

Some Exercises on Pairings

- A symmetric **pairing** is a efficient map $e : G \times G \mapsto G_T \subset F^*_{r^n}$ satisfying
  1. **Bilinearity**: $\forall \alpha, \beta \in \mathbb{Z}_p$, 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
- Reduce the following expressions
  - $e\left(g^a, g\right) e\left(g, g^b\right)$
  - $e\left(g, g^a\right) e\left(g^b, g\right)$
  - $e\left(g^a, g^{-b}\right) e(u, v) e\left(g, g\right)^c$
  - $\prod_{i=1}^{m} e\left(g, g^{a_i}\right)^{b_i}$
- Show that if $e(u, v) = 1$ then $u = 1$ or $v = 1$

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 $e(g, \sigma) = e(g^x, H(m))$

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 $\hat{c} = \Pi_{i=0}^{q} \left(g^{\alpha s^i}\right)^{a_i}$

Checking Polynomial Evaluation

- Prover knows a polynomial $p(x) \in \mathbb{F}_p[x]$ of degree d
- Verifier wants to check that prover computes $g^{p(s)}$ for some randomly chosen $s \in \mathbb{F}_p$
- Verifier does not care which $p(x)$ is used but cares about the evaluation point s
- Verifier sends $g^{s^i}, i = 0, 1, 2, \dots, d$ 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 $g^{\alpha s^i}, i = 0, 1, 2, \dots, d$ for some randomly chosen $\alpha \in \mathbb{F}_p^*$
- 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 $g^{\alpha p(t)}$?

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 $\mathbb{F}$,

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

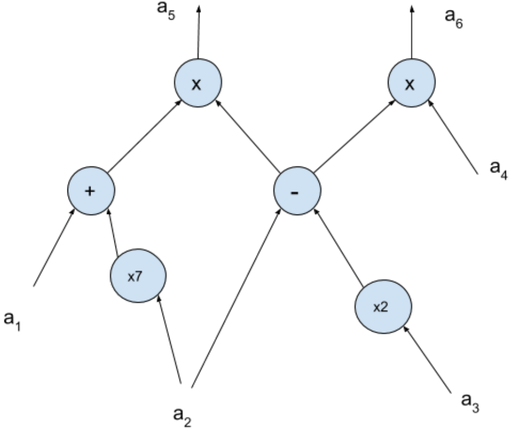
when s is chosen uniformly from S.

- 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
- **Application:** Suppose prover wants to prover that he knows a secret polynomial $p(x)$ which is divisible by another public polynomial $t(x)$
  - Verifier sends $g^{s^i}, g^{\alpha s^i}, i = 0, 1, 2, \ldots, d$ to prover
  - Prover computes $h(x) = \frac{p(x)}{t(x)} = \sum_{i=0}^{d} h_i x^i$ and calculates $g^{h(s)}$ using the coefficients h_i
  - Verifier gets $g^{p(s)}, g^{h(s)}, g^{\alpha p(s)}, g^{\alpha h(s)}$ and checks

$$e\left(g, g^{p(s)}\right) = e\left(g^{h(s)}, g^{t(s)}\right)$$

$$e\left(g^{\alpha}, g^{p(s)}\right) = e\left(g^{\alpha p(s)}, g\right), \quad e\left(g^{\alpha}, g^{h(s)}\right) = e\left(g^{\alpha h(s)}, g\right)$$

Arithmetic Circuits



Circuits consisting of additions and multiplications modulo p

Quadratic Arithmetic Programs

Definition

A QAP Q over a field $\mathbb{F}$ contains three sets of 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 $f : \mathbb{F}^n \mapsto \mathbb{F}^{n'}$ having input variables with labels $1, 2, \ldots, n$ and output variables with labels $n+1, \ldots, n+n'$. We say that Q computes f if for $N = n + n'$:

$(a_1, a_2, \ldots, a_N) \in \mathbb{F}^N$ is a valid assignment of f's inputs and outputs, if and only if there exist $(a_{N+1}, \ldots, a_m)$ such that $t(x)$ divides $p(x)$ where

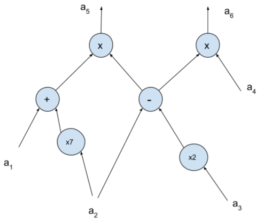
$$p(x) = \left(v_0(x) + \sum_{k=1}^{m} a_k v_k(x)\right) \cdot \left(w_0(x) + \sum_{k=1}^{m} a_k w_k(x)\right) - \left(y_0(x) + \sum_{k=1}^{m} a_k y_k(x)\right).$$

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

The size of Q is m, and the degree of Q is the degree of $t(x)$.

- Arithmetic circuits can be mapped to QAPs efficiently

QAP for an Arithmetic Circuit



- $a_5 = (a_1 + 7a_2)(a_2 - 2a_3)$ and $a_6 = (a_2 - 2a_3)a_4$
- Choose distinct $r_5, r_6 \in \mathbb{F}$ and $t(x) = (x - r_5)(x - r_6)$
- Choose polynomials $\{v_k(x)\}, \{w_k(x)\}, \{y_k(x)\}, k = 0, 1, \ldots, m$ such that

$$\sum_{k=0}^{6} a_k v_k(r_5) = a_1 + 7a_2, \quad \sum_{k=0}^{6} a_k w_k(r_5) = a_2 - 2a_3, \quad \sum_{k=0}^{6} a_k y_k(r_5) = a_5,$$

$$\sum_{k=0}^{6} a_k v_k(r_6) = a_2 - 2a_3, \quad \sum_{k=0}^{6} a_k w_k(r_6) = a_4, \quad \sum_{k=0}^{6} a_k y_k(r_6) = a_6.$$

Pinocchio SNARK from QAP

- Let $R = \{(u, wit)\} \subset \mathbb{F}^n \times \mathbb{F}^{n_1}$ be a relation where $u \in \mathbb{F}^n$ is the statement and $wit \in \mathbb{F}^{n_1}$ is the witness
- Suppose R can verified with an arithmetic circuit, i.e. there is an arithmetic function f such that $f(u) = 1$ iff there exists a wit such that $(u, wit) \in R$
- A QAP for f is derived which has $N = n + 1$ input-output variables
- Prover has to show he knows $(a_1, \ldots, a_m)$ such that $t(x)$ divides $v(x)w(x) - y(x)$ where $t(x)$ has degree d
- **Example**
    - Let $R = \left\{(u, wit) \in \{0,1\}^{256} \times \{0,1\}^{100} \mid u = \text{SHA256}(wit)\right\}$
    - The corresponding f will compute SHA256(wit) and compare it to u
    - f has $N = 256 + 1 = 257$ input-output-related variables
    - The QAP for f will have additional variables $a_{N+1}, \ldots, a_m$ corresponding to witness values and other circuit gate inputs and outputs

Pinocchio SNARK from QAP

- Let $R = \{(u, wit)\} \subset \mathbb{F}^n \times \mathbb{F}^{n_1}$ be a relation where $u \in \mathbb{F}^n$ is the statement and $wit \in \mathbb{F}^{n_1}$ is the witness
- Suppose R can verified with an arithmetic circuit, i.e. there is an arithmetic function f such that $f(u) = 1$ iff there exists a wit such that $(u, wit) \in R$
- A QAP for f is derived which has $N = n + 1$ input-output variables
- Prover has to show he knows $(a_1, \ldots, a_m)$ such that $t(x)$ divides $v(x)w(x) - y(x)$ where $t(x)$ has degree d
- **Common Reference String Generation**
  - Let $[m] = \{1, 2, \ldots, m\}$. Indices $\{1, 2, \ldots, N\}$ are for IO-related variables while $\mathcal{I}_{mid} = \{N + 1, \ldots, m\}$ are indices of non-IO-related variables
  - Choose $r_v, r_w, s, \alpha_v, \alpha_w, \alpha_y, \beta, \gamma \xleftarrow{\$} \mathbb{F}^*$ and set $r_y = r_v r_w$, $g_v = g^{r_v}$, $g_w = g^{r_w}$, and $g_y = g^{r_y}$
  - Evaluation key
    - Generate $\{g_v^{v_k(s)}\}_{k \in \mathcal{I}_{mid}}, \{g_w^{w_k(s)}\}_{k \in \mathcal{I}_{mid}}, \{g_y^{y_k(s)}\}_{k \in \mathcal{I}_{mid}}$
    - Generate $\{g_v^{\alpha_v v_k(s)}\}_{k \in \mathcal{I}_{mid}}, \{g_w^{\alpha_w w_k(s)}\}_{k \in \mathcal{I}_{mid}}, \{g_y^{\alpha_y y_k(s)}\}_{k \in \mathcal{I}_{mid}}$
    - Generate $\{g^{s^i}\}_{i \in [d]}, \left\{g_v^{\beta v_k(s)} g_w^{\beta w_k(s)} g_y^{\beta y_k(s)}\right\}_{k \in \mathcal{I}_{mid}}$
  - Verification key
    - Generate $\{g_v^{v_k(s)}\}_{k \in \{0\} \cup [N]}, \{g_w^{w_k(s)}\}_{k \in \{0\} \cup [N]}, \{g_y^{y_k(s)}\}_{k \in \{0\} \cup [N]}$
    - Generate $g^{\alpha_v}, g^{\alpha_w}, g^{\alpha_y}, g^{\gamma}, g^{\beta\gamma}, g_y^{t(s)}$

Proof Generation for Pinocchio SNARK

- Prover will prove that $(u, wit) \in R$ by showing that $f(u) = 1$
- Prover computes QAP coefficients $(a_1, \ldots, a_m)$ such that

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

- For

$$v_{mid}(x) = \sum_{k\in\mathcal{I}_{mid}} a_k v_k(x),$$
$$w_{mid}(x) = \sum_{k\in\mathcal{I}_{mid}} a_k w_k(x),$$
$$y_{mid}(x) = \sum_{k\in\mathcal{I}_{mid}} a_k y_k(x)$$

the prover outputs the proof π as

$$g_v^{v_{mid}(s)}, \quad g_w^{w_{mid}(s)}, \quad g_y^{y_{mid}(s)}, \quad g^{h(s)},$$
$$g_v^{\alpha_v v_{mid}(s)}, \quad g_w^{\alpha_w w_{mid}(s)}, \quad g_y^{\alpha_y y_{mid}(s)}$$
$$g_v^{\beta v_{mid}(s)} g_w^{\beta w_{mid}(s)} g_y^{\beta y_{mid}(s)}$$

- Verifier sees alleged proof as $g^{V_{mid}}, g^{W_{mid}}, g^{Y_{mid}}, g^{H}, g^{V'_{mid}}, g^{W'_{mid}}, g^{Y'_{mid}}$, and g^{Z}

Proof Verification for Pinocchio SNARK

- Verification key
  - $\{g_v^{v_k(s)}\}_{k \in \{0\} \cup [N]}, \{g_w^{w_k(s)}\}_{k \in \{0\} \cup [N]}, \{g_y^{y_k(s)}\}_{k \in \{0\} \cup [N]}$
  - $g^{\alpha_v}, g^{\alpha_w}, g^{\alpha_y}, g^{\gamma}, g^{\beta\gamma}, g_y^{t(s)}$
- Verifier computes $g_v^{v_{io}(s)} = \prod_{k \in [N]} \left(g_v^{v_k(s)}\right)^{a_k}$ and similarly $g_w^{w_{io}(s)}, g_y^{y_{io}(s)}$ and checks divisibility

$$e\left(g_v^{v_0(s)} g_v^{v_{io}(s)} g_v^{V_{mid}}, g_w^{w_0(s)} g_w^{w_{io}(s)} g_w^{W_{mid}}\right) = e\left(g_y^{t(s)}, g^H\right) e\left(g_y^{y_0(s)} g_y^{y_{io}(s)} g_y^{Y_{mid}}, g\right)$$

- Verifier checks the $v_{mid}(s), w_{mid}(s), y_{mid}(s)$ are the correct linear combinations by checking

$$e\left(g_v^{V'_{mid}}, g\right) = e\left(g_v^{V_{mid}}, g^{\alpha_v}\right), \quad e\left(g_w^{W'_{mid}}, g\right) = e\left(g_w^{W_{mid}}, g^{\alpha_w}\right)$$
$$e\left(g_y^{Y'_{mid}}, g\right) = e\left(g_y^{Y_{mid}}, g^{\alpha_y}\right)$$

- Verifier checks that the same variables a_i were used in all three linear combinations $v_{mid}(s), w_{mid}(s), y_{mid}(s)$ by checking

$$e\left(g^Z, g^{\gamma}\right) = e\left(g_v^{V_{mid}} g_w^{W_{mid}} g_y^{Y_{mid}}, g^{\beta\gamma}\right)$$

Converting the SNARK into a zkSNARK

- Proof π has $g_v^{v_{mid}(s)}, g_w^{w_{mid}(s)}, g_y^{y_{mid}(s)}$ which reveals information about $\{a_{N+1}, \ldots, a_m\}$ which has the witness values
- Prover chooses $\delta_v, \delta_w, \delta_y \xleftarrow{\$} \mathbb{F}^*$ and uses $v_{mid}(x) + \delta_v t(x)$ instead of $v_{mid}(x)$, $w_{mid}(x) + \delta_w t(x)$ instead of $w_{mid}(x)$, and $y_{mid}(x) + \delta_y t(x)$ instead of $y_{mid}(x)$
- Add $g_v^{t(s)}, g_w^{t(s)}, g_v^{\alpha_v t(s)}, g_w^{\alpha_w t(s)}, g_y^{\alpha_y t(s)}, g_v^{\beta t(s)}, g_w^{\beta t(s)}, g_y^{\beta t(s)}$ to the proving key
- Before adding the perturbations by $t(x)$ multplies we had

$$h(x)t(x) = (v_0(x) + v_{io}(x) + v_{mid}(x)) \cdot (w_0(x) + w_{io}(x) + w_{mid}(x)) - (y_0(x) + y_{io}(x) + y_{mid}(x)).$$

- Now we have

$$\begin{aligned} h'(x)t(x) = {} & (v_0(x) + v_{io}(x) + v_{mid}(x) + \delta_v t(x)) \cdot (w_0(x) + w_{io}(x) + w_{mid}(x) + \delta_w t(x)) \\ & - \big(y_0(x) + y_{io}(x) + y_{mid}(x) + \delta_y t(x)\big). \end{aligned}$$

- The extra terms on the right are all divisible by $t(x)$ and can be incorporated into the new proof π'

Proof Generation for Pinocchio zkSNARK

- Prover computes $h'(x)$ as

$$h'(x) = \frac{(v_0(x)+v_{io}(x)+v_{mid}(x))\cdot(w_0(x)+w_{io}(x)+w_{mid}(x)) - (y_0(x)+y_{io}(x)+y_{mid}(x))}{t(x)}$$
$$+\delta_v(w_0(x)+w_{io}(x)+w_{mid}(x))+\delta_w(v_0(x)+v_{io}(x)+v_{mid}(x))+\delta_v\delta_w t(x)-\delta_y.$$

- For

$$v^{\dagger}_{mid}(x) = \sum_{k\in\mathcal{I}_{mid}} a_k v_k(x) + \delta_v t(x),$$
$$w^{\dagger}_{mid}(x) = \sum_{k\in\mathcal{I}_{mid}} a_k w_k(x) + \delta_w t(x),$$
$$y^{\dagger}_{mid}(x) = \sum_{k\in\mathcal{I}_{mid}} a_k y_k(x) + \delta_y t(x)$$

  the prover outputs the proof π as

$$g_v^{v^{\dagger}_{mid}(s)}, \quad g_w^{w^{\dagger}_{mid}(s)}, \quad g_y^{y^{\dagger}_{mid}(s)}, \quad g^{h'(s)},$$
$$g_v^{\alpha_v v^{\dagger}_{mid}(s)}, \quad g_w^{\alpha_w w^{\dagger}_{mid}(s)}, \quad g_y^{\alpha_y y^{\dagger}_{mid}(s)}$$
$$g_v^{\beta v^{\dagger}_{mid}(s)} g_w^{\beta w^{\dagger}_{mid}(s)} g_y^{\beta y^{\dagger}_{mid}(s)}$$

- Verifier sees alleged proof as $g^{V_{mid}}, g^{W_{mid}}, g^{Y_{mid}}, g^{H}, g^{V'_{mid}}, g^{W'_{mid}}, g^{Y'_{mid}}$, and g^{Z}

Proof Verification for Pinocchio zkSNARK

- The same proof verification procedure is used

$$e\left(g_v^{v_0(s)} g_v^{v_{io}(s)} g_v^{V_{mid}}, g_w^{w_0(s)} g_w^{w_{io}(s)} g_w^{W_{mid}}\right) = e\left(g_y^{t(s)}, g^H\right) e\left(g_y^{y_0(s)} g_y^{y_{io}(s)} g_y^{Y_{mid}}, g\right)$$

$$e\left(g_v^{V'_{mid}}, g\right) = e\left(g_v^{V_{mid}}, g^{\alpha_v}\right), \quad e\left(g_w^{W'_{mid}}, g\right) = e\left(g_w^{W_{mid}}, g^{\alpha_w}\right)$$
$$e\left(g_y^{Y'_{mid}}, g\right) = e\left(g_y^{Y_{mid}}, g^{\alpha_y}\right)$$

$$e\left(g^Z, g^\gamma\right) = e\left(g_v^{V_{mid}} g_w^{W_{mid}} g_y^{Y_{mid}}, g^{\beta\gamma}\right)$$

- Since $g_v^{t(s)}, g_w^{t(s)}, g_v^{\alpha_v t(s)}, g_w^{\alpha_w t(s)}, g_y^{\alpha_y t(s)}, g_v^{\beta t(s)}, g_w^{\beta t(s)}, g_y^{\beta t(s)}$ have been added to the proving key, verifier is convinced only multiples of $t(x)$ have been added in the appropriate places
- Verifier is convinced that QAP divisibility condition still holds

Defining zkSNARKs

- Let R be a relation for an NP language L
- A **SNARG** system consists of $\Pi = (Gen, P, V)$
  - For security parameter κ, $crs \leftarrow Gen(1^\kappa)$
  - For $(u, w) \in R$, prover generates $\pi \leftarrow P(crs, u, w)$
  - If π is a valid proof, $V(crs, u, \pi) = 1$ and 0 otherwise
- **Completeness:** For all $(u, w) \in R$,

$$\Pr\left[V(crs, u, \pi) = 0 \mid crs \leftarrow Gen(1^\kappa), \pi \leftarrow P(crs, u, w)\right] = \mathrm{negl}(\kappa)$$

- **Soundness:** For all PPT provers P^*,

$$\Pr\left[V(crs, u, \pi) = 1 \wedge u \notin L \mid crs \leftarrow Gen(1^\kappa), \pi \leftarrow P^*(1^\kappa, crs, u)\right] = \mathrm{negl}(\kappa)$$

- **Succinctness:** Proof length $|\pi| = \mathrm{poly}(\kappa)\mathrm{polylog}\left(|u| + |w|\right)$
- **SNARK:** A SNARG with an extractor $\mathcal{E}$. For any statement u, we require a PPT extractor $\mathcal{E}_u$ such that for any $\pi \leftarrow P(crs, u, w)$ the witness is given by $w \leftarrow \mathcal{E}_u(crs, \pi)$.
- **zkSNARK:** A SNARK is zero-knowledge if there exists a simulator (S_1, S_2) such that S_1 outputs a simulated CRS *crs* and a trapdoor τ, S_2 takes as input *crs*, a statement u and trapdoor τ and outputs a simulated proof π. For $(u, w) \in R$,

$$\Pr\left[\pi \mid crs \leftarrow Gen(1^\kappa), \pi \leftarrow P\left(crs, u, w\right)\right] \approx$$
$$\Pr\left[\pi \mid (crs, \tau) \leftarrow S_1(1^\kappa), \pi \leftarrow S_2\left(crs, u, \tau\right)\right]$$

Simulator Construction for Pinocchio zkSNARK

- S_1 generates Pinocchio *crs* with trapdoor $\tau = (s, r_v, r_w, \alpha_v, \alpha_w, \alpha_y, \beta)$
- Pinocchio proof is of the form $g^{V_{mid}}, g^{W_{mid}}, g^{Y_{mid}}, g^{H}, g^{V'_{mid}}, g^{W'_{mid}}, g^{Y'_{mid}}$, and g^{Z}
- S_2 picks random $v(x), w(x), y(x)$ such that $t(x)$ divides $v(x) \cdot w(x) - y(x)$
- S_2 sets $v_{mid}(x) = v(x) - v_0(x) - v_{io}(x)$ and similarly for $w_{mid}(x), y_{mid}(x)$
- Using the trapdoor information, S_2 outputs the proof π as

$$
\begin{gathered}
g_v^{v_{mid}(s)}, \quad g_w^{w_{mid}(s)}, \quad g_y^{y_{mid}(s)}, \quad g^{h(s)}, \\
g_v^{\alpha_v v_{mid}(s)}, \quad g_w^{\alpha_w w_{mid}(s)}, \quad g_y^{\alpha_y y_{mid}(s)} \\
g_v^{\beta v_{mid}(s)} g_w^{\beta w_{mid}(s)} g_y^{\beta y_{mid}(s)}
\end{gathered}
$$

- The proof has the same distribution as the Pinocchio proof

ZCash CRS Generation in Brief

- Let us restrict our attention to the generation of $g^s, g^{s^2}, \dots, g^{s^d}$
- Suppose n parties will participate in the CRS generation
- The value of s should not be made public
- Each party generates a random exponent s_i
- First party publishes $g^{s_1}, g^{s_1^2}, \dots, g^{s_1^d}$
- Second party publishes $g^{s_1 s_2}, g^{s_1^2 s_2^2}, \dots, g^{s_1^d s_2^d}$
- Last party publishes $g^{s_1 s_2 \cdots s_n}, \dots, g^{s_1^d s_2^d \cdots s_n^d}$
- 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/EE720/2019/notes/lecture-21.pdf`
- Jens Groth's lecture in the 9th BIU Winter School on Cryptography
  - `https://cyber.biu.ac.il/event/the-9th-biu-winter-school-on-cryptography/`
  - NIZKs from Pairings `https://cyber.biu.ac.il/wp-content/uploads/2019/02/BarIlan2019.pdf`
  - NIZKs from Pairings
    `https://www.youtube.com/watch?v=_mAKh7LFPOU`
- Pinocchio: Nearly Practical Verifiable Computation,
  `https://eprint.iacr.org/2013/279.pdf`
- Why and How zk-SNARK Works by Maksym Petkus
  `https://arxiv.org/abs/1906.07221`
- Sections 7, 8 of *Quadratic Span Programs and Succinct NIZKs without PCPs*, GGPR13 `https://eprint.iacr.org/2012/215`