zk_paper-week07

论基于配对的非交互式参数的大小

On the Size of Pairing-based Non-interactive Arguments

abstract:

- **first contribution** is a pairing-based (preprocessing) SNARK for arithmetic circuit satisfiability, which is an NP-complete language. In our SNARK we work with asymmetric pairings for higher efficiency, a proof is only 3 group elements, and verification consists of checking a single pairing product equations using 3 pairings in total. Our SNARK is zero-knowledge and does not reveal anything about the witness the prover uses to make the proof.

- **second contribution** we answer an open question: gives the first lower bound for pairing-based SNARGs

Succinct NIZK

small: We construct a NIZK argument for arithmetic circuit satisfiability where a proof consists of only 3 group elements

easy to verify: verifier just needs to compute a number of exponentiations proportional to the statement size and check a single pairing product equation

Performance comparison

- CRS size(common reference string)

Having cryptographic pairings, we are now ready to set up secure public and reusable parameters. Let us assume that we trust a single honest party to generate secrets s and α. As soon as α and all necessary powers of s with corresponding α-shifts are encrypted ($g^{\alpha}, g^{s^i}, g^{\alpha s^i}$ for i in $0, 1, \ldots, d$), the raw values must be deleted.

These parameters are usually referred to as *common reference string* or CRS. After CRS is generated any prover and any verifier can use it in order to conduct non-interactive zero-knowledge proof protocol. While non-crucial, the optimized version of CRS will include encrypted evaluation of the *target polynomial* $g^{t(s)}$.

Moreover CRS is divided into two groups (for i in $0, 1, \ldots, d$):

- Proving key[17]: $(g^{s^i}, g^{\alpha s^i})$
- Verification key: $(g^{t(s)}, g^{\alpha})$

- size of the proof
- the prover's computation
- the verifier's computation
- the number of pairing product equations used to verify a proof

Lower bounds

it is natural to ask what the minimal proof size is?

this paper proves that LIPs(**linear interactive proof**) with a linear decision procedure do not exist

Definition 2.5 (Linear Interactive Proof (LIP)). *A* **linear interactive proof** *over a finite field* $\mathbb{F}$ *is defined similarly to a standard interactive proof [GMR89], with the following differences.*

- *Each message exchanged between the prover* P_{LIP} *and the verifier* V_{LIP} *is a vector* $\mathbf{q}_i \in \mathbb{F}^m$ *over* $\mathbb{F}$*.*
- *The honest prover's strategy is linear in the sense that each of the prover's messages is computed by applying some linear function* $\Pi_i \colon \mathbb{F}^m \to \mathbb{F}^k$ *to the verifier's previous messages* $(\mathbf{q}_1, \ldots, \mathbf{q}_i)$*. This function is determined only by the input* x*, the witness* w*, and the round number* i*.*
- *Knowledge should only hold with respect to affine prover strategies* $\Pi^* = (\Pi, \boldsymbol{b})$*, where* Π *is a linear function, and* $\mathbf{b}$ *is some affine shift.*

Bilinear groups

Definition of a Bilinear Map

Let G_1, G_2, and G_t be cyclic groups of the same order.

Definition

A *bilinear map* from $G_1 \times G_2$ to G_t is a function $e : G_1 \times G_2 \to G_t$ such that for all $u \in G_1$, $v \in G_2$, $a, b \in \mathbb{Z}$,

$$e(u^a, v^b) = e(u, v)^{ab} \quad .$$

Bilinear maps are called pairings because they associate pairs of elements from G_1 and G_2 with elements in G_t. Note that this definition admits degenerate maps which map everything to the identity of G_t.

Non-interactive zero-knowledge arguments of knowledge

$(\sigma, \tau) \leftarrow \mathsf{Setup}(R)$**:** The setup produces a common reference string σ and a simulation trapdoor τ for the relation R.

$\pi \leftarrow \mathsf{Prove}(R, \sigma, \phi, w)$**:** The prover algorithm takes as input a common reference string σ and $(\phi, w) \in R$ and returns an argument π.

$0/1 \leftarrow \mathsf{Vfy}(R, \sigma, \phi, \pi)$**:** The verification algorithm takes as input a common reference string σ, a statement ϕ and an argument π and returns 0 (reject) or 1 (accept).

$\pi \leftarrow \mathsf{Sim}(R, \tau, \phi)$**:** The simulator takes as input a simulation trapdoor and statement ϕ and returns an argument π.

The focus of this paper is on preprocessing SNARKs, where the common reference string may be long.

Quadratic arithmetic programs(QAP)

Definition 11 (Quadratic Arithmetic Programs (QAP))**.** *A quadratic arithmetic program (QAP) Q over field F contains three sets of polynomials $\mathcal{V} = \{v_k(x) : k \in \{0, \ldots, m\}\}$, $\mathcal{W} = \{w_k(x) : k \in \{0, \ldots, m\}\}$, $\mathcal{Y} = \{y_k(x) : k \in \{0, \ldots, m\}\}$, and a target polynomial $t(x)$, all from $F[x]$.*

Let f be a function having input variables with labels $1, \ldots, n$ and output variables with labels $m - n' + 1, \ldots, m$. We say that Q is a QAP that computes f if the following is true: $a_1, \ldots, a_n, a_{m-n'+1}, \ldots, a_m \in F^{n+n'}$ is a valid assignment to the input/output variables of f iff there exist $(a_{n+1}, \ldots, a_{m-n'}) \in F^{m-n-n'}$ such that

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

The size of Q is m. The degree of Q is $\deg(t(x))$.

Note that we can assume that all of the $v_k(x)$'s, $w_k(x)$'s, and $y_k(x)$'s have degree at most $\deg(t(x)) - 1$, since they can all be reduced modulo $t(x)$ without affecting the divisibility check (the check whether $t(x)$ divides the expression).

For the protocols, we will also need a slightly stronger definition, of a "strong QAP", which rules out the perverse possibility that different linear combinations are used for the $v_k(x)$'s, $w_k(x)$'s, and $y_k(x)$'s.

Quadratic Programs

[GGPR – EuroCrypt 2013]

- An efficient encoding of computation
  - Lends itself well to cryptographic protocols

- Thm: Let C be an arithmetic circuit that computes F.
  There is a Quadratic Arithmetic Program (QAP)
  of size O(|C|) that computes F

  $\Rightarrow$ Can verify any poly-time (or even NP) function

- Related theorem for Boolean circuits and
  Quadratic Span Programs (QSPs)

Non-interactive arguments from linear non-interactive proofs

We will therefore define a split NILP, which is a NILP where the common reference string can be split into two parts $\sigma = (\sigma 1, \sigma 2)$ and the prover's proof can be split into two parts $\pi = (\pi 1, \pi 2)$.

disclosure-free (不被揭露)common reference string as one where the prover does not gain useful information that can help her choose a special matrix Π.

Efficiency

Efficiency. The proof size is 2 elements in $\mathbb{G}_1$ and 1 element in $\mathbb{G}_2$. The common reference string contains a description of the relation R, n elements in $\mathbb{Z}_p$, $m + 2n + 3$ elements in $\mathbb{G}_1$, and $n + 3$ elements in $\mathbb{G}_2$.

The verifier does not need to know the entire common reference string, it suffices to know

$$\sigma_V = \left(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, [1]_1, \left\{\left[\frac{\beta u_i(x) + \alpha v_i(x) + w_i(x)}{\gamma}\right]_1\right\}_{i=0}^{\ell}, [1]_2, [\gamma]_2, [\delta]_2, [\alpha\beta]_T\right).$$

The verifier's reference string only contains $\ell + 2$ elements in $\mathbb{G}_1$, 3 elements in $\mathbb{G}_2$, and 1 element in $\mathbb{G}_T$.