zk-SNARKS

lambda, ThunderDB
zeqing.guo@thunderdb.io

March, 2018

Homomorphic Encryption

- **Definition:** Homomorphic encryption allows computation on ciphertext which, when decrypted, matches the operations as if they had performed on the plaintext
- An example - RSA: $\mathcal{E}(x_1) \cdot \mathcal{E}(x_2) = \mathcal{E}(x_1 \cdot x_2)$, so we have $\mathcal{D}(\mathcal{E}(x_1) \cdot \mathcal{E}(x_2)) = \mathcal{D}(\mathcal{E}(x_1 \cdot x_2)) = x_1 \cdot x_2$

Zero-Knowledge Proof

- **Definition:** zero-knowledge proof is a method by which one party (*prover*) can prove to another party (*verifier*) that she knows a value x, without conveying any information apart from the fact that she knows the value x.
- zk-SNARKS are short for *succinct non-interactive arguments of knowledge*
  - Succinct: the sizes of the message are tiny
  - Non-interactive: there is no or onlu little interaction
  - Arguments: the verifier is only protected against computationallu limited provers

P and NP

- Polynomial-time programs: programs whose runtime is at most n^k for some k are called "polynomial-time programs"
- **P** is the class of problems L that have polynomial-time programs
- **NP** is the class of problems L that have a polynomial-time program V that can be used to verify a fact given a polynomially-sized so-called witness for that fact.

Quadratic Span Programs

- A **Quadratic Span Program (QSP)** consists of a set of polynomials and the task is to find a linear combination of those that is a multiple of another given polynomials.
- A QSP over a field F for inputs of length n consists of
  - polynomials $v_0, \ldots, v_m, w_0, \ldots, w_m$ over this field F,
  - a polynomial t over F (the target polynomials)
  - an injective function $f : (i,j)|1 \leq i \leq n, j \in 0, 1 \rightarrow 1, \ldots, m$

Quadratic Span Programs (cont.)

- For each binary input string u, the function f restricts the polynomials that can be used. For formally:
  - $a_k, b_k = 1$ if $k = f(i, u[i])$ for some i, ($u[i]$ is the ith bit of u)
  - $a_k, b_k = 0$ if $k = f(i, 1 - u[i])$ for some i
  - the target polynomials t divides $v_a w_b$ where $v_a = v_0 + a_1 v_1 + \ldots + a_m v_m$ and $w_b = w_0 + b_1 w_1 + \ldots + b_m w_m$
- If we use $a_1, \ldots, a_m, b_1, \ldots, b_m$ as the *NP witness*, then we can see the that it costs polynomial t for all the verifiers to check that t divides $v_a w_b$
  - This can be facilitated by the prover in providing another polynomial h such that $th = v_a w_b$, so we only need to check $t(s)h(s) - v_a(s) w_b(s) = 0$

The zkSNARKS in Detail

Background

- Ellitic curve can be simply represented like $\mathcal{E}(\S) := g^x$
- Ellitic curve with pairing function: $e(g^x, g^y) = e(g, g)^{xy}$

The zkSNARKS in Detail (cont.)

Setup

1. Verifier generates Common reference string (CRS)
   - CRS: a random and secret field element(s) chosen by verifiers
2. Verifier generates $\mathcal{E}(\int'), \mathcal{E}(\int^{\infty}), \cdots, \mathcal{E}(\int^{\lceil})$, where d is the max degree of all polynomials in the group
3. Verifier generates a random number α and generates $\mathcal{E}(\alpha\int'), \mathcal{E}(\alpha\int^{\infty}), \cdots, \mathcal{E}(\alpha\int^{\lceil})$
4. Verifier destroys s, α, β_v, β_w, γ
   - Verifier uses α to check prover evaluated the polynomial correctly by check $e(\mathcal{E}(\{(\int)), g^{\alpha}) = e(\alpha\{(\int), g)$

$$\begin{cases} e(\rceil(\{(\int)), g^{\alpha}) & = e(g, g)^{\alpha f(s)} \\ e(\alpha\{(\int), g) & = e(g, g)^{\alpha f(s)} \end{cases} \rightarrow e(\rceil(\{(\int)), g^{\alpha}) = e(\alpha\{(\int), g)$$

The zkSNARKS in Detail (cont.)

Interaction

1. Prover publishes
   - $v_0, \cdots, v_m$ and $w_0, \cdots, w_m$
   - t
2. Verifier publishes
   - $\mathcal{E}(\int'), \mathcal{E}(\int^\infty), \cdots, \mathcal{E}(\int^\lceil)$ and $\mathcal{E}(\alpha\int'), \mathcal{E}(\alpha\int^\infty), \cdots, \mathcal{E}(\alpha\int^\lceil)$
   - $\mathcal{E}(\sqcup(\int)), \mathcal{E}(\alpha\sqcup(\int))$
   - $\mathcal{E}(\sqsubseteq_\prime(\int)), \cdots, \mathcal{E}(\sqsubseteq_\Updownarrow(\int)), \mathcal{E}(\alpha\sqsubseteq_\prime(\int)), \cdots, \mathcal{E}(\alpha\sqsubseteq_\Updownarrow(\int)),$
   - $\mathcal{E}(\sqsupseteq_\prime(\int)), \cdots, \mathcal{E}(\sqsupseteq_\Updownarrow(\int)), \mathcal{E}(\alpha\sqsupseteq_\prime(\int)), \cdots, \mathcal{E}(\alpha\sqsupseteq_\Updownarrow(\int)),$
   - $\mathcal{E}(\gamma), \mathcal{E}(\beta_\sqsubseteq\gamma), \mathcal{E}(\beta_\sqsupseteq\gamma)$
   - $\mathcal{E}(\beta_\sqsubseteq\sqsubseteq_\infty(\int)), \cdots, \mathcal{E}(\beta_\sqsubseteq\sqsubseteq_\Updownarrow(\int))$
   - $\mathcal{E}(\beta_\sqsupseteq\sqsupseteq_\infty(\int)), \cdots, \mathcal{E}(\beta_\sqsupseteq\sqsupseteq_\Updownarrow(\int))$
   - $\mathcal{E}(\beta_\sqsubseteq\sqcup(\int)), (E(\beta_w t(s)))$

The zkSNARKS in Detail (cont.)

Proof

- The indices that are not restricted in $f : (i, j) | 1 \leq i \leq n, j \in 0, 1 \rightarrow 1, \cdots, m$ with $a_1, \cdots, a_m, b_1, \cdots, b_m$ are called I_{free}
- $v_{free}(x) = \sum_{k \in I_{free}} a_k v_k$
- $v_{in}(x) = \sum_{k \in (I \setminus I_{free})} a_k v_k$

Prover generates a proof for verifiers.

- $V_{free} := \mathcal{E}(\sqsubseteq_{\{\nabla\rceil\rceil}(\int)), W := \mathcal{E}(\sqsupseteq(\int)), H := \mathcal{E}(\langle(\int))$
- $V'_{free} := \mathcal{E}(\alpha\sqsubseteq_{\{\nabla\rceil\rceil}(\int)), W' := \mathcal{E}(\alpha\sqsupseteq(\int)), H' := \mathcal{E}(\alpha\langle(\int))$
- $Y := \mathcal{E}(\beta_{\sqsubseteq}\sqsubseteq_{\{\nabla\rceil\rceil}(\int) + \beta_{\sqsupseteq}\sqsupseteq(\int))$

The zkSNARKS in Detail (cont.)

Verification

Verifiers verify the proof as follows:

1. $e(V'_{free}, g) = e(V_{free}, g^{\alpha})e(W', \mathcal{E}(\infty)) = e(W, \mathcal{E}(\alpha)), e(H', \mathcal{E}(\infty)) = e(H, \mathcal{E}(\alpha))$
2. $e(\mathcal{E}(\gamma), Y) = e(\mathcal{E}(\beta_{\sqsubseteq}\gamma), V_{free})e(\mathcal{E}(\beta_w\gamma), W)$
3. $e(\mathcal{E}(v_0(s))\mathcal{E}(v_{in}(s))V_{free}, \mathcal{E}(w_0(s))W) = e(H, \mathcal{E}(t(s)))$
   - This checks that $(v_0(s) + a_1v_1(s) + \cdots + a_mv_m(s))(w_0(s) + b_1w_1(s) + \cdots + b_mw_m(s)) = h(s)t(s)$