Notes on Hyrax-PCS

This article briefly introduces the principles of Hyrax-PCS, whose security assumption is Discrete Log. Its main idea is to prove an Inner Product and adopts the recursive folding approach to gradually fold two vectors of length n into two vectors of length $n/2$, reducing the inner product calculation to the inner product calculation of vectors with only half the length. The recursive folding idea mainly comes from [BCC+16] and [BBB+18], with the main problem being that the Verifier's computational complexity is $O(n)$. To make Hyrax meet the requirements of zkSNARK, Hyrax rearranges the vector into a $\sqrt{n}\times\sqrt{n}$ matrix, then reduces the inner product calculation to the inner product calculation of vectors with length $\sqrt{n}$. This results in the Verifier's computational complexity being optimized to $O(\sqrt{n})$. At the same time, after optimization by [BCC+16], Hyrax's Proof size (communication complexity) is also optimized to $O(\log n)$.

1. Evaluation Proof of MLE Polynomials

Whether the MLE polynomial is in Coefficients form or Evaluation form, we can prove the evaluation of the polynomial at certain points through the "Inner Product Argument".

$$f(X_0,X_1,\ldots,X_{n-1})=\sum_{i=0}^{2^n-1}a_i\cdot\tilde{eq}(\mathsf{bits}(i),X_0,X_1,\ldots,X_{n-1})\tag{1}$$

$$f(X_0,X_1,\ldots,X_{n-1})=\sum_{i=0}^{2^n-1}c_i\cdot X_0^{i_0}\cdot X_1^{i_1}\cdot\ldots\cdot X_{n-1}^{i_{n-1}},\quad \text{where } \mathsf{bits}(i)=(i_0,i_1,\ldots,i_{n-1})\tag{2}$$

If we have an inner product proof protocol, we can easily construct an evaluation proof for MLE polynomials.

Public Input

1. Commitment of $\vec{a}$: $C_a=\mathsf{cm}(a_0,a_1,\ldots,a_{2^n-1})$
2. $\vec{u}=(u_0,u_1,\ldots,u_{n-1})$
3. $v=\tilde{f}(u_0,u_1,\ldots,u_{n-1})$

Witness

1. $\vec{a}$

Inner Product Protocol

Prover computes vector $\vec{e}$, length 2^n,

$$\begin{aligned}e_0&=\tilde{eq}(\mathsf{bits}(0),u_0,u_1,\ldots,u_{n-1})\\ e_1&=\tilde{eq}(\mathsf{bits}(1),u_0,u_1,\ldots,u_{n-1})\\ \cdots\\ e_{2^n-1}&=\tilde{eq}(\mathsf{bits}(2^n-1),u_0,u_1,\ldots,u_{n-1})\end{aligned}\tag{3}$$

Prover and Verifier use an Inner Product Argument protocol to prove that the inner product of $\vec{a}$ and $\vec{e}$ equals v. Below, we introduce a simple inner product proof that proves the inner product of two hidden vectors equals a public value.

2. Mini-IPA

Let's start with the simplest case. Suppose Prover has two vectors $\vec{a}$ and $\vec{b}$, satisfying $\langle\vec{a},\vec{b}\rangle=c$ (note: c is a *public value* here).

Proof Goal

Prover has knowledge $(\vec{a}, \vec{b})$ (two vectors of equal length, denoted as m, so there are $2m$ witnesses in total), and $\langle\vec{a}, \vec{b}\rangle = c$

Public Parameters

To compute the Pedersen Commitment of vectors, we need to select a set of random group elements $g_1, g_2, g_3, \ldots, g_m, h \in \mathbb{G}$.

Public Input

1. Inner product result c
2. Commitment of vector $\vec{a}$: $C_a = \mathsf{cm}(\vec{a}; \rho_a)$,
3. Commitment of vector $\vec{b}$: $C_b = \mathsf{cm}(\vec{b}; \rho_b)$

Witnesses

1. $(\vec{a}, \rho_a)$
2. $(\vec{b}, \rho_b)$

Basic Protocol Idea

Prover introduces two "blinder vectors", $\vec{r}$ and $\vec{s}$. These two vectors are flattened into one vector through a challenge number μ (from Verifier):

$$\vec{a}' = \vec{r} + \mu \cdot \vec{a} \qquad \vec{b}' = \vec{s} + \mu \cdot \vec{b} \tag{4}$$

Then calculate the inner product (or dot product) of $\vec{a}' \cdot \vec{b}'$.

$$\langle\vec{a}', \vec{b}'\rangle = (\vec{r} + \mu \cdot \vec{a})(\vec{s} + \mu \cdot \vec{b}) = \mu^2(\langle\vec{a}, \vec{b}\rangle) + \mu(\langle\vec{a}, \vec{s}\rangle + \langle\vec{b}, \vec{r}\rangle) + \langle\vec{r}, \vec{s}\rangle \tag{5}$$

Observing $\vec{a}'$ and $\vec{b}'$, we find that both vectors have μ terms. After flattening the vectors and performing the inner product operation, we get a **quadratic polynomial** in μ, where the "coefficient" of the μ^2 term is exactly the inner product of vectors $\vec{a}$ and $\vec{b}$ (should equal c), and the constant term $\vec{r} \cdot \vec{s}$ is exactly the inner product of the two "blinder vectors". However, the coefficient of μ looks somewhat *messy*. We can ignore the messy coefficient of μ for now and focus on the coefficient of the μ^2 term. According to the Schwartz-Zippel theorem, as long as Prover can successfully respond to Verifier's challenge, the coefficients of all terms in the polynomial must be (with high probability) correct.

We let Prover not only commit to the "blinder vectors" in the first step of the protocol but also commit to the coefficients of the polynomial (in μ) after expanding the inner product. Then in the third step, Prover only needs to send the two flattened vectors $\vec{a}'$ and $\vec{b}'$, which is just right. Verifier first verifies if a' can open A', then verifies if b' can open B', and finally Verifier verifies if $\vec{a}' \cdot \vec{b}'$ can open the commitment C'. Let's see how the protocol is specifically defined:

Protocol

Round 1

Prover sends commitments C_r and C_s of two "blinder vectors" $\vec{r}$ and $\vec{s}$; also sends polynomial coefficient commitments C_0 and C_1:

$$
\begin{aligned}
C_r &= \mathsf{cm}(\vec{r};\rho_r) \\
C_s &= \mathsf{cm}(\vec{s};\rho_s) \\
C_0 &= \mathsf{cm}(\langle\vec{r},\vec{s}\rangle;\rho_0) \\
C_1 &= \mathsf{cm}(\langle\vec{a},\vec{s}\rangle+\langle\vec{b},\vec{r}\rangle;\rho_1)
\end{aligned}
\tag{6}
$$

Round 2

1. Verifier replies with a challenge number μ
2. Prover sends two flattened vectors $\vec{a}'$, $\vec{b}'$, three random numbers mixed with μ: ρ'_a, ρ'_b, ρ'_{ab}

$$
\begin{aligned}
\vec{a}' &= \vec{r}+\mu\cdot\vec{a} \\
\vec{b}' &= \vec{s}+\mu\cdot\vec{b}
\end{aligned}
\tag{7}
$$

$$
\begin{aligned}
\rho'_a &= \rho_r+\mu\cdot\rho_a \\
\rho'_b &= \rho_s+\mu\cdot\rho_b \\
\rho'_{ab} &= \rho_0+\mu\cdot\rho_1
\end{aligned}
\tag{8}
$$

Verification

Verifier homomorphically verifies in group $\mathbb{G}$: $\vec{a}'$ and $\vec{b}'$, and their inner product

$$
\begin{aligned}
\mathsf{cm}(\vec{a}';\rho'_a) &\overset{?}{=} C_r+\mu\cdot C_a \\
\mathsf{cm}(\vec{b}';\rho'_b) &\overset{?}{=} C_s+\mu\cdot C_b
\end{aligned}
\tag{9}
$$

$$
\mathsf{cm}(\langle\vec{a}',\vec{b}'\rangle;\rho'_{ab}) \overset{?}{=} \mu^2\cdot\mathsf{cm}(c;0)+\mu\cdot C_1+C_0 \tag{10}
$$

The biggest problem with this protocol is that Verifier's computational complexity is $O(n)$, because Verifier needs to compute $\langle\vec{a}',\vec{b}'\rangle$. Also, $\vec{b}$ is hidden information, but for the MLE Evaluation proof protocol, $\vec{e}$ (computed from $\vec{u}$) is a public value, so we need to adjust the protocol.

3. Square-root inner product argument

The Hyrax paper proposes a simple and direct approach to reduce Verifier's computational complexity to $O(\sqrt{n})$. A Sublinear Verifier is a basic requirement of zkSNARK. We still only consider the Coefficients form of $\tilde{f}$, i.e., $\vec{c}$ is the coefficient of $\tilde{f}$.

We assume $n=4$, so $\vec{a}$ has a length of 16, and we can arrange this vector into a matrix:

$$
\begin{bmatrix}
c_0 & c_1 & c_2 & c_3 \\
c_4 & c_5 & c_6 & c_7 \\
c_8 & c_9 & c_{10} & c_{11} \\
c_{12} & c_{13} & c_{14} & c_{15}
\end{bmatrix}
\tag{11}
$$

The MLE polynomial represented by $\vec{a}$ can be expressed in the following form:

$$
\tilde{f}(X_0,X_1,X_2,X_3) = [1 \quad X_2 \quad X_3 \quad X_2X_3]
\begin{bmatrix}
c_0 & c_1 & c_2 & c_3 \\
c_4 & c_5 & c_6 & c_7 \\
c_8 & c_9 & c_{10} & c_{11} \\
c_{12} & c_{13} & c_{14} & c_{15}
\end{bmatrix}
\begin{bmatrix}
1 \\ X_0 \\ X_1 \\ X_0X_1
\end{bmatrix}
\tag{12}
$$

The result of this matrix calculation is as follows:

$$
\tilde{f}(X_0,X_1,X_2,X_3) = c_0+c_1X_0+c_2X_1+c_3X_2+c_4X_0X_1+\cdots+c_{14}X_1X_2X_3+c_{15}X_0X_1X_2X_3 \tag{13}
$$

We first split $\vec{u}$ into two short vectors:

$$\vec{u} = (u_0, u_1, u_2, u_3) = (u_0, u_1) \parallel (u_2, u_3) \tag{14}$$

Then $\tilde{f}(u_0, u_1, u_2, u_3)$ can be represented as:

$$\tilde{f}(u_0, u_1, u_2, u_3) = [1 \quad u_2 \quad u_3 \quad u_2u_3] \begin{bmatrix} c_0 & c_1 & c_2 & c_3 \\ c_4 & c_5 & c_6 & c_7 \\ c_8 & c_9 & c_{10} & c_{11} \\ c_{12} & c_{13} & c_{14} & c_{15} \end{bmatrix} \begin{bmatrix} 1 \\ u_0 \\ u_1 \\ u_0u_1 \end{bmatrix} \tag{15}$$

Then, we calculate the commitments of the matrix composed of $\vec{c}$ by rows, obtaining C_0, C_1, C_2, C_3,

$$\begin{aligned} C_0 &= \mathsf{cm}(c_0, c_1, c_2, c_3; \rho_0) \\ C_1 &= \mathsf{cm}(c_4, c_5, c_6, c_7; \rho_1) \\ C_2 &= \mathsf{cm}(c_8, c_9, c_{10}, c_{11}; \rho_2) \\ C_3 &= \mathsf{cm}(c_{12}, c_{13}, c_{14}, c_{15}; \rho_3) \end{aligned} \tag{16}$$

Then we can use $(1, u_2, u_3, u_2u_3)$ and (C_0, C_1, C_2, C_3) to perform an inner product operation, obtaining C^*:

$$C^* = C_0 + u_2C_1 + u_3C_2 + u_2u_3C_3 \tag{17}$$

Then C^* can be seen as the inner product of the column vector of matrix M_c and $(1, u_2, u_3, u_2u_3)$, denoted as $\vec{d} = (d_0, d_1, d_2, d_3)$,

$$\begin{aligned} d_0 &= c_0 + c_4 \cdot u_0 + c_8 \cdot u_2 + c_{12} \cdot u_2u_0 \\ d_1 &= c_1 + c_5 \cdot u_0 + c_9 \cdot u_2 + c_{13} \cdot u_2u_0 \\ d_2 &= c_2 + c_6 \cdot u_0 + c_{10} \cdot u_2 + c_{14} \cdot u_2u_0 \\ d_3 &= c_3 + c_7 \cdot u_0 + c_{11} \cdot u_2 + c_{15} \cdot u_2u_0 \end{aligned} \tag{18}$$

It's easy to verify:

$$C^* = \mathsf{cm}(d_0, d_1, d_2, d_3; \rho^*) \tag{19}$$

Here $\rho^* = \rho_0 + \rho_1u_2 + \rho_2u_3 + \rho_3u_2u_3$.

Using this approach, we construct a simple MLE polynomial commitment scheme.

Public Input

1. Commitment of $\vec{a}$: $C_a = \mathsf{cm}(a_0, a_1, \ldots, a_{2^n-1})$
2. $\vec{u} = (u_0, u_1, \ldots, u_{n-1})$
3. $v = \tilde{f}(u_0, u_1, \ldots, u_{n-1})$

Witness

1. $\vec{a}$

Commitment

1. Prover rearranges $\vec{a}$ into a matrix $M_a \in \mathbb{F}_p^{l\times h}$:

$$M_a = \begin{bmatrix} a_0 & a_1 & a_2 & \cdots & a_{l-1} \\ a_l & a_{l+1} & a_{l+2} & \cdots & a_{2l-1} \\ a_{2l} & a_{2l+1} & a_{2l+2} & \cdots & a_{3l-1} \\ a_{(h-1)l} & a_{(h-1)l+1} & a_{(h-1)l+2} & \cdots & a_{hl-1} \end{bmatrix} \tag{20}$$

2. Prover calculates commitments $C_0, C_1, \ldots, C_{h-1}$ by row

$$\begin{aligned} C_0 &= \mathsf{cm}(a_0, a_1, \ldots, a_{l-1}; \rho_0) \\ C_1 &= \mathsf{cm}(a_l, a_{l+1}, \ldots, a_{2l-1}; \rho_1) \\ C_2 &= \mathsf{cm}(a_{2l}, a_{2l+1}, \ldots, a_{3l-1}; \rho_2) \\ \cdots &= \quad \cdots \\ C_{h-1} &= \mathsf{cm}(a_{(h-1)l}, a_{(h-1)l+1}, \ldots, a_{hl-1}; \rho_{h-1}) \end{aligned} \tag{21}$$

Evaluation Proof Protocol

1. Prover and Verifier split $\vec{u}$ into two short vectors, represented by $\vec{u}_L$ and $\vec{u}_R$ respectively:

$$\begin{aligned} \vec{u}_L &= (u_0, u_1, \ldots, u_{\log(l)-1}) \\ \vec{u}_R &= (u_{\log(l)}, u_{\log(l)+1}, \ldots, u_{n-1}) \end{aligned} \tag{22}$$

Obviously

$$\vec{u} = \vec{u}_L \parallel \vec{u}_R \tag{23}$$

Round 1

1. Prover calculates $\vec{e} = (e_0, e_1, \ldots, e_{h-1})$, length h:

$$\begin{aligned} e_0 &= \tilde{eq}(\mathsf{bits}(0), \vec{u}_R) \\ e_1 &= \tilde{eq}(\mathsf{bits}(1), \vec{u}_R) \\ \cdots &= \quad \cdots \\ e_{h-1} &= \tilde{eq}(\mathsf{bits}(h-1), \vec{u}_R) \end{aligned} \tag{24}$$

2. Prover calculates the matrix multiplication of $\vec{e}$ and M_a, obtaining a new vector $\vec{b}$, length l

$$\begin{aligned} b_0 &= \langle \vec{e}, (a_0, a_l, \ldots, a_{(h-1)l}) \rangle \\ b_1 &= \langle \vec{e}, (a_1, a_{l+1}, \ldots, a_{(h-1)l+1}) \rangle \\ \cdots &= \quad \cdots \\ b_{l-1} &= \langle \vec{e}, (a_{l-1}, a_{l+l-1}, \ldots, a_{(h-1)l+l-1}) \rangle \end{aligned} \tag{25}$$

2. Prover calculates the commitment C^* of $\vec{b}$

$$C^* = \mathsf{cm}(\vec{b}; \rho^*) \tag{26}$$

Round 2.

Prover and Verifier conduct an Inner Product Argument protocol to complete the inner product proof of $\vec{b}$ and $\vec{d}$.

$$\begin{aligned} d_0 &= \tilde{eq}(\mathsf{bits}(0), \vec{u}_L) \\ d_1 &= \tilde{eq}(\mathsf{bits}(1), \vec{u}_L) \\ \cdots &= \quad \cdots \\ d_{h-1} &= \tilde{eq}(\mathsf{bits}(h-1), \vec{u}_L) \end{aligned} \tag{27}$$

1. Prover first samples a random number vector $\vec{r}$, used to protect the information of $\vec{b}$, then calculates its commitment:

$$C_r = \mathsf{cm}(\vec{r}; \rho_r) \tag{28}$$

2. Prover calculates the inner product of $\vec{r}$ and $\vec{b}$, obtaining v

$$s = \vec{r} \cdot \vec{d} \tag{29}$$

$$C_0 = \mathsf{cm}(s; \rho_s) \tag{30}$$

$$C_1 = \mathsf{cm}(0; \rho_t) \tag{31}$$

Round 3.

1. Verifier sends a random number μ
2. Prover calculates and sends the following values:

$$\vec{z_b} = \vec{r} + \mu \cdot \vec{b} \tag{32}$$

$$z_\rho = \rho_r + \mu \cdot \rho^* \tag{33}$$

$$z_t = \rho_s + \mu^{-1} \cdot \rho_t \tag{34}$$

Verification

Verifier verifies:

$$C_r + \mu \cdot C^* \overset{?}{=} \mathsf{cm}(\vec{z_b}; z_\rho) \tag{35}$$

$$C_0 + \mu^{-1} \cdot C_1 + \mu \cdot \mathsf{cm}(v; 0) \overset{?}{=} \mathsf{cm}(\langle \vec{z_b}, \vec{d} \rangle; z_t) \tag{36}$$

4. Bulletproofs Optimization

In the Round-2 of the "inner product proof" protocol (Mini-IPA) mentioned above, since Prover needs to send a vector of length l ($\vec{z_b}$), the overall communication of the protocol is $O(l)$. If the vector is quite long, the final proof size will be relatively large. J. Bootle et al. proposed a very interesting idea in the [BCC+16] paper, using a recursive method to gradually fold the proof, achieving **compression of the proof size**.

Suppose we have a vector $\vec{a} = (a_1, a_2, a_3, a_4)$ of length 4, we can split it in half into two vectors $\vec{a}_1 = (a_1, a_2)$ and $\vec{a}_2 = (a_3, a_4)$, then stack them vertically to form a matrix:

$$\begin{bmatrix} a_1 & a_2 \\ a_3 & a_4 \end{bmatrix} \tag{37}$$

Then we perform a "flattening" operation on this 2×2 matrix (with the help of a random number vector):

$$(x, x^{-1}) \begin{bmatrix} a_1 & a_2 \\ a_3 & a_4 \end{bmatrix} = (a_1 x + a_3 x^{-1}, \quad a_2 x + a_4 x^{-1}) = \vec{a}' \tag{38}$$

As shown in the above formula, we left multiply the matrix by a random number vector (x, x^{-1}), then obtain a vector $\vec{a}'$ of length 2. We can view this action as a special vertical flattening. This trick is slightly different from the vertical flattening in the previous text, not using the naive flattening vector (x^0, x^1). We call this action "folding".

We can notice that the length of the folded vector $\vec{a}'$ is only half of $\vec{a}$. Doing this recursively, by Verifier continuously sending challenge numbers and Prover repeatedly recursively folding, the vector can eventually be folded into a vector of prime length. However, if we are allowed to append some redundant values to the vector to align the vector length to 2^k, then after k folds, we can fold the vector into a number with a length of only 1. This is equivalent to performing **horizontal flattening** on a matrix.

This preliminary idea faces **the first problem**, which is that after the vector is cut in half, the commitment A of the original vector $\vec{a}$ seems unusable. So how can Verifier obtain the commitment of the folded vector after Prover performs a folding action? From another perspective, Pedersen commitment, in a sense, is also a kind of inner product, that is, the inner product of the "vector to be committed" and the "base vector":

$$\mathsf{cm}_{\vec{G}}(\vec{a}) = a_1 G_1 + a_2 G_2 + \cdots + a_m G_m \tag{39}$$

The next trick is crucial. We split the base vector $\vec{G}$ in the same way, then fold it similarly, but use a **different** "challenge vector":

$$(x^{-1}, x)\begin{bmatrix}G_1 & G_2\\ G_3 & G_4\end{bmatrix} = (G_1x^{-1} + G_3x, \quad G_2x^{-1} + G_4x) = \vec{G}' \tag{40}$$

Please note that the two challenge vectors (x, x^{-1}) and (x^{-1}, x) above look *symmetric*. The purpose of doing this is to create a special constant term. When calculating the inner product of the new vectors $\vec{a}'$ and $\vec{G}'$ together, its constant term is exactly the inner product of the original vector $\vec{a}$ and $\vec{G}$. But for the non-constant terms (coefficients of x^2 and x^{-2} terms), they are some values that look *rather messy*.

Let's expand the calculation. First, we split $\vec{a}$ into two sub-vectors $\vec{a} = (\vec{a}_1, \vec{a}_2)$, $\vec{G} = (\vec{G}_1, \vec{G}_2)$, then perform folding respectively to get $\vec{a}' = \vec{a}_1x + \vec{a}_2x^{-1}$, $\vec{G}' = \vec{G}_1x^{-1} + \vec{G}_2x$,

$$\vec{a}' \cdot \vec{G}' = (1, 1)\left(\begin{bmatrix}\vec{a}_1\cdot\vec{G}_1 & \vec{a}_1\cdot\vec{G}_2\\ \vec{a}_2\cdot\vec{G}_1 & \vec{a}_2\cdot\vec{G}_2\end{bmatrix} \circ \begin{bmatrix}1 & x^2\\ x^{-2} & 1\end{bmatrix}\right)\begin{pmatrix}1\\1\end{pmatrix} \tag{41}$$

The right side of the above formula is a polynomial in x, so we can clearly see the coefficients of x^2 and x^{-2}.

Next, let's solve the problem raised above: How to let Verifier calculate the commitment of the folded vector $\vec{a}'$:

$$A' = A + (\vec{a}_1 \cdot \vec{G}_2)\, x^2 + (\vec{a}_2 \cdot \vec{G}_1)\, x^{-2} \tag{42}$$

We let Verifier calculate $A' = \vec{a}' \cdot \vec{G}'$. Obviously, Verifier can calculate $\vec{G}'$ on their own, then Verifier can calculate A' based on $\vec{a}'$ sent by Prover.

The commitment A' is a polynomial in x, where the constant term is the commitment A of the original vector, and the coefficients of the x^2 and x^{-2} terms can be calculated and sent by Prover. What about the coefficients of the x^1 and x^{-1} terms? They happen to be *zero*. Because we cleverly used two symmetric challenge vectors to perform folding operations on $\vec{a}$ and $\vec{G}$ respectively, the coefficients of the x and x^{-1} terms are exactly canceled out, while making the constant term equal to the inner product of the original vectors.

The new commitment A' is easy to calculate $A' = A + Lx^2 + Rx^{-2}$, where $L = (\vec{a}_1 \cdot \vec{G}_2)$, $R = (\vec{a}_2 \cdot \vec{G}_1)$. The commitments L and R appear to be the result of cross inner products of the two sub-vectors. In this way, the problem is solved. When Prover needs to send a vector $\vec{v}$ of length m, Prover can choose to send the vector $\vec{v}'$ that is **folded and flattened**, which has a length of only $m/2$. Then Verifier can also verify the correctness of the original vector by verifying the opening of the folded and flattened vector.

Recursive folding also has its cost. In addition to Verifier having to calculate $\vec{G}'$ extra, Prover also has to calculate L and R extra, and they also need to add one round of interaction. We can see the complete process of recursive folding through the following recursive folding inner product proof protocol.

Public Parameters

$\vec{G}, \vec{H} \in \mathbb{G}^n; U, T \in \mathbb{G}$

Public Input

$P = \vec{a}\vec{G} + \vec{b}\vec{H} + cU + \rho T$

Witnesses: $\vec{a}, \vec{b} \in \mathbb{Z}_p^n; c \in \mathbb{Z}_p$

Step 1 (Commitment Step): Prover sends commitments of two mask vectors P_0 and C_1, C_2:

1. $P_0 = \vec{a}_0\vec{G} + \vec{b}_0\vec{H} + \rho_0 T$
2. $C_1 = \vec{a}_0 \cdot \vec{b}_0 \cdot U + \rho_1 T$
3. $C_2 = (\vec{a} \cdot \vec{b}_0 + \vec{a}_0 \cdot \vec{b}) \cdot U + \rho_2 T$

Step 2 (Challenge): Verifier replies with a challenge number z

Step 3: Prover calculates the flattened vectors $\vec{a}'$, $\vec{b}'$, ρ' with z, and sends ρ_c

1. $\vec{a}' = \vec{a} + z\vec{a}_0$
2. $\vec{b}' = \vec{b} + z\vec{b}_0$
3. $\rho' = \rho + z\rho_0$
4. $\rho_c = z^2\rho_1 + z\rho_2$

Verification

Verifier can calculate P'

1. $P' = P + zP_0 + zC_2 + z^2C_1 - \rho_c T$

Thus, Prover and Verifier can continue to run the rIPA protocol to prove $\vec{a}' \cdot \vec{b}' \overset{?}{=} c'$, where the commitments of these three values are merged into $P' = \vec{a}'\vec{G} + \vec{b}'\vec{H} + (\vec{a}' \cdot \vec{b}')U + \rho' T$.

5. Complete Protocol

Below is the complete protocol combining recursive folding and Square-root IPA, which supports the Zero-Knowledge property. If the ZK property is not needed, simply remove the H part related to ρ.

Public Parameters

- $G_0, G_1, G_2, \ldots, G_{2^n-1}, H, U \in \mathbb{G}$.

Calculating Commitment

1. Prover rearranges $\vec{a}$ into a matrix $M_a \in \mathbb{F}_p^{l\times h}$:

$$M_a = \begin{bmatrix} a_0 & a_1 & a_2 & \cdots & a_{l-1} \\ a_l & a_{l+1} & a_{l+2} & \cdots & a_{2l-1} \\ a_{2l} & a_{2l+1} & a_{2l+2} & \cdots & a_{3l-1} \\ a_{(h-1)l} & a_{(h-1)l+1} & a_{(h-1)l+2} & \cdots & a_{hl-1} \end{bmatrix} \tag{43}$$

2. Prover calculates commitments $C_0, C_1, \ldots, C_{h-1}$ by row

$$\begin{aligned} C_0 &= \mathsf{cm}(a_0, a_1, \ldots, a_{l-1}; \rho_0) \\ C_1 &= \mathsf{cm}(a_l, a_{l+1}, \ldots, a_{2l-1}; \rho_1) \\ C_2 &= \mathsf{cm}(a_{2l}, a_{2l+1}, \ldots, a_{3l-1}; \rho_2) \\ \cdots &= \cdots \\ C_{h-1} &= \mathsf{cm}(a_{(h-1)l}, a_{(h-1)l+1}, \ldots, a_{hl-1}; \rho_{h-1}) \end{aligned} \tag{44}$$

Here, $\mathsf{cm}(\vec{a}; \rho)$ is defined as:

$$\mathsf{cm}(\vec{a}; \rho) = \sum_{i=0}^{l-1} a_i G_i + \rho H \tag{45}$$

Evaluation Proof Protocol

Public Input

1. Commitment of $\vec{a}$: $(C_0, C_1, \ldots, C_{h-1})$
2. $\vec{u} = (u_0, u_1, \ldots, u_{n-1}) = \vec{u}_L \parallel \vec{u}_R$, where $|\vec{u}_L| = \log(h)$, $|\vec{u}_R| = \log(l)$

3. $v = \tilde{f}(u_0, u_1, \ldots, u_{n-1})$

Witness

1. $\vec{a}$
2. $(\rho_0, \rho_1, \ldots, \rho_{h-1})$

Proof Protocol

Round 1

1. Prover calculates $\vec{e}$:

$$\begin{aligned} e_0 &= \tilde{eq}(\mathsf{bits}(0), \vec{u}_R) \\ e_1 &= \tilde{eq}(\mathsf{bits}(1), \vec{u}_R) \\ \cdots &= \quad \cdots \\ e_{h-1} &= \tilde{eq}(\mathsf{bits}(h-1), \vec{u}_R) \end{aligned} \tag{46}$$

2. Prover calculates the matrix multiplication of $\vec{e}$ and M_a, obtaining $\vec{b}$, length l

$$\begin{aligned} b_0 &= \langle \vec{e}, (a_0, a_l, \ldots, a_{(h-1)l}) \rangle \\ b_1 &= \langle \vec{e}, (a_1, a_{l+1}, \ldots, a_{(h-1)l+1}) \rangle \\ \cdots &= \quad \cdots \\ b_{l-1} &= \langle \vec{e}, (a_{l-1}, a_{l+l-1}, \ldots, a_{(h-1)l+l-1}) \rangle \end{aligned} \tag{47}$$

3. Prover calculates the commitment C^* of $\vec{b}$

$$C^* = \mathsf{cm}(\vec{b}; \rho^*) \tag{48}$$

Round 2.

Prover and Verifier conduct an IPA protocol to complete the inner product proof of $\vec{b}$ and $\vec{d}$, $\vec{d}$ is calculated as follows:

$$\begin{aligned} d_0 &= \tilde{eq}(\mathsf{bits}(0), \vec{u}_L) \\ d_1 &= \tilde{eq}(\mathsf{bits}(1), \vec{u}_L) \\ \cdots &= \quad \cdots \\ d_{h-1} &= \tilde{eq}(\mathsf{bits}(h-1), \vec{u}_L) \end{aligned} \tag{49}$$

1. Verifier sends a random number γ
2. Prover and Verifier calculate $U' \in \mathbb{G}$

$$U' = \gamma \cdot U \tag{50}$$

Round 3 (Repeated $i = 0, 1, \ldots, n-1$).

First introduce the following symbols, for example, $\vec{b}_L$ represents the first half of $\vec{b}$, $\vec{b}_R$ represents the second half of $\vec{b}$.

$$\begin{aligned} \vec{b}_L^{(i)} &= (b_0^{(i)}, b_1^{(i)}, \ldots, b_{2^{n-1}-1}^{(i)}) \\ \vec{b}_R^{(i)} &= (b_{2^{n-1}}^{(i)}, b_{2^{n-1}+1}^{(i)}, \ldots, b_{2^n-1}^{(i)}) \\ \vec{d}_L^{(i)} &= (d_0^{(i)}, d_1^{(i)}, \ldots, d_{2^{n-1}-1}^{(i)}) \\ \vec{d}_R^{(i)} &= (d_{2^{n-1}}^{(i)}, d_{2^{n-1}+1}^{(i)}, \ldots, d_{2^n-1}^{(i)}) \\ \vec{G}_L^{(i)} &= (G_0^{(i)}, G_1^{(i)}, \ldots, G_{2^{n-1}-1}^{(i)}) \\ \vec{G}_R^{(i)} &= (G_{2^{n-1}}^{(i)}, G_{2^{n-1}+1}^{(i)}, \ldots, G_{2^n-1}^{(i)}) \end{aligned} \tag{51}$$

Note the initial values here, $\vec{b}^{(0)} = \vec{b}$, $\vec{d}^{(0)} = \vec{d}$, $\vec{G}_L^{(0)} = \vec{G}_L$, $\vec{G}_R^{(0)} = \vec{G}_R$.

1. Prover sends $L^{(i)}$ and $R^{(i)}$:

$$\begin{aligned} L^{(i)} &= \mathsf{cm}_{\vec{G}_L^{(i)}}(\vec{b}^{(i)}; \rho_L^{(i)}) + \langle \vec{b}_R^{(i)}, \vec{d}_L^{(i)} \rangle \cdot U' \\ R^{(i)} &= \mathsf{cm}_{\vec{G}_R^{(i)}}(\vec{b}^{(i)}; \rho_R^{(i)}) + \langle \vec{b}_L^{(i)}, \vec{d}_R^{(i)} \rangle \cdot U' \end{aligned} \tag{52}$$

2. Verifier sends a random number $\mu^{(i)}$,
3. Prover calculates and sends the following values:

$$\begin{aligned} \vec{b}^{(i+1)} &= \vec{b}_L^i + \mu^{(i)} \cdot \vec{b}_R^i \\ \vec{d}^{(i+1)} &= \vec{d}_L^i + {\mu^{(i)}}^{-1} \cdot \vec{d}_R^i \end{aligned} \tag{53}$$

4. Prover and Verifier calculate $\vec{G}^{(i+1)}$

$$\vec{G}^{(i+1)} = \vec{G}_L^{(i)} + {\mu^{(i)}}^{-1} \cdot \vec{G}_R^{(i)} \tag{54}$$

5. Prover and Verifier recursively perform Round 3 until $i = n - 1$
6. Prover calculates

$$\hat{\rho} = \rho^* + \sum_{i=0}^{n-1} \mu^{(i)} \cdot \rho_L^{(i)} + {\mu^{(i)}}^{-1} \cdot \rho_R^{(i)} \tag{55}$$

Round 4.

1. Prover calculates and sends R, where $r, \rho_r \in \mathbb{F}_p$ are random numbers sampled by Prover

$$R = r \cdot (G^{(n-1)} + b^{(n-1)} \cdot U') + \rho_r \cdot H \tag{56}$$

Round 5.

1. Verifier sends a random number $\zeta \in \mathbb{F}_p$
2. Prover calculates z and z_r

$$z = r + \zeta \cdot b^{(n-1)} \tag{57}$$

$$z_r = \rho_r + \zeta \cdot \hat{\rho} \tag{58}$$

Verification

1. Verifier calculates C^* and P

$$C^* = d_0 C_0 + d_1 C_1 + \ldots + d_{h-1} C_{h-1} \tag{59}$$

$$P = C^* + \sum_{i=0}^{n-1} \mu^{(i)} L^{(i)} + {\mu^{(i)}}^{-1} R^{(i)} \tag{60}$$

2. Verifier verifies if the following equation holds

$$R + \zeta \cdot P \overset{?}{=} z \cdot (G^{(n-1)} + b^{(n-1)} \cdot U') + z_r \cdot H \tag{61}$$

References

1. [WTSTW16] Riad S. Wahby, Ioanna Tzialla, abhi shelat, Justin Thaler, and Michael Walfish. "Doubly-efficient zkSNARKs without trusted setup." In 2018 IEEE Symposium on Security and Privacy (SP), pp. 926-943. IEEE, 2018. https://eprint.iacr.org/2016/263
2. [BBB+18] Bünz, Benedikt, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. "Bulletproofs: Short proofs for confidential transactions and more." In 2018 IEEE symposium on security and privacy (SP), pp. 315-334. IEEE, 2018. https://eprint.iacr.org/2017/1066
3. [BCC+16] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Jens Groth, and Christophe Petit. "Efficient Zero-Knowledge Arguments for Arithmetic Circuits in the Discrete Log Setting." In Advances in Cryptology–EUROCRYPT 2016: 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II 35, pp. 327-357. Springer Berlin Heidelberg, 2016. https://eprint.iacr.org/2016/263