Gemini-PCS (Part II)

- Tianyu ZHENG tian-yu.zheng@connect.polyu.hk

In the first part, we introduced the Tensor product check protocol in Gemini [BCH+22] for implementing multivariate polynomial evaluation proofs, and briefly explained how to apply it to practical proof systems to convert multivariate polynomials to univariate polynomials. In this part, we mainly focus on the security of the Tensor product check protocol and propose some optimizations based on Gemini.

Review

For ease of reading, let's first review the process of the tensor product check protocol:

[Tensor-product Check Protocol]

Target relation: $\langle \vec{f}, \otimes_{j=0}^{n-1}(1, \rho_j)\rangle = u$

Prover input: Public parameters, instance $x = (\rho_0, \ldots, \rho_{n-1}, u)$, secret $w = \vec{f}$

Verifier input: Public parameters, instance $x = (\rho_0, \ldots, \rho_{n-1}, u)$

1. The prover constructs a univariate polynomial $f^{(0)}(X) = f(X)$.
2. For $j \in 1, \ldots, n$, the prover computes

$$f^{(j)}(X) = f_e^{(j-1)}(X) + \rho_{j-1} \cdot f_o^{(j-1)}(X) \tag{1}$$

where $f_e^{(j-1)}$, $f_o^{(j-1)}$ are polynomials composed of even and odd order terms of $f^{(j-1)}$ respectively, satisfying $f^{(j-1)}(X) = f_e^{(j-1)}(X^2) + X \cdot f_o^{(j-1)}(X^2)$.

3. The prover sends Oracles of $f^{(0)}, f^{(1)}, \ldots, f^{(n-1)}$ to the verifier.
4. The verifier randomly selects a challenge value $\beta \leftarrow \mathbb{F}$ and makes the following queries to the Oracles:

$$e^{(j-1)} := f^{(j-1)}(\beta), \bar{e}^{(j-1)} := f^{(j-1)}(-\beta), \hat{e}^{(j-1)} := f^{(j)}(\beta^2) \tag{2}$$

where $j = 1, \ldots, n$, when $j = n$, ignore the query $f^{(n)}(\beta^2)$, and directly set $\hat{e}^{(n-1)} := u$.

5. For $j = 0, \ldots, n-1$, the verifier checks

$$\hat{e}^{(j)} = \frac{e^{(j)} + \bar{e}^{(j)}}{2} + \rho_j \cdot \frac{e^{(j)} - \bar{e}^{(j)}}{2\beta} \tag{3}$$

Security Analysis

[Completeness]

Given a polynomial coefficient vector satisfying $\langle \vec{f}, \otimes_{j=0}^{2}(1, \rho_j)\rangle = u$, a prover honestly executing the above protocol will certainly pass the verification. According to our discussion in the [Split-and-fold Check Protocol], when the prover correctly performs the split to obtain $f_e^{(j-1)}(X)$, $f_o^{(j-1)}(X)$, the folded $f^{(j)}(X)$ will definitely pass the verification, and $f^{(j)}(X)$ satisfies the following expression

$$f^{(j)}(X) = \sum_{i_0,\ldots,i_{n-1}\in\{0,1\}} f_{i_0\cdots i_{n-1}} \cdot \rho_0^{i_0} \cdots \rho_{j-1}^{i_{j-1}} \cdot X^{\langle \vec{i}_{[j:n-1]}, \vec{2}^{n-1-j}\rangle} \tag{4}$$

where $\vec{2}^{n-1-j} = (2^0, \ldots, 2^{n-1-j})$. Obviously, when $j = n$, $f^{(n)}(X) = \tilde{f}(\vec{\rho}) = u$.

[Soundness]

Here we adopt a different proof method from the original Gemini paper, which is easier to understand. Assume $\langle \vec{f}, \otimes_{j=0}^{2}(1, \rho_j)\rangle \neq u$, for a malicious prover, if it can output a series of interaction data, including Oracles of polynomials $f^{(j)}, j = 0, \ldots, n-1$, and pass the verification. Then there must be at least one pair of Oracles whose contained polynomials do not satisfy the split-and-fold relationship. That is, there exists a j such that

$$p_j(X) = f^{(j)}(X^2) - \frac{f^{(j-1)}(X) + f^{(j-1)}(-X)}{2} - \rho_{j-1}\frac{f^{(j-1)}(X) - f^{(j-1)}(-X)}{2X} \tag{5}$$

is a non-zero polynomial. Note that the highest order of $p_j(X)$ is $2^{n-(j-1)} - 1$. Let event E_j represent that $p_j(X)$ is a non-zero polynomial and $p_j(\beta) = 0$. According to the Schwartz-Zippel lemma, $Pr(E_j) \leq deg(p_j)/|F|$. Then, for all $p_1(X), \ldots, p_n(X)$, the probability that there exists a non-zero polynomial that happens to be 0 at point β can be bounded by the union bound

$$Pr(E_1 \vee \cdots \vee E_n) \leq Pr(E_1) + \cdots + Pr(E_n) = \sum_{j=1}^{n} \frac{deg(p_j)}{|F|} = \frac{2N}{|F|} \tag{6}$$

[About degree bound]

Note that in the original Gemini paper, the verifier needs to check that the order of each $f^{(j)}$ is strictly less than or equal to $2^{n-j} - 1$. This point is also reflected in the above proof: assuming that the highest order of $p_j(X)$ is $2^{n-(j-1)} - 1$.

However, after research, we find that the degree check is not necessary: even if a malicious prover is allowed to construct an illegal $f^{(j')}$ in each round, whose order is greater than the legal $f^{(j)}$ but less than or equal to N, we can still get a negligible soundness error (although slightly larger than the original). Specifically, for any E_j, we have $Pr(E_j) \leq N/|F|$, so we can get $Pr(E_1 \vee \cdots \vee E_n) \leq N \log N/|F|$ (when $N < D$, it's $D \log N/|F|$).

Therefore, the degree bound check in the Tensor-product check protocol based on KZG10 implemented in the first part can be ignored to reduce $\log N$ $\mathbb{G}_1$ elements in the proof.

Implementing Zero-Knowledge

Gemini did not discuss how to implement the ZK property of the tensor product check. Here we provide two feasible schemes.

[Scheme One]

Adopting a similar idea to implementing zk sumcheck in the paper [CAS17], we can directly add a blinding polynomial $g(X)$ of the same size to the original polynomial $f(X)$. That is, for each non-zero coefficient monomial in $f(X)$, $g(X)$ contains a corresponding monomial with a random coefficient. Let $\langle \vec{g}, \otimes_{j=0}^{n-1}(1, \rho_j)\rangle = v$.

Next, the prover only needs to additionally commit to $g(X)$ and send the commitment value $cm(g(X))$ and v to the verifier. The verifier then randomly selects a challenge c to combine the tensor product relations of f and g as

$$u + c \cdot v = \langle \vec{f} + c \cdot \vec{g}, \otimes_{j=0}^{n-1}(1, \rho_j) \rangle \tag{7}$$

Then, the prover and verifier jointly complete the Tensor product check protocol for the above relation. We won't elaborate further on the specific construction of this scheme.

[Scheme Two]

The above method is very straightforward, but the disadvantage is that the prover needs to add an additional random polynomial as large as $f(X)$ (length N). Referring to the optimization scheme of zk sumcheck in [CFS17] in Libra [XZZPS19], we can also propose an optimized scheme to implement the zk tensor product protocol, which can significantly reduce the size of the blinding polynomial.

The idea of this optimization scheme is: since the prover only sends a total of $3n$ point values in the tensor product check, the blinding polynomial needs to contain at least $3n$ random coefficients to ensure the zero-knowledge property of the protocol.

Specifically, let the blinding polynomial be:

$$g(X) = a_{0,1}X + a_{0,2}X^2 + \sum_{i=1}^{n-1}\left(a_{i,1}X^{3i} + a_{i,2}X^{3i+1} + a_{i,3}X^{3i+2}\right) + a_n \tag{8}$$

[Zero-Knowledge Tensor Product Check Protocol]

To prove $\langle \vec{f}, \otimes_{j=0}^{n-1}(1, \rho_j) \rangle = u$

1. The prover constructs a blinding polynomial $g(X)$ and pads its coefficient vector with zeros to length N
2. The prover computes and sends the following data to the verifier

$$\begin{aligned} v &= \langle \vec{g}, \otimes_{j=0}^{n-1}(1, \rho_j) \rangle \\ C_g &= cm(g(X)) \end{aligned} \tag{9}$$

3. The verifier randomly selects $c \in F^*$ and sends it to the prover, then computes $u + c \cdot v, C_f + c \cdot C_g$.
4. The prover and verifier run the tensor product check protocol to prove the following relation

$$u + c \cdot v = \langle \vec{f} + c \cdot \vec{g}, \otimes_{j=0}^{n-1}(1, \rho_j) \rangle \tag{10}$$

For convenience, let $h(X) = f(X) + c \cdot g(X)$. The proof that the above construction satisfies zero-knowledge is as follows:

[Proof]

First, the simulator S can be constructed according to the following steps:

1. S first inputs a random challenge value $c \neq 0$
2. S uniformly randomly generates vector $\vec{h}$ and computes polynomial $h(X)$, as well as $C_h = \mathsf{Commit}(h(X)), w = \langle \vec{h}, \otimes_{j=0}^{n-1}(1, \rho_j) \rangle$

3. S computes $v = (w - u)/c$ and commitment $C_g = (C_h / C_f)^{1/c}$
4. S runs a tensor product check protocol with V^*, where the proof relation is $w = \langle \vec{h}, \otimes_{j=0}^{n-1}(1, \rho_j) \rangle$.

Obviously, the messages in steps 1 and 3 are probabilistically indistinguishable from those sent by an honest prover P.

Next, we only need to show that the tensor product check protocol run by S and V^* in step 4 also satisfies this property. Specifically, because the protocol satisfies soundness, for each oracle $h^{(j)}, j = 0, \ldots, n-1$, it satisfies

$$h^{(j)}(X^2) = \frac{h^{(j-1)}(X) + f^{(j-1)}(-X)}{2} + \rho \cdot \frac{h^{(j-1)}(X) - h^{(j-1)}(-X)}{2X} \tag{11}$$

Note that for $h^{(j-1)}(X)$ on the right side of the equation, its corresponding oracle also satisfies an equation related to $h^{(j-2)}(X)$. Therefore, we can always expand the right expression satisfied by any $h^{(j)}(X^2)$ into a form that only includes $h^{(0)}(X), h^{(0)}(-X), h^{(0)}(X^2)$. Thus, the response obtained by V^* querying the oracle $h^{(j)}$ at any point β must be a linearly independent constraint on $\vec{h}$.

In summary, after performing the tensor product check protocol, V^* will obtain $3 \cdot (n-1)$, 2 values on $h^{(0)}(X)$, and one value of $h^{(n)}(X)$, i.e., $u + c \cdot v$. Because h contains a blinding polynomial of size $3n$, the verifier cannot interpolate to obtain all coefficients of the blinding polynomial, so this protocol is indistinguishable from the protocol executed by an honest verifier.

References

[BCH+22] Bootle, Jonathan, Alessandro Chiesa, Yuncong Hu, **et al. "Gemini: Elastic SNARKs for Diverse Environments." *Cryptology ePrint Archive* (2022). https://eprint.iacr.org/2022/420

[CFS17] Chiesa, Alessandro, Michael A. Forbes, and Nicholas Spooner. "A zero knowledge sumcheck and its applications." *arXiv preprint arXiv:1704.02086* (2017).

[XZZPS19] Xie, T., Zhang, J., Zhang, Y., Papamanthou, C., & Song, D. "Libra: Succinct zero-knowledge proofs with optimal prover computation." https://eprint.iacr.org/2019/317