Teng Liu Zero Knowledge Proof Graded Exercise 9 November 21, 2023

Exercise 1.a.1. Using $G1, \cdots, G_L$ and $\text{Custom}_1, \cdots, \text{Custom}_L$, write down a summation equation connecting the values of the wire functions at level k and level $k+1$.

Solution: The summation equation should be:

$$w_i(\vec{z}) = \sum_{i}^{L} \sum_{\substack{\vec{b_1} \in \{0,1\}^{l_{k+1}} \\ \cdots \\ \vec{b_m} \in \{0,1\}^{l_{k+1}}}} \text{Custom}_i(\vec{z}, \vec{b_1}, \cdots, \vec{b_m}) \cdot G_i(w_{i+1}(\vec{b_1}), \cdots, w_{i+1}(\vec{b_m}))$$

The first summation iterates among all the custom polynomials, and the second summation iterates all the possible encoding of the gates in layer $i+1$ and if it is a real layout the Custom function would return 1.

Exercise 1.a.2. When $m \geq 2$, state in detail how to modify the sumcheck and 2-to-1 reductions to verify this circuit. State the soundness error and communication cost in terms of m, d, l_k, l_{k+1}.

Solution: The initial claim of this protocol is just the same as GKR protocol. And what we need to care about is just the sumcheck part from layer i to layer $i+1$ and the 2-to-1 reduction part.

1. **For the sumcheck part**:

   In layer i the prover should claim that $\tilde{w_i}(\vec{z_i}) = v_i$, and according to the summation equation we present in the previous problem:

   $$w_i(\vec{z}) = \sum_{i}^{L} \sum_{\substack{\vec{b_1} \in \{0,1\}^{l_{k+1}} \\ \cdots \\ \vec{b_m} \in \{0,1\}^{l_{k+1}}}} \text{Custom}_i(\vec{z}, \vec{b_1}, \cdots, \vec{b_m}) \cdot G_i(w_{i+1}(\vec{b_1}), \cdots, w_{i+1}(\vec{b_m}))$$

   the prover should do the calculation and simplify the RHS of the equation into a polynomial $p(\vec{b_1}, \cdots, \vec{b_m})$ just like in the GKR protocol we simplify the summation into a polynomial $p(\vec{b}, \vec{c})$.

   Then the prover should claim that:

   $$\sum_{\substack{\vec{b_1} \in \{0,1\}^{l_{k+1}} \\ \cdots \\ \vec{b_m} \in \{0,1\}^{l_{k+1}}}} p(\vec{b_1}, \cdots, \vec{b_m}) = v_i$$

And this claim could be verified using normal sumcheck protocol with $m * l_{i+1}$ rounds.

Then the verifier could generate random vectors $\vec{\beta_1}, \cdots, \vec{\beta_m}$ from $\mathbb{F}^{l_{i+1}}$. Verifier now should be convinced that the summation of polynomial p is exactly the correct result but we still need to verify that this polynomial p is equal to the wire function w_i so these random vectors are used to prove that prover didn't send a malicious polynomial p with the correct summation.

Once the prover received the random vector, it should claim that:

$$p(\vec{\beta_1}, \cdots, \vec{\beta_m}) = u_{i+1}$$

To verify the above claim the prover should calculate the wire functions for verifier because it would be costy to let verifier to calculate the wire function itself, i.e.:

$$\begin{aligned} B_1 &= \tilde{w_{i+1}}(\vec{\beta_1}) \\ &\cdots \\ B_m &= \tilde{w_{i+1}}(\vec{\beta_m}) \end{aligned}$$

Then the verifier should check that:

$$\sum_i^L \text{Custom}_i(\vec{z}, \vec{\beta_1}, \cdots, \vec{\beta_m}) \cdot G_i(B_1, \cdots, B_m) = u_{i+1}$$

Then the verifier need to verify that the value $B_1, \cdots, B_m$ are not maliciously calculated, and it goes to the next part.

2. **For the 2-to-1 reduction part**:

   In GKR protocol, we have $L_{i+1} = (1 - X) \cdot \vec{\beta_{i+1}} + X \cdot \vec{\gamma_{i+1}}$, it's actually a convex combination of the two vector $\vec{\beta_{i+1}}$ and $\vec{\gamma_{i+1}}$, so here we can generalise this to m vectors:

$$\begin{aligned} L_{i+1}(X_1, \cdots, X_{m-1}) &= X_1 \cdot \vec{\beta_1} + \cdots X_{m-1} \cdot \vec{\beta_{m-1}} + (1 - \sum_i X_i) \cdot \vec{\beta_m} \\ Q_{i+1}(X_1, \cdots, X_{m-1}) &= \tilde{w_{i+1}}(L_{i+1}(X_1, \cdots, X_{m-1})) \end{aligned}$$

The prover need to send Q to the verifier and verifier need to check the following:

$$\begin{aligned} Q_{i+1}(1, 0, \cdots, 0) &= B_1 \\ Q_{i+1}(0, 1, \cdots, 0) &= B_2 \\ &\cdots \\ Q_{i+1}(0, 0, \cdots, 0) &= B_m \end{aligned}$$

Finally the verifier should generate random $X_1, \cdots, X_{m-1}$ from $\mathbb{F}$ and send back the prover.

Since verifier already have the coefficients of polynomial Q so it could calculate the value $Q_{i+1}(X_1, \cdots, X_{m-1})$ by itself, say it's v_{i+1} and the prover need to prove that $w_{i+1}(L_{i+1}(X_1, \cdots, X_{m-1})) = v_{i+1}$ to show that prover didn't send a malicious Q.

So this whole process reduce m claims into 1 claim about the wire function value. So it could be recursively verified by next layer.

3. **For the communication cost:**

   In sumcheck part, the communication contains a sumcheck protocol of a polynomial with $m * l_{i+1}$ variables so the cost of sumcheck protocol is $O(m * l_{i+1})$. Then verifier need to send back random vector, which also cost $O(m * l_{i+1})$. The other cost is just constant so the communication cost of sumcheck part is $O(m * l_{i+1})$.

   In m-to-1 part, it's a bit different. Since the prover need to send the coefficients of Q to verifier, so the communication cost is dominated by the number of monomials that polynomial Q has. In this case there are at most $O(\sum_i^m \binom{l_{i+1}}{i})$ monomials where $\binom{\cdot}{\cdot}$ means binomial coefficient. So the communication cost should be $O(\sum_i^m \binom{l_{i+1}}{i})$.

4. **For the soundness error**:

   In sumcheck part, the soundness error only comes from prover sending malicious p with accidentally correct value at that random point (in the other cases it will always reduce a false statement to a false statement or reject). According to Schwarz-Zippel Lemma, the soundness error is bounded by the degree of the polynomial p and the size of field $\mathbb{F}$, which is $(m + d) \cdot l_{i+1}/|\mathbb{F}|$. That's because the degree of p is at most $(m+d) \cdot l_{i+1}$ (because in the worst case, Custom function would provide degree $m \cdot l_{i+1}$ and the custom gate G would provide degree $d \cdot l_{i+1}$)

   In m-to-1 part, the soundness error is the same: $l_{i+1}/|\mathbb{F}|$ because the degree of polynomial Q is still l_{i+1}, the only difference is that Q now has multiple variables.

Exercise 1.b. Write down a single custom gate $G : \mathbb{F} \to \mathbb{F}$ for layer 2 of C, and a function Custom describing the locations of G.

Solution: Here the custom gate is $G(X) = (X + K)^3$ and the the function:

$$\text{Custom}(I, J) = \begin{cases} 1, & I == J \\ 0, & otherwise \end{cases}$$

Here I and J stands for the binary representation of the position of the input and output. The first n bits of I and J represents the row index and the last n bits is for the column index.

Exercise 1.c.

Solution: This is pretty much similar to the layer sum equation in GKR protocol that we use to represent wire value function as a sum of previous layer wire value times some position function describing the gate position like add or mul.

Here the first statement means that the value of position i in layer i comes from the value of postion $\text{ShiftRows}^{-1}(\mathbf{i})$ in layer $i + 1$. It means that the value of postion $\mathbf{i}$ in layer $i + 1$ will **contribute** to the value of postion $\text{ShiftRows}(\mathbf{i})$ in layer i.

And the second statements states that it will iterate all the postions of layer $i+1$ to find the one that will contribute to position $\mathbf{i}$. So among every item in the summation, only one of the $\text{Eq}(\mathbf{i}; \text{ShiftRows}(\mathbf{j}))$ will be true or will be value 1. So the other items would be 0 times some value, and only the targeted item would be 1 times $W_{k+1}(\mathbf{j})$. So the two statemensts are identical.

Exercise 1.d.

Solution: Since we already transfered the wire function W_k into a summation form, we can use sumcheck protocol to help us.

First the prover claims $\tilde{W_k}(\vec{r}) = v_k$.

Then just like the GKR protocol, the prover calculate a polynomial p_k such that:

$$p_k(\vec{j}) = \tilde{\text{Eq}}(\vec{r}; \tilde{\text{ShiftRows}}(\vec{j})) \cdot \tilde{W_{k+1}}(\vec{j})$$

Then the prover could claim that: $\sum_{\vec{j} \in \{0,1\}^{2n}} p_k(\vec{j}) = v_k$. This claim could be verified by sumcheck protocol.

Then the verifier need to verify that prover didn't send a malicious polynomial p with correct summation. The verifier should use the random vector $\vec{b}$ and the value u_k generated in the process of sumcheck protocol for p. The check works as follow:

For the check, verifier could calculate the first half of the equation: $\tilde{\text{Eq}}(\vec{r}; \tilde{\text{ShiftRows}}(\vec{b}))$ and the latter half should be provided by the prover $\tilde{W_{k+1}}(\vec{b})$. Once the prover send the verifier the value of $\tilde{W_{k+1}}(\vec{b}) = v_{k+1}$, the verifier could calculate the value of $p_k(\vec{b})$ by itself and it should check whether it equals to the value u_k.

If all the check passes, then the problem is reduced to this problem: whether the provided value of $\tilde{W_{k+1}}(\vec{b})$ is correct or not. So we get a protocol that could reduce the check of $\tilde{W}_k(\vec{r}) = v_k$ to the check of $\tilde{W_{k+1}}(\vec{b}) = v_{k+1}$.

This protocol is complete because it always reduces a true statemenet to another true statement so the check would always pass. In detail, consider this equation:

$$\tilde{W}_k(\vec{r}) = \sum_{\vec{j}} \tilde{\text{Eq}}(\vec{r}; \text{ShiftRows}(\vec{j})) \cdot \tilde{W_{k+1}}(\vec{j})$$

The LHS and the RHS are both the MLE form of $W_k(\vec{r})$ so they are the same as polynomial of $\vec{r}$ and they are still equal even if at point outside $\{0,1\}$.

So $\sum_{\vec{j}\in\{0,1\}^{2n}} p_k(\vec{j}) = \tilde{W}_k(\vec{r}) = v_k$. By the completeness of sumcheck protocol, this check should pass and we also have $p_k(\vec{b}) = u_{k+1}$. Then by the definition of p and W_{k+1}, the check of $p_k(\vec{b}) = \tilde{\text{Eq}}(\vec{r}; \text{ShiftRows}(\vec{b})) \cdot \tilde{W_{k+1}}(\vec{b}) = u_{k+1}$ passes.

Finally, by the definition of $\tilde{W_{k+1}}(\vec{b}) = v_{k+1}$, we know that the check of $\tilde{W_{k+1}}(\vec{b}) = v_{k+1}$ passess.

So the completeness of the protocol is proved.

For the soudness, we will show that this protocol always reduces false claim to false claim with high probabilty or reject it.

Let $\tilde{W}_k(\vec{r}) \neq v_k$, by the same reasoning of the completeness of this protocol, we have $\sum_{\vec{j}\in\{0,1\}^{2n}} p_k(\vec{j}) = \tilde{W}_k(\vec{r}) \neq v_k$. Then according to the soundness of sumcheck protocol, sumcheck will reject this claim or the final statement is false with high probabilty, which means $p_k(\vec{b}) = \tilde{\text{Eq}}(\vec{r}; \text{Shift}\tilde{\text{Rows}}(\vec{b})) \cdot \tilde{W_{k+1}}(\vec{b}) \neq u_{k+1}$ with soundness error $2 * 2n/|\mathbb{F}| = 4n/|\mathbb{F}|$.

Then if prover send the correct $\tilde{W_{k+1}}(\vec{b}) = v_{k+1}$, then $p_k(\vec{b}) = \tilde{\text{Eq}}(\vec{r}; \text{Shift}\tilde{\text{Rows}}(\vec{b})) \cdot \tilde{W_{k+1}}(\vec{b}) \neq u_{k+1}$ so the verifier will reject.

If prover send the malicious $\tilde{W_{k+1}}(\vec{b}) = v'_{k+1} \neq v_{k+1}$, then the wrong claim is reduced to another wrong claim: $\tilde{W_{k+1}}(\vec{b}) = v'_{k+1}$.

So according to the above analysis, we proved the completeness and soundness, with the soundness error of $4n/|\mathbb{F}|$.

Exercise 1.e. Give an explicit expression of (i, j) entry of W_0. Then give a reduction from output state to the output of gate of layer 1.

Solution: According to the definition of matrix multiplication, we have:

$$W_0(i, j) = \sum_{k}^{N} M(i, k) * W_1(k, j)$$

This is again in a form of summation so we can utilize sumcheck protocol again.

First the prover claim:

$$\tilde{W}_0(i,j) = v_0$$

Then transform the equation about $\tilde{W}_0(i,j)$ into this form:

$$\tilde{W}_0(\vec{i},\vec{j}) = \sum_{\vec{k}\in\{0,1\}^n} \tilde{M}(\vec{i},\vec{k}) * \tilde{W}_1(\vec{k},\vec{j})$$

Here $\vec{\cdot}$ means the binary representation of the row or column index.

Then we further let polynomial $p(\vec{k}) = \tilde{M}(\vec{i},\vec{k}) * \tilde{W}_1(\vec{k},\vec{j})$ and we have:

$$\sum_{\vec{k}\in\{0,1\}^n} p(\vec{k}) = v_0$$

So we can apply sumcheck protocol to the above equation.

And sumcheck protocol will reduce the claim into this claim:

$$p(\vec{b}) = u_1$$

where $\vec{b}$ is generated randomly by the verifier during the sumcheck protocol processs and u_1 is the value, given by the prover, of polynomial p at this random point.

Then the verifier need to check whether this claim is valid or not so it requires the prover to provide the value of $\tilde{W}_1(\vec{b},\vec{j})$ so that it could calculate the value of $p(\vec{b}) = \tilde{M}(\vec{i},\vec{b}) * \tilde{W}_1(\vec{b},\vec{j})$.

Once the prover provides the value of $\tilde{W}_1(\vec{b},\vec{j}) = v_1$, the verifier could calculate $p(\vec{b})$ and check whether $p(\vec{b}) = u_1$. If not, it rejects; otherwise the problem is reduced to checking the provided value v_1 is correct or not:

$$\tilde{W}_1(\vec{b},\vec{j}) = v_1$$

So, we successfully reduce a claim about the MLE of the output state of the circuit to a claim about the MLE of the state of layer 1.

Exercise 1.f. Using the techs from the previous parts of the question to design an interactive proof to prove $\mathcal{X}, \mathcal{Y} \in \mathbb{F}^{N\times N}$ satisfies $C(\mathcal{X}) = \mathcal{Y}$, and state the communication complexity of the protocol.

Solution: Firstly we need an initial reduction, so that we can reduce many output checks into one. According to the uniqueness of MLE, we can equivalently check the MLE form of the output value. Also according to Schwarz Zippel Lemma, it's enough to check only the value at one random point of the two polynomials to make sure that the two polynomials are identical.

Then, for the chedck of output layer 0 of the circuit, we already have a reduction (in answer 1.e) to the value in the layer 1.

Then according to the answer in problem 1.d, we have a reduction from value in layer 1 to the value in layer 2.

Finally the verifier could use sumcheck protocol to reduce the claim of layer 2 to the claim of input layer, which can be trivially checked.

According to the equation in answer to problem 1.a:

$$W_2(\vec{z}) = \sum_{\vec{b}\in\{0,1\}^{2n}} Custom(\vec{z},\vec{b}) \cdot (W_3(\vec{b}) + K)^3$$

Then by the similar process, we transform it into:

$$v_2 = \tilde{W}_2(\vec{z}) = \sum_{\vec{b}\in\{0,1\}^{2n}} \tilde{\text{Custom}}(\vec{z},\vec{b}) \cdot (\tilde{W}_3(\vec{b}) + K)^3$$

Then conclude the RHS into this form:

$$\sum_{\vec{b}} p(\vec{b}) = v_2$$

where:

$$p(\vec{b}) = \tilde{\text{Custom}}(\vec{z},\vec{b}) \cdot (\tilde{W}_3(\vec{b}) + K)^3$$

Then we apply sumcheck protocol to the above summation equation and then the sumcheck protocol reduce it to a claim in this form:

$$p(\vec{s}) = \tilde{\text{Custom}}(\vec{z},\vec{s}) \cdot (\tilde{W}_3(\vec{s}) + K)^3 = u_3$$

where $\vec{s}$ is a random vector generated by verifier during sumcheck protocol and u_3 is the value provided by prover. Then the prover need to provide the value of $W_3(\vec{s}) = v_3$ and then the verifier checks whether $p(\vec{s}) = \tilde{\text{Custom}}(\vec{z},\vec{s}) \cdot (\tilde{W}_3(\vec{s}) + K)^3 = u_3$ using the value v_3.

Finally, the claim is reduced to layer 3: $W_3(\vec{s}) = v_3$, and the verifier should be able to check it by itself.

The communication complexity depends on each step in this proof.

The first step is initial reduction and it costs constant time because we only require verifier to send a random value.

The next step is reduction from layer 0 to layer 1. In answer 1.e we present such a reduction and the communication complexity is $O(n)$ because it basically just uses sumcheck protocol and the other communication cost is constant.

Then the next step is reduction from layer 1 to layer 2. In answer 1.d we present such a reduction and the communication complexity is still $O(n)$ because it's still basically a sumcheck protocol.

Then the final step is reduction from layer 2 to layer 3. All the same, still $O(n)$.

I didn't find anything related to $|\mathbb{F}|$ so I must miss something, but all the communication complexity that I can think of is $O(n)$. If we use the unit *bit* instead of the unit *an-F-element*, then the size of $\mathbb{F}$ would be of use because each time we transfer an element in $\mathbb{F}$, the larger *mathbbF* is, the more bits we will need for communication. So if we use the unit of *bit* then the communication cost would be $O(n \log |\mathbb{F}|)$.