# [name]: Efficient zero-knowledge proof with optimal prover computation

November 5, 2018

## 1 Preliminary

In this section, we will introduce some useful results and definitions.

### 1.1 Interactive Proof

Traditional proof involves two static objects: a prover $\mathcal{P}$ and a verifier $\mathcal{V}$. The prover $\mathcal{P}$ takes a statement $x$ as input and generate a string $\pi$ as a proof, then the verifier $\mathcal{V}$ checks if the statement $x$ and proof $\pi$ are correct. A interactive proof is a stronger notion of proof, it allows a prover $\mathcal{P}$ to convince a verifier $\mathcal{V}$ of the validity of some statement. The interactive proof runs in several rounds, allows the verifier to ask questions in each round based on prover's answers of previous rounds. We phrase this in term of $\mathcal{P}$ trying to convince $\mathcal{V}$ that $f(x) = 1$. The proof system is interesting iff the running time of $\mathcal{V}$ is less than the time of directly computing the function $f$.

We formalize the "interactive proof" in the following: https://www.overleaf.com/project/5bda2d6969cdac040f3

**Definition 1.** Let f be a boolean function. A pair of interactive machines $\langle \mathcal{P}, \mathcal{V} \rangle$ is an interactive proof for f with soundness $\epsilon$ if the following holds:

- **Completeness.** For every $x$ such that $f(x) = 1$ it holds that $\Pr[\langle \mathcal{P}, \mathcal{V} \rangle(x) = \mathit{accept}] = 1$.
- **$\epsilon$-Soundness.** For any $x$ with $f(x) \neq 1$ and any $\mathcal{P}^*$ it holds that $\Pr[\langle \mathcal{P}^*, \mathcal{V} \rangle = \mathit{accept}] \leq \epsilon$

### 1.2 Sum Check Protocol

The sum check problem is a fundamental problem that serves as a building block for varies applications. Informally the problem requires us to sum on a binary hypercube $(b_1, b_2, ..., b_l)$ for a given polynomial $g(x_1, x_2, ..., x_l)$. Directly compute the function requires exponential computation, Lund et al. [3] proposed a interactive proof protocol such that a computational unbounded prover $\mathcal{P}$ can convince a computational bounded verifier $\mathcal{V}$ that

$$H = \sum_{b_1 \in \{0,1\}} \sum_{b_2 \in \{0,1\}} ... \sum_{b_l \in \{0,1\}} g(b_1, b_2, ..., b_l)$$

Using this protocol, even a polynomial bounded verifier can verify the statement above. Now we formally define the problem and provide a description of the protocol.

**Definition 2.** Let $g$ be a $l$-variate polynomial $g(b_1, b_2, ..., b_l)$ over a field $\mathbb{F}$; the prover's goal is to convince that

$$H = \sum_{b_1 \in \{0,1\}} \sum_{b_2 \in \{0,1\}} ... \sum_{b_l \in \{0,1\}} g(b_1, b_2, ..., b_l)$$

**Protocol 1** (**Sum Check**). *The protocol proceeds in $l$ rounds.*

- *In the first round, the prover sends a univariate polynomial*

$$g_1(x_1) \overset{def}{=} \sum_{b_2 \in \{0,1\}} ... \sum_{b_l \in \{0,1\}} g(x_1, b_2, b_3, ..., b_l)$$

  *, the verifier checks $H = g_1(0) + g_1(1)$. Then the verifier sends a random number $r_1$ to prover, and sets $G_1 \overset{def}{=} g_1(r_1)$.*

- *In $i$-th round, where $2 \leq i \leq l-1$, the prover sends*

$$g_i(x_i) \overset{def}{=} \sum_{b_{i+1} \in \{0,1\}} \sum_{b_{i+2} \in \{0,1\}} ... \sum_{b_l \in \{0,1\}} g(r_1, r_2, ..., r_{i-1}, x_i, b_{i+1}, b_{i+2}, ..., b_l)$$

  *. Then the verifier checks $G_{i-1} = g_i(0) + g_i(1)$, and then sends a random number $r_i$ to prover. The verifier sets $G_i \overset{def}{=} g_i(r_i)$.*

- *In $l$-th round, the prover sends*

$$g_l(x_l) \overset{def}{=} g(r_1, r_2, ..., r_{l-1}, x_l)$$

  *, the verifier checks $G_{l-1} = g_l(0) + g_l(1)$. Then verifier generate a random number $r_l$ and sets $G_l \overset{def}{=} g_l(r_l)$. The verifier also compute $Answer \overset{def}{=} g(r_1, r_2, ..., r_l)$ locally. Verifier will accept iff $G_l = Answer$.*

**Definition 3** (**Multi-linear Extension**). Let $V : \{0,1\}^l \to \mathbb{F}$ be a function. A *multi-linear extension* is a unique polynomial $\tilde{V} : \mathbb{F}^l \to \mathbb{F}$ defined as:

$$\tilde{V}(x_1, x_2, ..., x_l) \overset{def}{=} \sum_{b \in \{0,1\}^l} \prod_{i=1}^{l} [((1-x_i)(1-b_i) + x_i b_i) \times V(b)]$$

where $b_i$ is $i$-th bit of b.

## 1.3 CMT Protocol

CMT Protocol (Cormode et al.) [1] is based on the work of Goldwasser et al. [2], gives us a efficient implementation of GKR protocol. We will futhur improve CMT protocol to optimal prover time and make it zero knowledge without any assumption on the circuit. In this section, we will introduce the original CMT protocol here.

Assume $C$ is a *layered arithmetic circuit* with depth $d$ over a finite field $\mathbb{F}$. The gates in $i$-th layer takes input from $i+1$-th layer and outputs to $i-1$-th layer; layer 0 is the output layer, and layer $d$ is the input layer.

The protocol proceeds layer by layer. The prover starts by sending the output value to the verifier. Then the protocol starts from the output layer. In $i$-th round both party runs the sum check protocol on the previous prover's claim, for the first round, they run check sumcheck to check the output. The sumcheck will eventually reduce the claim into the evalution to one specific point. Each point of current layer corresponds to two point of next layer. The verifier then reduce these two points into one point. And recursively check on this point of next layer. For the final input layer, the verifier checks the claim by calculating the value by himself.

#### 1.3.1 Protocol Details

## 2 Zero knowledge proof

We have learned the interactive proof in the prelim part. We would introduce the zero knowledge proof system based on the interactive proof system in this section.

**Argument Systems.** Let $R$ be an $NP$ relation. An argument system for $R$ is a protocol between computationally bounded prover $\mathcal{P}$ and a verifier $\mathcal{V}$ at the end of which $\mathcal{V}$ is convinced in the validity of a statement made by $\mathcal{P}$ of the from "there exists $w$ such that $(x; w) inR$" for some input $x$. In the sequel we focus on arguments of knowledge which have the stronger property that if the prover manages to convince the verifier of the statement's validity, then the prover must know $w$. We use $\mathcal{G}$ to represent private key $pk$ and verificaiton key $vk$ generation phase. Formally, consider the definition below.

**Definition 4.** Let $R$ be an NP relation and let $\lambda$ be a security parameter. A tuple of algorithm $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ is a zero knowledge argument fro $R$ if the following holds.

- Correctness. For every $(vk, pk)$ output by $\mathcal{G}(1^\lambda)$ and $(x, w) \in R$ we have

$$\langle \mathcal{P}(pk, w), \mathcal{V}(vk) \rangle (x) = accept$$

- Soundness. For any probabilistic polynomial time prover $\mathcal{P}^*$ there exists a probabilisitic polynomial extractore $\epsilon$ which runs on the same randomness as $\mathcal{P}^*$ such that for any $x$ it holds that

$$Pr[\langle \mathcal{P}^*(pk), \mathcal{V}(vk) \rangle (x) = accept \wedge (x, w) \in R] \leq neg(\lambda)$$

- Zero knowledge. There exists a probabilistic polynomial simulator $\mathcal{S}$ such that for any probabilistic polynomail time adversary $\mathcal{V}^*$, auxiliary input $z \in \{0,1\}^{poly(\lambda)}$ the following holds

$$\{View_{V^*}(\mathcal{P}(x, w) \leftrightarrow \mathcal{V}^*(x, z))\} \simeq_c \{\mathcal{S}^{\mathcal{V}^*}(x, z)\}$$

where $\simeq_c$ means computational inditinguishability.

## 3 Zero Knowledge Sumcheck

In $GKR$ protocol, for every layer of the circuit, the prover needs to prove one equation to verifier based on the sumcheck protocol. That means if we want to get the zero knowledge $GKR$ protocol, we should first design an efficient zero knowlege sumcheck.

For sumcheck protocol, the prover needs to prove $a = \sum\limits_{x_1, x_2, \cdots, x_n \in \{0,1\}} f(x_1, x_2, \cdots, x_n)$ to the verifier. And the verifier accepts it if and only if the verifier verifies $a = f(r_1, r_2, \cdots, r_n)$, where $r_1, r_2, \cdots, r_n$ are randomly chosen by verifier and $a$ is given by the prover, at the end of the protocol. We could use verifiable polynomial delegation to help verifier verify the value of $f(r_1, r_2, \cdots, r_n)$. So Zero knowledge sumcheck means that the verifier could learn nothing about the polynomial $f$ except for the value of $a$.

So the idea to make sumcheck protocol zero knowledge is very easy. We hope to use another sumcheck polynomial with the same degree and variables to mask $\sum\limits_{x_1,x_2,\cdots,x_n\in\{0,1\}} f(x_1,x_2,\cdots,x_n)$. In addition, we hope that the mask polynomial is as simple as possible since the time complexity of verifiable polynomial delegation is related to the number of items in the polynomial.

Consider the simplest example, where $f(x_1,x_2,\cdots,x_n)$ is a multilinear polynomial or the degree of $f$ is only one. Then we hope to use $g(x_1,x_2,\cdots,x_n)=a_0+a_1x_1+a_2x_2+\cdots+a_nx_n$ to mask $f$, where $a_0,a_1,\cdots,a_n$ is randomly chosen from $\mathbb{F}$. Suppose that $b=\sum\limits_{x_1,x_2,\cdots,x_n\in\{0,1\}} g(x_1,x_2,\cdots,x_n)$ and we would use the sumcheck protocol to verify

$$a+b=\sum_{x_1,x_2,\cdots,x_n\in\{0,1\}}[f(x_1,x_2,\cdots,x_n)+g(x_1,x_2,\cdots,x_n)]$$

.

In sumcheck protocol, we know that in the beginning, the prover sends $a+b$ to the verifier. Later on, the prover sends $g_i(x_i)$to the verifier in round $i$, where $1\le i\le n$. And the verifier verifies $g_i(0)+g_i(1)=g_{i-1}(r_{i-1})$. Finally, the verifier queries $f(r_1,r_2,\cdots,r_n)$ and verifies whether it equals to $g_n(r_n)$.

In our simple example, where $f$ and $g$ is multilinear, the prover could use $g_i(0)$ and $g_i(1)$ to replace $g_i(x_i)$. So we could construct our simulator $\mathcal{S}$. The simulator $\mathcal{S}$ given straightline access to $\mathcal{V}^*$ and the oracle access to $f$, works as follows

1. Draw a multilinear polynomial $Z_{sim}=a_0+a_1x_1+\cdots+a_nx_n$, where $a_0,a_1,\cdots,a_n$ are uniformly randomly chosen from $\mathbb{F}$.
2. Draw a multilinear polynomial $Q_{sim}=\in\mathbb{F}[x^{\le 1}_{1,2,\cdots,n}]$ uniform at random conditioned on $a=\sum\limits_{x_1,x_2,\cdots,x_n\in\{0,1\}} Z_{sim}(x_1,x_2,\cdots,x_n)$.
3. Begin simulating $\mathcal{V}^*$. The simulator sends $b=\sum\limits_{x_1,x_2,\cdots,x_n\in\{0,1\}} Z_{sim}(x_1,x_2,\cdots,x_n)$ to $\mathcal{V}^*$. Then engage in the sumcheck protocol on the claim "$\sum\limits_{x_1,x_2,\cdots,x_n\in\{0,1\}} Q_{sim}(x_1,x_2,\cdots,x_n)+Z_{sim}(x_1,x_2,\cdots,x_n)=a+b$".
4. Let $\vec{c}\in\mathbb{F}^n$ be the point chosen by $\mathcal{V}^*$ in the sumcheck protocol above. $\mathcal{V}^*$ utilizes the verifiable polynomial delegation(VPD) to verify the value of $Q_{sim}(\vec{r})+Z_{sim}(\vec{r})$.
5. Finally, output the view of the simulated $\mathcal{V}^*$.

Next, we need to prove this protocol is zero knowledge. Firstly,

## References

[1] G. Cormode, M. Mitzenmacher, and J. Thaler, *Practical verified computation with streaming interactive proofs*, in Proceedings of the 3rd Innovations in Theoretical Computer Science Conference, ITCS '12, New York, NY, USA, 2012, ACM, pp. 90–112.

[2] S. Goldwasser, Y. T. Kalai, and G. N. Rothblum, *Delegating computation: Interactive proofs for muggles*, J. ACM, 62 (2015), pp. 27:1–27:64.

[3] C. Lund, L. Fortnow, H. Karloff, and N. Nisan, *Algebraic methods for interactive proof systems*, J. ACM, 39 (1992), pp. 859–868.