KZG Extractability based on ROM

KZG10 and other polynomial commitment proofs are often used to construct SNARKs, typically by compiling polynomial oracles in an Interactive Oracle Proof using PCS.

Considering security, the Knowledge Soundness of the IOP itself is easy to guarantee. However, for the SNARK obtained after compiling the IOP with PCS, proving its Knowledge Soundness property is not as straightforward.

Compared to the IOP model where the prover sends an oracle containing an entire polynomial (in the IOP model, the length of the oracle is the same as the polynomial, but the verifier doesn't read it entirely), in SNARK, the prover only sends commitments to the polynomials, which contain very little information: just the values of the polynomial at a few points.

Therefore, we can only guarantee the Knowledge Soundness of SNARK if we ensure that the polynomial commitment itself is "extractable". (For a detailed argument, refer to Interactive Oracle Proofs by Eli Ben-Sasson et al.)

Unfortunately, Kate et al. did not prove that the KZG10 protocol has extractability. Therefore, to use KZG10 in constructing SNARKs, we must reprove its security.

A series of previous works, including Sonic [MBK+19], Plonk [GWC19], Marlin [CHM+19], proposed schemes to prove that the KZG10 scheme satisfies extractability based on non-falsifiable assumptions (Knowledge Assumptions) or based on idealized group models (Idealized Group Model) such as GGM, AGM. It can be said that most SNARK systems constructed based on the KZG scheme currently indirectly rely on the idealized group model.

At the same time, SNARK systems use the Fiat-Shamir transform to achieve non-interactive proofs, which means they also rely on another strong idealized model, namely the Random Oracle Model (ROM). This situation puts us in a rather bad position: our SNARK systems would have the flaws of both models! In recent years, some papers have attacked them separately.

Compared to the idealized group model, the ROM model assumption is weaker (which means stronger security). If we can prove the security of the KZG scheme under the ROM model, we can remove the SNARK system's dependence on the idealized group model, thus increasing our confidence in its security.

In this context, Lipmaa, Parisella, and Siim published their work "Constant-Size zk-SNARKs in ROM from Falsifiable Assumptions" (hereinafter referred to as [LPS24]) this year, making a significant step towards our goal. Their contributions include:

1. Proving the special soundness property of the KZG scheme under the ROM model based on a newly proposed falsifiable assumption
2. Further proving that the KZG scheme satisfies black-box extractability, for use in compiling IOPs
3. Making partial progress in proving the knowledge soundness property of Plonk under the ROM model

In this article, we will focus on introducing the work of the first point.

Special Soundness

To introduce special soundness, we first need to understand interactive proofs and their security definitions.

Interactive Proofs and Knowledge Soundness

【Definition 1: Public-coin Interactive Proofs】

An interactive protocol between two parties (prover and verifier) for proving a target relation R is called an Interactive Proof, denoted as $\Pi = (P, V)$, where P, V are the prover and verifier algorithms respectively. Specifically,

- Prover input: public statement (denoted as x), secret witness (denoted as w)
- Verifier input: public statement (denoted as x)
- The prover and verifier interact through a series of exchanges, and the collection of all interaction messages is called a transcript
- The verifier outputs 1 for acceptance, 0 for rejection.

If all random numbers used by the verifier during the interaction are public, we call such an interactive protocol a Public-coin Interactive Proof. Additionally, if the prover sends k messages and the verifier sends $k-1$ messages during the entire interaction, we call it a $(2k-1)$-step protocol.

As is well known, to ensure that an interactive proof is secure, it needs to satisfy two security properties:

- **Completeness:** For any honest prover P executing the protocol, and if there **exists** w such that (x, w) satisfies relation R, then P can cause the verifier to output acceptance by executing the protocol.
- **Soundness:** For any potentially malicious prover, and if there **does not exist** w such that (x, w) satisfies relation R, then P cannot cause the verifier to output acceptance by executing the protocol.

The above two security properties ensure the basic security of interactive proofs, but the definition of Soundness can only ensure that a certain statement x indeed belongs to relation R, and cannot meet the security requirements of some application scenarios. For example, in an identity authentication system, we require the prover to prove their identity: that is, they "possess" a private key sk corresponding to the public key pk satisfying $pk = sk \cdot G$. If the proof only ensures the Soundness property, then the verifier only knows the conclusion that "pk belongs to the cyclic group $\mathbb{G}$ constituted by the generator G". But this conclusion cannot guarantee that the prover actually possesses the private key sk. In fact, we can prove $pk \in \mathbb{G}$ without knowing sk, for example, using Fermat's Little Theorem.

Therefore, we need a stronger security definition, namely **"**Knowledge Soundness"

【Definition 2: Knowledge Soundness】

For an interactive proof $\Pi = (P, V)$, if there exists a polynomial-time algorithm P^* that can forge a proof to make the verifier accept with a non-negligible probability ϵ without knowing the w corresponding to x, then there must exist a polynomial-time extractor algorithm E, which uses P^* as a rewindable Oracle, can extract a w satisfying x with a non-negligible probability ϵ'. We call $|\epsilon' - \epsilon|$ the soundness error, and if the size of this error is negligible, then Π satisfies the Knowledge Soundness property.

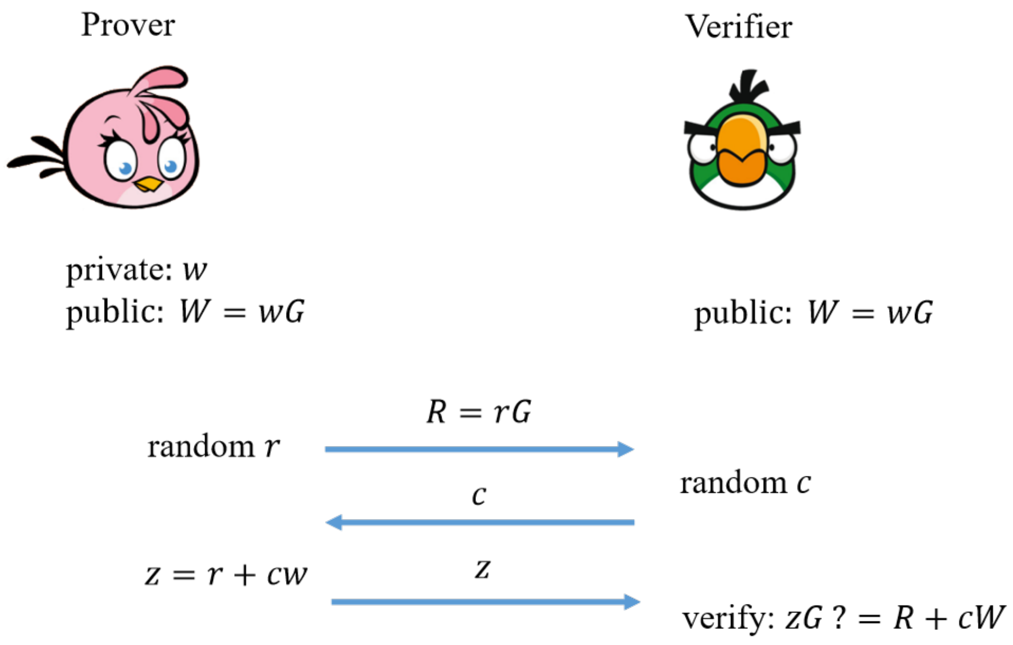
【Note】If a Public-coin Interactive Proof satisfies Completeness and Soundness for any adversary, we call it a Proof of Knowledge. If Soundness is only satisfied for polynomial-time adversaries, we call it an Argument of Knowledge.

It can be seen that the key to the definition of Knowledge Soundness lies in emphasizing the feasibility of constructing an extractor algorithm. That is to say, if a malicious prover claims that it is feasible to forge a legal proof without knowing w, then constructing an extractor for w based on this malicious prover is equally feasible, which contradicts the claim of the malicious prover. This ensures that any prover who can output a legal proof must "possess" the secret value w.

Proof of Knowledge Soundness (Taking Schnorr Protocol as an Example)

We have already given a relatively specific definition of Knowledge Soundness. So how do we prove that an interactive proof protocol satisfies this property? Obviously, the most direct answer is to construct an extractor, but how to construct an extractor is a deep subject (plainly speaking, LPS24 is doing this). To facilitate explaining the work of LPS24, let's start with a relatively simple example to explain the proof idea of knowledge soundness.

As shown in the figure below, the Schnorr protocol [Sch90] is a 3-step interactive proof conducted between the prover and the verifier. By executing this protocol, the prover can prove to the verifier that she possesses a secret value w satisfying the discrete logarithm relation $W = wG$



Their interaction process is as follows

- The prover generates a random value $r \leftarrow \mathbb{F}$, calculates $R = rG$ and sends it to the verifier
- The verifier generates a random value $c \leftarrow \mathbb{F}$ as a challenge and sends it to the prover
- The prover calculates the public value $z = r + cw$ and sends it to the verifier

Finally, the verifier checks $zG \overset{?}{=} R + cW$ based on the messages received in the protocol. For convenience, we denote the transcript of the Schnorr protocol as (R, c, z).

It's easy to prove that the Schnorr protocol satisfies the Completeness property, which we won't elaborate on here.

Next, we focus on the Knowledge Soundness property:

According to Definition 2, we first give the conclusion: If there exists a polynomial-time algorithm P^* that can forge a legal Schnorr proof, then there must exist a polynomial-time extractor algorithm E that can extract the satisfying secret value w by rewinding P^*.

So how to construct an E to complete the proof? It might be difficult to write out the algorithm directly, so let's break down this work into the following steps:

- First, we construct a sub-algorithm E_{ss}, given two transcripts about W as input, denoted as $(R, c_1, z_1), (R, c_2, z_2)$, requiring R to be the same and c_1, c_2 to be different. This sub-algorithm should be able to output w satisfying $W = wG$
- Then, we construct another sub-algorithm E_{rw}, E_{rw} calls P^* as an Oracle, first obtains a legal transcript (R, c_1, z_1), then E_{rw} rewinds P^* to the second step of the Schnorr protocol, tries to send a challenge value different from c_1 to P^* with the same R, until P^* outputs another legal transcript (R, c_2, z_2)
- Finally, the E algorithm first runs E_{rw} to obtain two transcripts that meet the conditions, and then runs E_{ss} to obtain w

【Implementing sub-algorithm E_{ss}】

The implementation of sub-algorithm E_{ss} often appears in the security proofs of various papers. Simply put, E_{ss} can obtain the public values z_1, z_2 from the two input transcripts. Assuming P^* honestly calculated these two values, they should satisfy the following form:

$$\begin{aligned} z_1 &= r + c_1 w' \\ z_2 &= r + c_2 w' \end{aligned} \tag{1}$$

By solving the equation, we can calculate $w' = (z_1 - z_2)/(c_1 - c_2)$ as a possible secret value, and we only need to check $w'G \overset{?}{=} W$ to know if it's legal. If they are equal, E' directly outputs the legal secret value $w = w'$, and the algorithm completes. If they are not equal, E' can use the obtained result to construct a reduction to break the discrete logarithm assumption:

$$\frac{(z_1 - z_2)}{c_1 - c_2} G = W \tag{2}$$

Since the probability of breaking the discrete logarithm assumption is negligible, we can conclude that the difference between the success probability of E_{ss} and the success probability of P^* is also negligible.

【Obtaining transcripts】

We have implemented the first step, now let's look at the second step, which requires algorithm E_{rw} to call P^* to obtain two legal transcripts. It should be noted that Definition 2 assumes that P^* can only successfully output legal proofs with probability ϵ each time, which means that P^* cannot always succeed. Moreover, the running time of P^* is assumed to be within polynomial time, which also limits that E_{rw} cannot call P^* indefinitely, because considering the feasibility of the algorithm, the total running time of E_{rw} also needs to be within polynomial time.

Therefore, to successfully complete the second step, we must prove the following two points

1. E_{rw} is a polynomial-time algorithm
2. E_{rw} also successfully outputs w with a non-negligible probability

From Knowledge Soundness to Special Soundness

The paper [Cra96] gives a quite elegant proof of the properties of the E_{rw} algorithm. Since the process is quite long and similar to the content of [LPS24] that will be introduced later, we won't describe it here. In any case, the above process is summarized as a lemma:

【Rewinding Lemma】

For a 3-step interactive proof $\Pi = (P, V)$, if there exists a polynomial-time algorithm P^* that can forge a legal transcript with a non-negligible probability, then there must exist a polynomial-time extractor algorithm E_{rw} that can obtain another legal transcript (satisfying the same R and different c) by rewinding P^*, and the success probability of E_{rw} is also non-negligible.

The Rewinding Lemma is not limited to the Schnorr protocol. In fact, for any 3-step Sigma protocol, the extractor algorithm E_{rw} can rewind to obtain additional $k-1$ (polynomial number) legal transcripts.

Therefore, the Rewinding Lemma actually simplifies the process of proving Knowledge Soundness for researchers designing specific protocols. For protocols designed based on the Sigma model, we usually only need to give the construction of sub-algorithm E_{ss} in the security proof. To formally describe this process, cryptographers proposed a new definition, namely Special Soundness

Definition 3: Special Soundness

For a 3-step interactive proof $\Pi = (P, V)$, if there exists a polynomial-time extraction algorithm E_{ss}, given its input as x and two legal transcripts, denoted as $(R, c_1, z_1), (R, c_2, z_2)$, can output the secret value w, then we say Π satisfies Special Soundness.

【Note】The above definition is also called 2-special soundness. If the input of extraction algorithm E includes k transcripts, it is called k-special soundness

With the development of interactive proofs, researchers are not limited to constructing only 3-step protocols. To meet this demand, Special Soundness has been further extended to $(2n-1)$-step interactive proofs. That is, for the $j \in [1, n]$ round, the extractor algorithm E_{rw} needs to obtain additional $k_j - 1$ transcripts by rewinding P^*. Finally, the input of sub-algorithm E_{ss} is no longer a simple transcript vector, but a transcript tree with height n, denoted as $(k_1, \dots, k_n)-$transcript tree. Correspondingly, the property satisfied by this protocol is called $(k_1, \dots, k_n)-$special soundness. For the specific definition of this part, interested readers can read [BCC+16] and [ACK21]. The [LPS24] introduced in this article only uses k-special soundness.

LPS24: KZG10 with Special Soundness

We have already introduced the basic process of the KZG10 polynomial commitment scheme in our previous articles. In [LPS24], the authors first write the KZG10 scheme in a form that conforms to interactive proofs, where the prover has public input $ck = (p, [(\sigma^i)_{i=0}^n]_1, [1, \sigma]_2)$ (i.e., parameters $p \leftarrow Pgen(1^\lambda)$ and SRS), secret input $f(X)$ polynomial, and the verifier only has public input ck. The two parties conduct the following interactive protocol:

- The prover calculates the polynomial commitment $C = [f(\sigma)]_1$ and sends it to the verifier
- The verifier chooses a random r as the evaluation point and sends it to the prover

- The prover calculates and sends the value $v = f(r)$, proof $\pi = [q(\sigma)]_1$, where $q(X) = (f(X) - v)/(X - r)$

The verifier checks $e(C - [v]_1, [1]_2) \overset{?}{=} e(\pi, [\sigma - r]_2)$ based on the interaction data

Similarly, we call the collection of interaction messages between the two parties a transcript. If a transcript tr can pass verification, it is called accepting. Furthermore, if a vector $\vec{tr}$ containing $n+1$ transcripts satisfies the following two requirements, it is admissible:

1. All transcripts in $\vec{tr}$ contain the same polynomial commitment C
2. For any two transcripts $tr_i, tr_j, i, j \in [0, n]$, their evaluation points are different, i.e., $r_i \neq r_j$

In addition to defining the interactive form of the KZG10 scheme, the authors of [LPS24] also proposed a new difficult problem assumption, named Adaptive Rational Strong Diffie-Hellman assumption, abbreviated as ARSDH assumption, defined as follows

【$(n+1)$-ARSDH assumption】

If for any polynomial-time adversary algorithm A, given parameters $p \leftarrow Pgen(1^\lambda)$, SRS generated by random value σ, $([(\sigma^i)_{i=0}^n]_1, [1, \sigma]_2)$, A is required to output a pair $[g]_1, [\varphi]_1$, and a set S of size $n+1$, satisfying the following relation:

$$[g]_1 \neq [0]_1 \wedge e([g]_1, [1]_2) = e([\varphi]_1, [Z_S(\sigma)]_2) \tag{3}$$

If the probability of A succeeding is negligible, then the $(n+1)$-ARSDH assumption is said to hold for the bilinear group parameter generation algorithm $p \leftarrow Pgen(1^\lambda)$.

ARSDH is a relaxation of a known assumption RSDH, where RSDH requires that A cannot choose the set S by itself. In addition, [LPS24] also proves that $(n+1)$-ARSDH can imply the $(n+1)$-SDH assumption (ARSDH implies SDH), that is, if SDH can be broken, then ARSDH can also be broken. Because SDH can imply the evaluation binding property of KZG10, we get the following conclusion

$$(n+1)\text{-ARSDH} \rightarrow (n+1)\text{-SDH} \rightarrow \text{KZG10's binding} \tag{4}$$

The preparatory knowledge has been introduced. Next, following the proof idea of the Schnorr protocol introduced earlier, we first give the construction of the extractor algorithm E_{ss} based on transcripts, that is, proving that KZG satisfies special soundness, and then prove the rewinding lemma.

Special Soundness of KZG

First, let's give the definition:

For a polynomial commitment scheme PC, if there exists a polynomial-time extraction algorithm E_{ss}, given its input as ck and a vector $\vec{tr}$ of length $n+1$ of transcripts, satisfying

1. Any $tr_j \in \vec{tr}$ is accepting (passes verification)
2. $\vec{tr}$ is admissible (C is the same, r is different)

E can output the secret value $f(X)$, satisfying $C = \text{Com}(ck; f) \wedge f(r_j) = v_j, \forall j \in [0, n]$, then we say PC satisfies $(n+1)$-Special Soundness

Obviously, the idea of designing the E_{ss} algorithm is to try to extract a polynomial $f'(X)$ from $\vec{tr}$, and $f'(X)$ is either a legal secret value or an instance that breaks the $(n+1)\text{-ARSDH}$ assumption.

Let's write out the verification relation corresponding to each tr_j:

$$\begin{gathered} e(C-[v_0]_1,[1]_2) = e(\pi_0,[\sigma - r_0]_2) \\ \vdots \\ e(C-[v_n]_1,[1]_2) = e(\pi_n,[\sigma - r_n]_2) \end{gathered} \tag{5}$$

Let $I=[0,n]$, $L_0(X),\ldots,L_n(X)$ be the Lagrange polynomials interpolating the values $S=\{v_i\}_{i\in I}$ on the set I, the expression of $L_j(X)$ is

$$L_j(X) = \frac{\prod_{i\in I/\{j\}}(X-r_i)}{\prod_{i\in I/\{j\}}(r_j-r_i)} \tag{6}$$

Now, multiply both sides of each verification equation by the value of the Lagrange polynomial at σ, for example, the $j\in[0,n]$-th equation is

$$e(C-[v_j]_1,[1]_2)\cdot L_j(\sigma) = e(\pi_j,[\sigma-r_j]_2)\cdot L_j(\sigma) \tag{7}$$

And add all $n+1$ equations, we can get

$$e(\sum_{j\in I}(C-[v_j]_1)\cdot L_j(\sigma),[1]_2) = e(\sum_{j\in I}\pi_j\cdot L_j(\sigma),[\sigma-r_j]_2) \tag{8}$$

Let $\sum_{j\in I}[v_j]_1\cdot L_j(\sigma) = [L(\sigma)]_1$, the left side is

$$\begin{aligned} LHS &= e(C-\sum_{j\in I}[v_j]_1\cdot L_j(\sigma),[1]_2) \\ &= e(C-[L(\sigma)]_1,[1]_2) \end{aligned} \tag{9}$$

Let $\sum_{j\in I}\left(\pi_j/\prod_{i\in I/\{j\}}(r_j-r_i)\right) = \varphi$, the right side is

$$\begin{aligned} RHS &= e(\sum_{j\in I}\pi_j\cdot L_j(\sigma),[\sigma-r_j]_2) \\ &= e([\sum_{j\in I}q_j(\sigma)\cdot\frac{\prod_{i\in I/\{j\}}(\sigma-r_i)}{\prod_{i\in I/\{j\}}(r_j-r_i)}]_1,[\sigma-r_j]_2) \\ &= e([\sum_{j\in I}\frac{q_j(\sigma)}{\prod_{i\in I/\{j\}}(r_j-r_i)}]_1,[Z_S(\sigma)]_2) \\ &= e([\varphi]_1,[Z_S(\sigma)]_2) \end{aligned} \tag{10}$$

Finally, we get the equation

$$LHS = e(C-[L(\sigma)]_1,[1]_2) = e([\varphi]_1,[Z_S(\sigma)]_2) = RHS \tag{11}$$

Based on this equation, the extraction algorithm E_{ss} first obtains $v_0,\ldots,v_n$ from $n+1$ transcripts, and calculates $L(X)$ and $[L(\sigma)]_1 = [\sum_{j\in I}v_j\cdot L_j(\sigma)]_1$. Compare $[L(\sigma)]_1 \overset{?}{=} C$, and perform the following operations based on the result

- If $[L(\sigma)]_1 = C$, E_{ss} directly outputs $L(X)$ as the secret polynomial, and the algorithm completes.

- If $[L(\sigma)]_1 \neq C$, E_{ss} uses $L(X)$ to construct a reduction to break the $(n+1)\text{-ARSDH}$ assumption:
    - E_{ss} calculates $[g]_1 = C - [L(\sigma)]_1$, and outputs $[g]_1$, $[\varphi]_1$ as an instance to break $(n+1)\text{-ARSDH}$
    - Obviously, the above instance satisfies

$$e([g]_1, [1]_2) = e([\varphi]_1, [Z_S(\sigma)]_2) \tag{12}$$

Proof completed.

Rewinding Lemma

The above proof ensures that KZG10 satisfies $(n+1)$-special soundness, but to further ensure knowledge soundness, we also need to prove that it is feasible for E_{rw} to obtain $n+1$ satisfying transcripts by rewinding, that is, the rewinding lemma.

Specifically, for the following E_{rw} algorithm, we need to prove that it can succeed with a non-negligible probability in polynomial time

1. E_{rw} randomly selects r and calls P^* to obtain tr_0
2. Check the validity of tr_0, if valid, continue; if not valid, return to step 1 and select another r'
3. E_{rw} runs a loop algorithm, selecting a new r in each round, and rewinds P^* to obtain a new transcript, with the termination condition being
    1. E_{rw} obtains $(n+1)$ transcripts that meet the requirements (i.e., satisfying accepting and admissible) → Algorithm succeeds
    2. E_{rw} has traversed all possible r, but still hasn't obtained $(n+1)$ transcripts that meet the requirements → Algorithm fails

The [LPS24] paper adopts the same proof idea as [ACK21], letting H be a Boolean matrix with row indices as the set $\{\vec{r} = (r_p, r_{ck}, r_A) \in \{0,1\}^{poly(\lambda)}\}$, where r_p, r_{ck}, r_A are the random numbers used by the $Pgen$ algorithm, SRS, and the adversary respectively. The column indices of H are the challenge value space $\mathbb{F}$. When P^* generates a legal transcript for challenge value r under a certain random number setting $\vec{r}$, we set the corresponding element in H to 1, i.e., $H[\vec{r}][r] = 1$.

Next, we analyze the success probability and running time of the E_{rw} algorithm respectively

【Probability Analysis】

Define events as follows:

- Event A: tr_0 passes verification
- Event B: $\forall j \in [1, n]$, tr_j passes verification

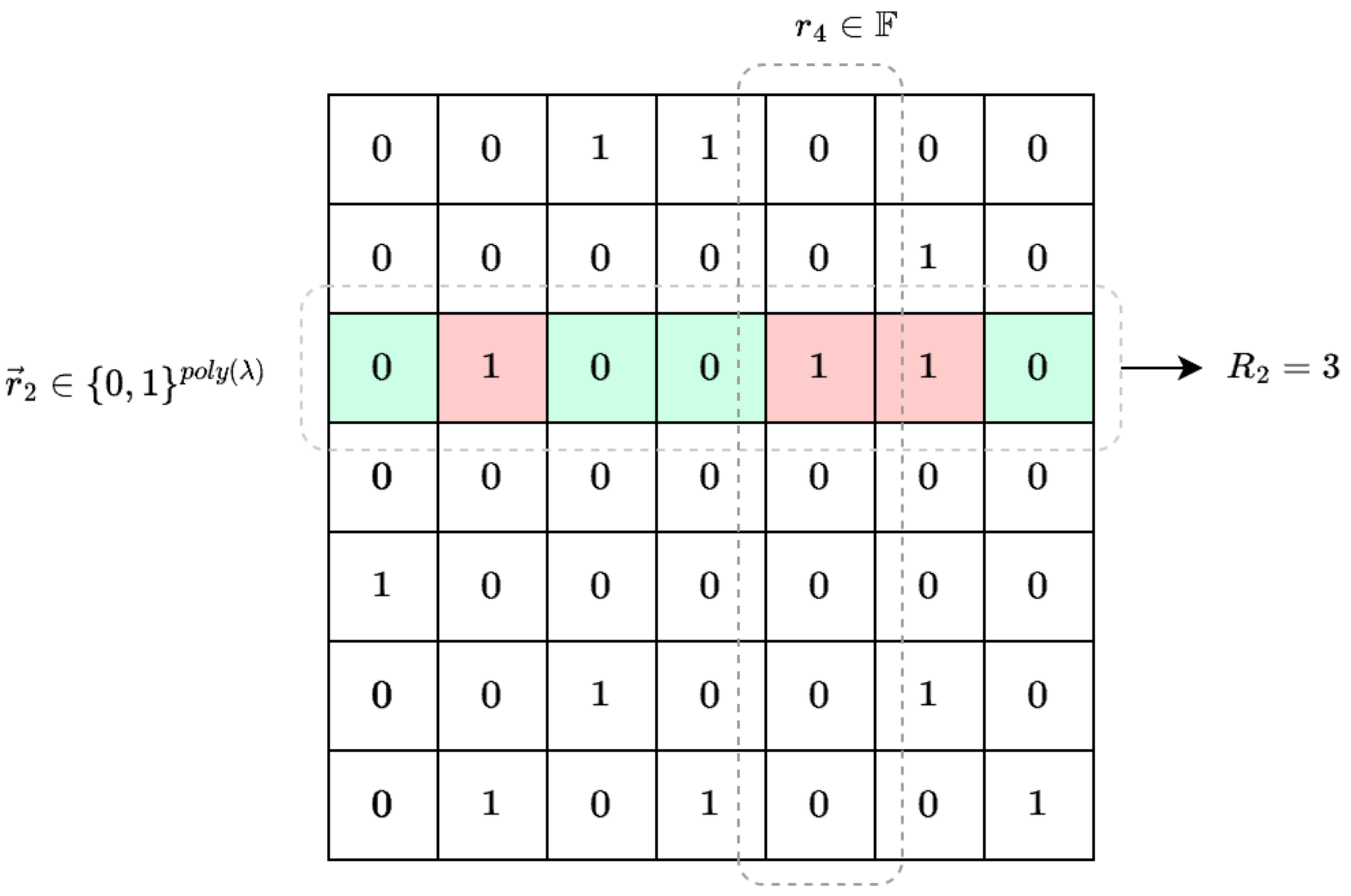
Then the probability of E_{rw} succeeding is calculated as the probability of $A \to B$, that is

$$\begin{aligned} Pr[A \to B] &= Pr[A \wedge (A \to B)] + Pr[\neg A \wedge (A \to B)] \\ &= Pr[A \wedge B] + Pr(\neg A) \end{aligned} \tag{13}$$

【Note】: The truth table of $A \to B$ is

A	B	$A \to B$
T	T	T
T	F	F
F	T	T
F	F	T

Consider the probability $Pr[A \wedge B]$, the event $A \wedge B$ occurs if and only if P^* outputs a legal tr_0 under the random parameter setting $\vec{r}$, and the row $H[\vec{r}]$ where $\vec{r}$ is located has at least $n+1$ "1" elements.



Let R_j be the number of rows in H that contain j "1" elements, for example, $R_2 = 3$ in the above figure. The number of rows containing $\geq n+1$ "1" elements can be calculated as $\sum_{j=n+1}^{|\mathbb{F}|} j \cdot R_j$, the probability $Pr[A \wedge B]$ is calculated as follows

$$\begin{aligned} Pr[A \wedge B] &= \frac{\sum_{j=n+1}^{|\mathbb{F}|} j \cdot R_j}{|\vec{r}| \cdot |\mathbb{F}|} \\ &= \frac{\sum_{j=0}^{|\mathbb{F}|} j \cdot R_j}{|\vec{r}| \cdot |\mathbb{F}|} - \frac{\sum_{j=0}^{n} j \cdot R_j}{|\vec{r}| \cdot |\mathbb{F}|} \\ &= Pr[A] - \frac{\sum_{j=0}^{n} j \cdot R_j}{|\vec{r}| \cdot |\mathbb{F}|} \end{aligned} \tag{14}$$

And because $\sum_{j=0}^{n} j \cdot R_j = \sum_{j=1}^{n} j \cdot R_j \le \sum_{j=1}^{n} n \cdot R_j \le n|\vec{r}|$

We can get the lower bound of $Pr[A \wedge B]$

$$Pr[A \wedge B] \ge Pr[A] - \frac{n|\vec{r}|}{|\vec{r}||\mathbb{F}|} = Pr[A] - \frac{n}{|\mathbb{F}|} \tag{15}$$

Finally, we get the lower bound of the success probability of E_{rw}

$$\begin{aligned} Pr[A \to B] &= Pr[A \wedge B] + Pr(\neg A) \\ &\ge Pr[A] - \frac{n}{|\mathbb{F}|} + Pr[\neg A] \\ &= 1 - \frac{n}{|\mathbb{F}|} \end{aligned} \tag{16}$$

【Running Time Analysis】

For the E_{rw} algorithm, it can be considered that its running time is mainly related to the time of calling the P^* algorithm. And because P^* is a polynomial-time algorithm, we only need to calculate the number of times E_{rw} calls the P^* algorithm, denoted as Q, to deduce that the time complexity of E_{rw} algorithm is $poly(\lambda) \cdot Q$.

Consider that E_{rw} successfully obtains a legal tr_0 in step 2 (i.e., event A occurs), E_{rw} continues to run the loop in step 3. Since E_{rw} needs to call the P^* algorithm once in each round of the loop, we can obtain Q by calculating the expected number of loop iterations.

Let's first discuss the problem of calculating the number of loop iterations separately: Given a random parameter $\vec{r}$, assuming the corresponding row vector $H[\vec{r}]$ in H contains $\delta_{\vec{r}}|\mathbb{F}|$ "1" elements, $|\mathbb{F}|$ is the length of the vector $H[\vec{r}]$. On the premise of already selecting one "1" element in $H[\vec{r}]$ (i.e., tr_0), solve for the expected number of times E_{rw} selects n "1" elements from the remaining $|\mathbb{F}| - 1$ items.

To calculate the expectation of Q, we need to introduce the concept of Negative HyperGeometric distribution (NHG distribution)

NHG distribution: Given a blind box containing N balls, of which K balls are marked, it is required to take out only one ball at a time, and not put it back, until $k \le K$ marked balls are taken out. Let X be the total number of all balls taken out when the ball-taking ends, the expectation of X is $E[NHG_{N,K,k}] = k(N+1)/(K+1)$.

Correspondingly, when the number of "1" elements contained in $H[\vec{r}]$ is greater than n, Q conforms to the NHG distribution. Assuming that each $H[\vec{r}]$ contains $\delta_{\vec{r}}|\mathbb{F}|$ "1" elements, we can calculate the expectation of Q as follows:

- $H[\vec{r}]$ contains at least $n+1$ "1" elements, $E[Q|A \wedge \vec{r}] = E[NHG_{N,K,k}] + 1 = n/\delta_{\vec{r}} + 1$, where $N = \mathbb{F} - 1, K = \delta_{\vec{r}}|\mathbb{F}| - 1, k = n$
- $H[\vec{r}]$ contains less than $n+1$ "1" elements, i.e., $\delta_{\vec{r}}|\mathbb{F}| \le n$, algorithm E_{rw} will keep executing the loop until traversing all elements in $H[\vec{r}]$, obviously $E[Q|A \wedge \vec{r}] = |\mathbb{F}| \le n/\delta_{\vec{r}}$

The above considers the case when event A occurs. Since for any $\vec{r}$, $H[\vec{r}]$ contains $\delta_{\vec{r}}|\mathbb{F}|$ "1" elements, the probability of event A occurring is $Pr[A] = \delta_{\vec{r}}$, calculate

$$\begin{aligned} E[Q|\vec{r}] &= E[Q|A \wedge \vec{r}] \cdot Pr[A] + E[Q|\neg A \wedge \vec{r}] \cdot Pr[\neg A] \\ &\leq \frac{n}{\delta_{\vec{r}}} \cdot \delta_{\vec{r}} + 1 \cdot (1 - \delta_{\vec{r}}) = n + 1 - \delta_{\vec{r}} \leq n + 1 \end{aligned} \tag{17}$$

For all $\vec{r} \in \{0,1\}^{poly(\lambda)}$, calculate the expectation of Q as follows

$$E[Q] = \sum_{\vec{r}} E[Q|\vec{r}] \cdot Pr[\vec{r}] \leq \sum_{1}^{|\vec{r}|} \frac{n+1}{|\vec{r}|} = n + 1 \tag{18}$$

Proof completed.

References

[CHM+19] Chiesa, Alessandro, Yuncong Hu, Mary Maller, et al. "Marlin: Preprocessing zkSNARKs with Universal and Updatable SRS." *Cryptology ePrint Archive* (2019). https://eprint.iacr.org/2019/1047

[MBK+19] Maller Mary, Sean Bowe, Markulf Kohlweiss, et al. "Sonic: Zero-Knowledge SNARKs from Linear-Size Universal and Updatable Structured Reference Strings." *Cryptology ePrint Archive* (2019). https://eprint.iacr.org/2019/099

[GWC19] Ariel Gabizon, Zachary J. Williamson, Oana Ciobotaru. "PLONK: Permutations over Lagrange-bases for Oecumenical Noninteractive arguments of Knowledge." *Cryptology ePrint Archive* (2019). https://eprint.iacr.org/2019/953

[LPS24] Helger Lipmaa, Roberto Parisella, Janno Siim. "Constant-Size zk-SNARKs in ROM from Falsifiable Assumptions." Cryptology ePrint Archive (2024). https://eprint.iacr.org/2024/173

[ACK21] *Thomas Attema, Ronald Cramer, and Lisa Kohl* "A Compressed Sigma-Protocol Theory for Lattices" Cryptology ePrint Archive (2021). https://eprint.iacr.org/2021/307

[Sch90] Claus-Peter Schnorr. "Efficient identification and signatures for smart cards." In Gilles Brassard, editor, CRYPTO'89, volume 435 of LNCS, pages 239–252. Springer, Heidelberg, August 1990.

[Cra96] Ronald Cramer. "Modular Design of Secure yet Practical Cryptographic Protocols". PhD thesis, CWI and University of Amsterdam, 1996.