Bulletproofs: inner-product argument

July 24, 2025

Distributed Lab

zkdl-camp.github.io

github.com/ZKDL-Camp

Plan

Introduction

Bulletproofs: just some basic linear algebra

Bulletproofs: Motivation

- **Bulletproofs:** zero-knowledge proofs with logarithmic proof size
- No trusted setup, just some basic linear algebra
- Straightforward cryptographic assumption: discrete logarithm without bilinear pairings or other advanced assumptions
- Originally developed for efficient range proofs in confidential transactions
- Applicable for proving arithmetic circuits satisfiability (e.g., R1CS)
- Efficient polynomial commitment scheme could be derived (e.g., *IPA* polynomial commitment)
- Heart of *bulletproofs* – **inner-product argument**

Bulletproofs: Building blocks

- Zero-knowledge multiplication protocol **zk-mul**
- Inner-product argument **IPA**
- Application: *IPA* polynomial commitment scheme
- Application: range proofs
- Application: arithmetic circuits satisfiability

Preliminaries

- $\mathbb{G}$ – cyclic group of prime order p where **DLog** assumption holds
- $G, B \in \mathbb{G}$ – independent generators
- $\mathrm{G}, \mathrm{H} \in \mathbb{G}^n$ – vectors of generators with mutually unknown discrete log relations
- $\langle \mathrm{a}, \mathrm{b} \rangle = \sum_{i=1}^{n} a_i b_i$ – inner product of scalar vectors $\mathrm{a}, \mathrm{b} \in \mathbb{F}_p^n$
- $\langle \mathrm{a}, \mathrm{G} \rangle = \sum_{i=1}^{n} [a_i] G_i$ – inner product of a scalar vector $\mathrm{a} \in \mathbb{F}_p^n$ with a vector of generators $\mathrm{G} \in \mathbb{G}^n$
- $\mathrm{k}^n = (1, k, k^2, \ldots, k^{n-1})$

Zero-knowledge multiplication

Zero-knowledge multiplication

Goal: Prove knowledge of $a, b \in \mathbb{F}_p$ such that $c = ab$ without revealing a, b, c, i.e. consider the relation:

$$\mathcal{R}_{abc} = \{(\perp; c, a, b) \mid c = ab\}$$

Curious reader could argue that this is relatively simple problem as we have Σ-protocols framework, especially Chaum-Pedersen protocol for DH-triplets, e.g. we could prove slightly modified relation:

$$\mathcal{R}'_{abc} = \{(P, Q_a, Q_b, Q_c \in \mathbb{G}; a, b) \mid Q_c = [a]Q_b, Q_a = [a]P, Q_b = [b]P\}$$

Problem

Prover does not hide a, b, c values so that adversary could potentially learn them if they are values especially if they are small or have non-uniform distribution.

Multiplication of committed values

Solution

Use Pedersen commitments to bind to a, b, c values.

$$\mathcal{R}'_{abc} = \left\{ \begin{array}{l} (G, H, B, A, V; a, b, \alpha, \beta \in \mathbb{F}_p) \mid \\ V = [ab]G + [\beta]B, \\ A = [a]G + [b]H + [\alpha]B \end{array} \right\}$$

Here A is a binding Pedersen commitment to both a and b while V is a binding Pedersen commitment to their product ab.

Problem

How to prove that in zero-knowledge?

Zero-knowledge polynomial multiplication

In order to provide zero-knowledge proof of $\mathcal{R}'_{abc}$ we bring out polynomials!

$$l(x) = a + s_L x$$
$$r(x) = b + s_R x$$
$$t(x) = l(x)r(x) = ab + (s_L + s_R)bx + s_L s_R x^2$$

What we have now:

- $s_L, s_R \in \mathbb{F}_p$ are blinding factors
- $l(x)$ is a linear polynomial hiding value a
- $r(x)$ is a linear polynomial hiding value b
- $t(x)$ is a quadratic polynomial, constant term is the product ab, the product we typically want to prove knowledge of.

Zero-knowledge polynomial multiplication

Idea

If $l(x)r(x) = t(x)$ than with high probability for random $u \in \mathbb{F}_p$ we have $l(u)r(u) = t(u)$ due to *Schwartz-Zippel lemma*

Now we can build a zero-knowledge protocol **zk-mul** for proving a product of degree-one polynomials $l(x)r(x) = t(x)$:

- Prover computes and sends to $\mathcal{V}$ commitments to coefficients of $l(x), r(x), t(x)$:

$$A = [a]G + [b]H + [\alpha]B \qquad T_0 = [ab]G + [\tau_0]B$$
$$S = [s_L]G + [s_R]H + [\beta]B \qquad T_1 = [s_L + s_R]G + [\tau_1]B$$
$$T_2 = [s_L s_R]G + [\tau_2]B$$

  Here A – commitment to constant terms, S – commitment to degree-one coefficients of $l(x), r(x)$, T_i for $i = 0..2$ – commitments to coefficients of $t(x)$.

Zero-knowledge polynomial multiplication

- Verifier draws random challenge $u \in \mathbb{F}_p$ and sends it to prover
- Prover evaluates and sends to Verifier $(l_u, r_u, t_u, \alpha_u, \tau_u)$:

$$l_u = l(u), r_u = r(u), t_u = t(u) = l_u \cdot r_u,$$
$$\alpha_u = \alpha + \beta u, \tau_u = \tau_0 + \tau_1 u + \tau_2 u^2$$

- Verifier checks:

$$A + [u]S \stackrel{?}{=} [l_u]G + [r_u]H + [\alpha_u]B$$
$$[t_u]G + [\tau_u]B \stackrel{?}{=} T_0 + [u]T_1 + [u^2]T_2$$
$$t_u \stackrel{?}{=} l_u r_u$$

Zero-knowledge numbers multiplication

Now we could easily tweak our **zk-mul** protocol to prove relation

$$\mathcal{R}'_{abc} = \left\{ \begin{array}{l} (G, H, B, A, V; a, b, \alpha, \beta \in \mathbb{F}_p) \mid \\ V = [ab]G + [\beta]B, \\ A = [a]G + [b]H + [\alpha]B \end{array} \right\}$$

in zero-knowledge: just use prescribed commitment A as a commitment to constant terms of $l(x), r(x)$ and V as a commitment T_0 to constant coefficient of $t(x)$

zk-mul protocol: Security

Theorem

*Zero-knowledge polynomial multiplication protocol **zk-mul** is perfect complete, special sound and perfect honest-verifier zero-knowledge.*

Here we briefly show the completeness:

- First check:

$$A + [u]S \stackrel{?}{=} [l_u]G + [r_u]H + [\alpha_u]B$$
$$LHS = [a]G + [b]H + [\alpha]B + [us_L]G + [us_R]H + [u\beta]B$$
$$RHS = [a + us_L]G + [b + us_R]H + [\alpha + u\beta]B =$$
$$= [a]G + [b]H + [\alpha]B + [us_L]G + [us_R]H + [u\beta]B$$

As $LHS = RHS$ the check is satisfied.

zk-mul protocol: Security

- Second check:

$$[t_u]G + [\tau_u]B \stackrel{?}{=} T_0 + [u]T_1 + [u^2]T_2$$
$$LHS = [ab + t_1 u + t_2 u^2]G + [\tau_0 + \tau_1 u + \tau_2 u^2]B$$
$$RHS = [ab]G + [\tau_0]B + [u]([t_1]G + [\tau_1]B) + \\ + [u^2]([t_2]G + [\tau_2]B)$$

  As $LHS = RHS$ the check is satisfied.

- The third check is $t_u = l_u r_u$ holds by definition

Intuitively **zk-mul** protocol is also *zero-knowledge* as it is easy to simulate every step of the honest prover. *Special soundness* holds as it's easy to build an extractor similar to Okamoto's protocol extractor for extracting openings of Pedersen commitments.

zk-mul: inner-product version

We could easily generalize **zk-mul** protocol to prove $t(x) = \langle \mathrm{l}(x), \mathrm{r}(x) \rangle$ for polynomials with vector coefficients $\mathrm{l}(x), \mathrm{r}(x), \mathrm{t}(x) \in \mathbb{F}_p^n[x]$. Specifically using generalized **zk-mul** protocol we could also prove in zero-knowledge that inner-product relation holds for vectors $\mathrm{a}, \mathrm{b} \in \mathbb{F}_p^n$:

$$\mathcal{R}_{zkip} = \left\{ \begin{aligned} (\mathrm{G}, \mathrm{H}, G, B, A, V; \mathrm{a}, \mathrm{b}, \alpha, \gamma) | A = \langle \mathrm{a}, \mathrm{G} \rangle + \langle \mathrm{b}, \mathrm{H} \rangle + [\alpha]B, \\ V = [\langle \mathrm{a}, \mathrm{b} \rangle]G + [\gamma]B \end{aligned} \right\}$$

Problem

Proof size is linear in n as the prover should send evaluated vectors $\mathrm{l}_\mathrm{u}, \mathrm{r}_\mathrm{u}$. We will adress this problem in the next section: **inner-product argument**.

Inner-product argument

Motivation: Inner-product Argument

The heart of **bulletproofs** is **inner-product argument(IPA)** which allows to soundly prove inner-product relation between two vectors $\mathrm{a}, \mathrm{b} \in \mathbb{F}_p^n$:

$$\mathcal{R}_{ip} = \{(\mathrm{G}, \mathrm{H}, P, c; \mathrm{a}, \mathrm{b}) | P = \langle \mathrm{a}, \mathrm{G} \rangle + \langle \mathrm{b}, \mathrm{H} \rangle \wedge \langle \mathrm{a}, \mathrm{b} \rangle = c\}$$

Firstly, let's combine statements $P = \langle \mathrm{a}, \mathrm{G} \rangle + \langle \mathrm{b}, \mathrm{H} \rangle \wedge \langle \mathrm{a}, \mathrm{b} \rangle = c$ into a single statement by multiplying the second one by a random challenge $r \in \mathbb{F}_p$ and some orthogonal generator $B \in \mathbb{G}$, summing up:

$$\mathcal{R}'_{ip} = \{(\mathrm{G}, \mathrm{H}, Q, P'; \mathrm{a}, \mathrm{b}) | P' = \langle \mathrm{a}, \mathrm{G} \rangle + \langle \mathrm{b}, \mathrm{H} \rangle + [\langle \mathrm{a}, \mathrm{b} \rangle] Q\}$$

Where $Q = [r]B, P' = P + [cr]B = P + [c]Q$

Compression step

Assuming $n = 2^d$ define by
$\mathrm{G}_{\mathrm{lo}} = (G_1, \ldots, G_{n/2}), \mathrm{G}_{\mathrm{hi}} = (G_{n/2+1}, \ldots, G_n) \in \mathbb{G}^{n/2}$ – lower and higher halves of vector G and
$\mathrm{a}_{\mathrm{lo}} = (a_1, \ldots, a_{n/2}), \mathrm{a}_{\mathrm{hi}} = (a_{n/2+1}, \ldots, a_n) \in \mathbb{F}_p^{n/2}$ – lower and higher halves of $\mathrm{a} \in \mathbb{F}_p^n$.

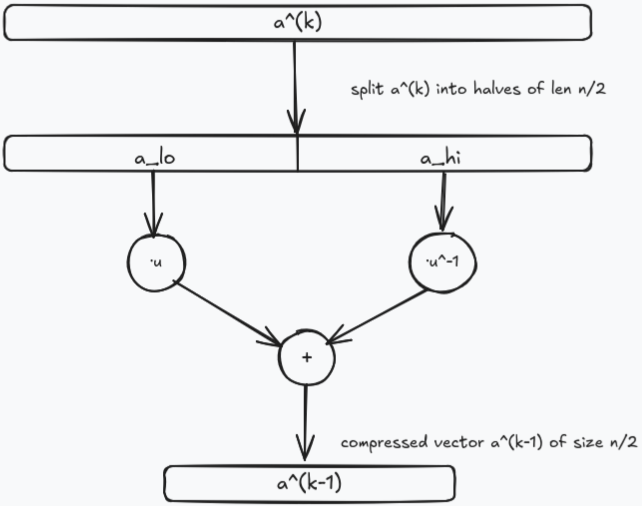
Let $u_k \in \mathbb{F}_p$ - be challenge scalar, define compressed vectors:

$$\mathrm{a}^{(k-1)} = \mathrm{a}_{\mathrm{lo}} \cdot u_k + u_k^{-1} \cdot \mathrm{a}_{\mathrm{hi}}$$
$$\mathrm{b}^{(k-1)} = \mathrm{b}_{\mathrm{lo}} \cdot u_k^{-1} + u_k \cdot \mathrm{b}_{\mathrm{hi}}$$
$$\mathrm{G}^{(k-1)} = \mathrm{G}_{\mathrm{lo}} \cdot u_k^{-1} + u_k \cdot \mathrm{G}_{\mathrm{hi}}$$
$$\mathrm{H}^{(k-1)} = \mathrm{H}_{\mathrm{lo}} \cdot u_k + u_k^{-1} \cdot \mathrm{H}_{\mathrm{hi}}$$

Note

If n is not a power of two we could pad vectors with zeroes to the next power of two.

Compression step: illustration



Commitment compression

Define $P_k \leftarrow P' = \langle \mathrm{a}, \mathrm{G} \rangle + \langle \mathrm{b}, \mathrm{H} \rangle + [\langle \mathrm{a}, \mathrm{b} \rangle] Q$ – current commitment to vectors a, b and define P_{k-1} using compressed vectors to have the same form as P_k, but in new basis $(\mathrm{G}^{(k-1)}, \mathrm{H}^{(k-1)})$:

$$P_{k-1} = \langle \mathrm{a}^{(k-1)}, \mathrm{G}^{(k-1)} \rangle + \langle \mathrm{b}^{(k-1)}, \mathrm{H}^{(k-1)} \rangle + [\langle \mathrm{a}^{(k-1)}, \mathrm{b}^{(k-1)} \rangle] Q$$

Substituting compressed vectors and applying bilinearity property of inner product we get:

$$\begin{aligned} P_{k-1} = & \langle \mathrm{a}_{\mathrm{lo}}, \mathrm{G}_{\mathrm{lo}} \rangle + \langle \mathrm{a}_{\mathrm{hi}}, \mathrm{G}_{\mathrm{hi}} \rangle & + u_k^2 \langle \mathrm{a}_{\mathrm{lo}}, \mathrm{G}_{\mathrm{hi}} \rangle + u_k^{-2} \langle \mathrm{a}_{\mathrm{hi}}, \mathrm{G}_{\mathrm{lo}} \rangle + \\ & \langle \mathrm{b}_{\mathrm{lo}}, \mathrm{H}_{\mathrm{lo}} \rangle + \langle \mathrm{b}_{\mathrm{hi}}, \mathrm{H}_{\mathrm{hi}} \rangle & + u_k^2 \langle \mathrm{b}_{\mathrm{hi}}, \mathrm{H}_{\mathrm{lo}} \rangle + u_k^{-2} \langle \mathrm{b}_{\mathrm{lo}}, \mathrm{H}_{\mathrm{hi}} \rangle + \\ & [\langle \mathrm{a}_{\mathrm{lo}}, \mathrm{b}_{\mathrm{lo}} \rangle + \langle \mathrm{a}_{\mathrm{hi}}, \mathrm{b}_{\mathrm{hi}} \rangle] Q & + [u_k^2 \langle \mathrm{a}_{\mathrm{lo}}, \mathrm{b}_{\mathrm{hi}} \rangle + u_k^{-2} \langle \mathrm{a}_{\mathrm{hi}}, \mathrm{b}_{\mathrm{lo}} \rangle] Q \end{aligned}$$

The first two columns precisely represent current commitment P_k, for the last two columns we define L_k, R_k as commitments to cross terms

Commitment compression

So that we represent new commitment P_{k-1} from the old one P_k and cross terms L_k, R_k:

$$P_{k-1} = P_k + [u_k^2]L_k + [u_k^{-2}]R_k$$
$$L_k = \langle \mathrm{a}_{\mathrm{lo}}, \mathrm{G}_{\mathrm{hi}} \rangle + \langle \mathrm{b}_{\mathrm{hi}}, \mathrm{H}_{\mathrm{lo}} \rangle + [\langle \mathrm{a}_{\mathrm{lo}}, \mathrm{b}_{\mathrm{hi}} \rangle]Q$$
$$R_k = \langle \mathrm{a}_{\mathrm{hi}}, \mathrm{G}_{\mathrm{lo}} \rangle + \langle \mathrm{b}_{\mathrm{lo}}, \mathrm{H}_{\mathrm{hi}} \rangle + [\langle \mathrm{a}_{\mathrm{hi}}, \mathrm{b}_{\mathrm{lo}} \rangle]Q$$

So the basic logic of compression step:

- Verifier draws challenge $u_k \xleftarrow{R} \mathbb{F}_p$ and sends it to prover
- Prover computes $\mathrm{a}^{(k-1)}, \mathrm{b}^{(k-1)}$ and L_k, R_k and sends them to verifier
- Verifier reconstructs P_{k-1} using $\mathrm{a}^{(k-1)}, \mathrm{b}^{(k-1)}$ and checks:

$$P_{k-1} = P_k + [u_k^2]L_k + [u_k^{-2}]R_k$$

Recursive compression

Remark

We wish not send $\mathrm{a}^{(k-1)}, \mathrm{b}^{(k-1)}$ directly as this's inefficient due to still linear sizes, instead we apply recursion to compress this vectors to just one element.

Here we come up with some kind of statement compression algorithm reducing size of all vectors in half per compression step. Repeating compression algorithm k times we end up with sending vectors $\mathrm{a}^{(0)}, \mathrm{b}^{(0)}$ each of length one and P_0 containing all accumulated cross-terms:

$$P_0 = [a_1^{(0)}]G_1^{(0)} + [b_1^{(0)}]H_1^{(0)} + [a_1^{(0)}b_1^{(0)}]Q$$

$$P_0 = P_k + \sum_{i=1}^{k}([u_i^2]L_i + [u_i^{-2}]R_i)$$

Verifier compares this P_0s and asserts inner-product correctness.

Inner-product argument protocol

Here we describe the **inner-product argument** protocol between prover $\mathcal{P}$ and verifier $\mathcal{V}$ for relation $\mathcal{R}'_{ip} = \{(\mathsf{G}, \mathsf{H}, Q, P'; \mathsf{a}, \mathsf{b}) | P' = \langle \mathsf{a}, \mathsf{G} \rangle + \langle \mathsf{b}, \mathsf{H} \rangle + [\langle \mathsf{a}, \mathsf{b} \rangle] Q\}$ from scratch.

- Prover $\mathcal{P}$ sets

$$(k, \mathsf{a}^{(k)}, \mathsf{b}^{(k)}, \mathsf{G}^{(k)}, \mathsf{H}^{(k)}, P_k) \leftarrow (d, \mathsf{a}, \mathsf{b}, \mathsf{G}, \mathsf{H}, P')$$

- Verifier $\mathcal{V}$ sets

$$(k, \mathsf{G}^{(k)}, \mathsf{H}^{(k)}, P_k) \leftarrow (d, \mathsf{G}, \mathsf{H}, P')$$

- While $k > 0$ then parties involve in **compression step protocol**

Compression step protocol

- Prover $\mathcal{P}$ computes and sends to $\mathcal{V}$

$$L_k = \langle \mathsf{a}_{\mathsf{lo}}{}^{(k)}, \mathsf{G}_{\mathsf{hi}}{}^{(k)} \rangle + \langle \mathsf{b}_{\mathsf{hi}}{}^{(k)}, \mathsf{H}_{\mathsf{lo}}{}^{(k)} \rangle + [\langle \mathsf{a}_{\mathsf{lo}}{}^{(k)}, \mathsf{b}_{\mathsf{hi}}{}^{(k)} \rangle] Q$$
$$R_k = \langle \mathsf{a}_{\mathsf{hi}}{}^{(k)}, \mathsf{G}_{\mathsf{lo}}{}^{(k)} \rangle + \langle \mathsf{b}_{\mathsf{lo}}{}^{(k)}, \mathsf{H}_{\mathsf{hi}}{}^{(k)} \rangle + [\langle \mathsf{a}_{\mathsf{hi}}{}^{(k)}, \mathsf{b}_{\mathsf{lo}}{}^{(k)} \rangle] Q$$

- $\mathcal{V}$ draws challenge $u_k \overset{R}{\leftarrow} \mathbb{F}_p$ and sends it to $\mathcal{P}$
- Both $\mathcal{P}$ and $\mathcal{V}$ compute:

$$\mathsf{G}^{(k-1)} = \mathsf{G}_{\mathsf{lo}}{}^{(k)} \cdot u_k^{-1} + u_k \cdot \mathsf{G}_{\mathsf{hi}}{}^{(k)}$$
$$\mathsf{H}^{(k-1)} = \mathsf{H}_{\mathsf{lo}}{}^{(k)} \cdot u_k + u_k^{-1} \cdot \mathsf{H}_{\mathsf{hi}}{}^{(k)}$$

- $\mathcal{P}$ computes:

$$\mathsf{a}^{(k-1)} = \mathsf{a}_{\mathsf{lo}}{}^{(k)} \cdot u_k + u_k^{-1} \cdot \mathsf{a}_{\mathsf{hi}}{}^{(k)}$$
$$\mathsf{b}^{(k-1)} = \mathsf{b}_{\mathsf{lo}}{}^{(k)} \cdot u_k^{-1} + u_k \cdot \mathsf{b}_{\mathsf{hi}}{}^{(k)}$$

Final step

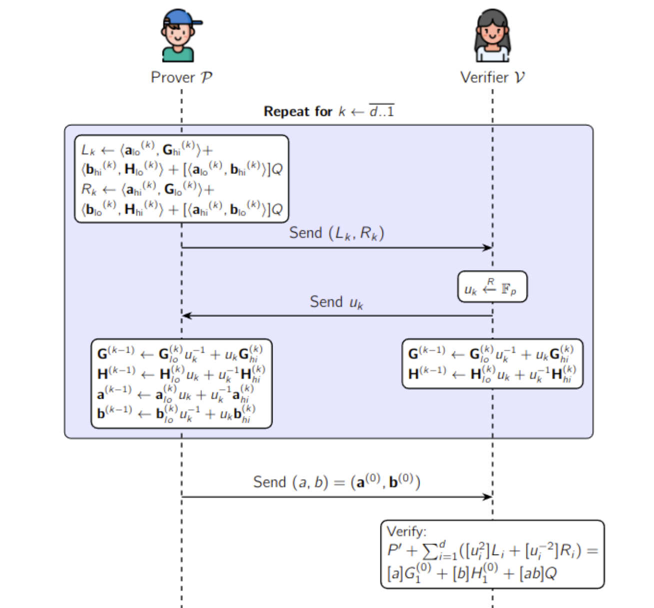
At the final step when $k = 0$ parties perform final check:

- Prover $\mathcal{P}$ sends $(a, b) \leftarrow (\mathrm{a}_1^{(0)}, \mathrm{b}_1^{(0)})$ to verifier $\mathcal{V}$
- Verifier performs final check:

$$P' + \sum_{i=1}^{d}([u_i^2]L_i + [u_i^{-2}]R_i) = [a]G_1^{(0)} + [b]H_1^{(0)} + [ab]Q$$

outputs **accept** if equality holds and **reject** otherwise.

Inner-product argument: illustration



Inner-product argument: security & performance

Remark

Overall communication complexity of **inner-product argument** is $2\log_2 n$ group elements plus 2 field elements so we come up with logarithmic proof size for our inner-product relation $\mathcal{R}_{ip}$.

Theorem (Inner-Product Argument)

*The **inner-product argument** for relation $\mathcal{R}_{ip}$ has perfect completeness and statistical witness-extended emulation for either extracting a non-trivial discrete logarithm relation between $\mathsf{G}, \mathsf{H}, Q$ or extracting valid witness a, b.*

Note

Zero-knowledge doesn't hold as if $n = 1$ then $\mathcal{P}$ sends witness pair a, b directly. We'll later compile efficient **inner-product argument** with zero-knowledge **zk-mul** protocol to achieve efficient zero-knowledge proofs for range proofs and arithmetic circuits.

What's next?


IPA
zk-mul
Range proofs &
Arithmetic circuits
@Petirep
+ JAKE-CLARK.TUMBLR
imgflip.com