Hashcaster

Yue Ying

December 27, 2024

Overview

Preliminary

Some important definitions

- Define the base field F_p and its extension field F_q, where $q = p^n$ and p is a prime. F_{p^n} is a finite field of characteristic p.
- Definition of a polynomial over the extension field:

$$P(x) = \sum_{i=0}^{n-1} b_i P_i(x),$$

  where b_i are elements of a basis of the extension field, and $P_i(x)$ are coordinate polynomials defined over the base field.
- Definition of the Frobenius map:

$$F(x) = x^p, \quad \text{for } x \in F_q.$$

  The Frobenius map is a linear transformation(proof is provided in the appendix):
  1. $F(x + y) = F(x) + F(y)$, for all $x, y \in F_q = F_{p^n}$.
  2. $F(c \cdot x) = c \cdot F(x)$, for all $x \in F_q = F_{p^n}$ and $c \in F_p$.

Some Important Definitions

- Definition of the Frobenius map:

$$F(x) = x^p, \quad \text{for } x \in F_q.$$

  The Frobenius map is a linear transformation (proof is provided in the appendix):
  1. $F(x+y) = F(x) + F(y)$, for all $x, y \in F_q = F_{p^n}$.
  2. $F(c \cdot x) = c \cdot F(x)$, for all $x \in F_q = F_{p^n}$ and $c \in F_p$.
- Multiple Frobenius mappings:

$$F^k(x) = x^{p^k}, \quad \text{with the inverse mapping defined as } F^{-k}(x) = x^{p^{-k}}.$$

- Definition of Frobenius twists: applying the Frobenius mapping to the basis of a polynomial over the extension field:

$$P_{\gg k}(x) = \sum (b_i^{p^k}) P_i(x),$$

  where $b_i^{p^k}$ denotes the Frobenius mapping applied to the coefficients b_i, and $P_i(x)$ are the coordinate polynomials over the base field.

Using Frobenius Twists to efficiently get $P_i(x)$

- Objective of Frobenius twists in our setting: to efficiently decompose a polynomial $P(x)$ defined over the extension field F_q into its coordinate polynomials $P_i(x)$
- Apply a series of Frobenius twists to $P(x)$, resulting in $P_{\gg 0}(x), P_{\gg 1}(x), \ldots, P_{\gg d-1}(x)$. Each twist is defined as:

$$P_{\gg i}(x) = \sum F^i(b_i)P_i(x) = \sum b_i^{p^i} P_i(x),$$

By expanding each twist, a system of linear equations is constructed:

$$\begin{aligned}
P_{\gg 0}(x) &= F^0(b_0)P_0(x) + F^0(b_1)P_1(x) + \cdots + F^0(b_{d-1})P_{d-1}(x), \\
P_{\gg 1}(x) &= F^1(b_0)P_0(x) + F^1(b_1)P_1(x) + \cdots + F^1(b_{d-1})P_{d-1}(x), \\
&\vdots \\
P_{\gg d-1}(x) &= F^{d-1}(b_0)P_0(x) + F^{d-1}(b_1)P_1(x) + \cdots + F^{d-1}(b_{d-1})P_{d-1}(x).
\end{aligned}$$

- Since the basis $\{b_0, b_1, \ldots, b_{d-1}\}$ of the extension field is linearly independent and the coefficient matrix of the system is full rank, the solution to the equations is unique, ensuring that each $P_i(x)$ can be uniquely determined.

Opening $P(x)$ on an Extension Field Element r

- Given $P(r)$ and r, how to quickly deducing $P_i(r)$?
- Trivial approach: First compute $P_i(x)$ from $P(x)$ using the method described on the previous slide, and then evaluate $P_i(r)$ for each component polynomial.
- Improvement:
  Key observation: Reduce the problem of opening P_i at a single point r to multiple openings of P at points in the inverse Frobenius orbit of r. Specifically:

  $$F^i(P(F^{-i}(r))) = P_{\gg i}(r),$$

  where F^i represents the i-th Frobenius mapping and F^{-i} its inverse. A simple derivation is as follows (detailed proof is provided in the appendix):

  $$F^i(P(F^{-i}(r))) = \sum_{j=0}^{d-1} F^i(b_j)F^i(P_j(F^{-i}(r))) = \sum_{j=0}^{d-1} F^i(b_j)P_j(r) = P_{\gg i}(r).$$

Opening $P(x)$ on an Extension Field Element r

- LHS:$F^i(P(F^{-i}(r)))$
  1. Compute the inverse Frobenius orbit of r:
  $$[r, F^{-1}(r), F^{-2}(r), \ldots, F^{-(d-1)}(r)] = [r, r^{p^{-1}}, r^{p^{-2}}, \ldots, r^{p^{-(d-1)}}].$$
  2. Evaluate $P(x)$ at the points in the inverse Frobenius orbit:
  $$P(F^{-i}(r)), \quad \text{for } i = 0, 1, \ldots, d-1.$$
  3. Compute Frobenius mappings on those evaluations and form the vector:
  $$\mathbf{b} = \begin{bmatrix} F^0(P(F^0(r))) \\ F^1(P(F^{-1}(r))) \\ F^2(P(F^{-2}(r))) \\ \vdots \\ F^{d-1}(P(F^{-(d-1)}(r))) \end{bmatrix}.$$

Opening $P(x)$ on an Extension Field Element r

- RHS: $P_{\gg i}(r)$

$$
\begin{aligned}
P_{\gg 0}(r) &= F^0(b_0)P_0(r) + F^0(b_1)P_1(r) + \cdots + F^0(b_{d-1})P_{d-1}(r), \\
P_{\gg 1}(r) &= F^1(b_0)P_0(r) + F^1(b_1)P_1(r) + \cdots + F^1(b_{d-1})P_{d-1}(r), \\
&\vdots \\
P_{\gg d-1}(r) &= F^{d-1}(b_0)P_0(r) + F^{d-1}(b_1)P_1(r) + \cdots + F^{d-1}(b_{d-1})P_{d-1}(r).
\end{aligned}
$$

1. Construct the matrix $\mathbf{A}$: Each row corresponds to the Frobenius mappings of the basis elements b_i, defined as:

$$
\mathbf{A} = \begin{bmatrix} F^0(b_0) & F^0(b_1) & \cdots & F^0(b_{d-1}) \\ F^1(b_0) & F^1(b_1) & \cdots & F^1(b_{d-1}) \\ \vdots & \vdots & \ddots & \vdots \\ F^{d-1}(b_0) & F^{d-1}(b_1) & \cdots & F^{d-1}(b_{d-1}) \end{bmatrix}.
$$

2. Construct the vector x:

$$
\mathbf{x}^T = \begin{bmatrix} P_0(r) & P_1(r) & \cdots & P_{d-1}(r) \end{bmatrix}.
$$

Opening $P(x)$ on an Extension Field Element r

- RHS: $P_{\gg i}(r)$

$$
\begin{aligned}
P_{\gg 0}(r) &= F^0(b_0)P_0(r) + F^0(b_1)P_1(r) + \cdots + F^0(b_{d-1})P_{d-1}(r), \\
P_{\gg 1}(r) &= F^1(b_0)P_0(r) + F^1(b_1)P_1(r) + \cdots + F^1(b_{d-1})P_{d-1}(r), \\
&\vdots \\
P_{\gg d-1}(r) &= F^{d-1}(b_0)P_0(r) + F^{d-1}(b_1)P_1(r) + \cdots + F^{d-1}(b_{d-1})P_{d-1}(r).
\end{aligned}
$$

4. Construct the matrix $\mathbf{A}$: Each row corresponds to the Frobenius mappings of the basis elements b_i
5. Construct the vector x
6. Solve the linear system in matrix form:

$$\mathbf{A}\mathbf{x} = \mathbf{b}$$

Since $\mathbf{A}$ is full-rank due to the linear independence of the basis $\{b_0, b_1, \ldots, b_{d-1}\}$, the solution $\mathbf{x}$ is unique.

ANDCHECK

The Setting of the Problems

- The setting of the problem is as follows:
  - Base Field: $\mathbb{F}_p$, where $p = 2$ (binary field).
  - Extension Field: $\mathbb{F}_q$, where $q = p^n = 2^n$.
- Challenge: For all $x, y \in \mathbb{F}_{2^n}$, performing a **bitwise** $\wedge$ (logical AND) operation is not directly supported in common prime fields.
- Proposed Solution: Using Frobenius mapping, we map the $\wedge$ operation in the extension field to arithmetic operations in the base field:

$$x \wedge y = \sum_{i=0}^{n-1} b_i \cdot x_i \cdot y_i,$$

where:
  - x_i and y_i are the projections of x and y along the b_i-basis directions, i.e., $x = \sum_{i=0}^{n-1} x_i b_i$ and $y = \sum_{i=0}^{n-1} y_i b_i$.
  - $x_i, y_i \in \mathbb{F}_2$. In $\mathbb{F}_2$, arithmetic multiplication is equivalent to the logical $\wedge$ operation:

$$x_i \cdot y_i = x_i \wedge y_i.$$

The Objective of the Whole System

- Perform the sumcheck of the form:

$$\sum_{x \in B^n} (P \wedge Q)(x) \cdot eq(x; q),$$

where $(P \wedge Q)(x) = \sum_{i=0}^{n} b_i \cdot P_i(x) \cdot Q_i(x)$.

Optimization on Sumcheck

Trick 1: drops the degree in t under summation to 2

- For the i-th round where $i \leq n$, the prover sends a polynomial of the form:

$$U(t) = \sum_{x_{>i} \in \mathbb{B}^{n-i-1}} (P \wedge Q)(r_{<i}, t, x_{>i}) \cdot eq(r_{<i}, t, x_{>i}; q),$$

which is a cubic polynomial. can be reaaranged as

$$U_i(t) = eq(r_{<i}, q_{<i}) eq(t, q_i) \sum_{x_{>i} \in \mathbb{B}^{n-i-1}} (P \wedge Q)(r_{<i}, t, x_{>i}) \times (x_{>i}, q_{>i})$$

which drops the degree in t under summation to 2.

Trick 2: Perform in Two Phases using different strategies

- The process involves two phases:
  - The first phase runs for c rounds.
  - The second phase runs for $n - c$ rounds.
- In the first c rounds:
  1. start by computing the table of evaluations of P, Q in the extended size $(0, 1, \infty)^{c+1} \times (0, 1)^{n-c-1}$
  2. evaluate $P \wedge Q$ on this whole subset
  3. Prover enters the first round and does the following:
     - Compute $\sum (P \wedge Q)(r_{<i}, t, x_{>i}) \times eq(x_{>i}, q_{>i})$: lookup table and compute evaluations of $(P \wedge Q)(t, x_{>0})$ for $t \in \{0, 1, \infty\}$, and add them up multiplied by $eq(x_{>0}, q_{>0})$
     - Obtain Obtain challenge r_0 from verifier.
     - Update table according to r_0

Phase 1 in details

1. The prover maintains two arrays, say A and B, which initially store all evaluations of P and Q over $\{0,1\}^n$. We will index entries of A and B by $x \in \{0,1\}^n$, so that at initialization, A[x] stores P(x) and B[x] stores Q(x).
2. Based on A[x] and B[x], prover computes an extending table storing all evaluations of P and Q over $\{0,1,\infty\}^{c+1} \times \{0,1\}^{n-c-1}$

Index	**P(x)**
First $c+1$ bits: $0,0,\ldots,0,$ Last $n-c-1$ bits: $0,0,\ldots,0$	$P(0,0,\ldots,0,0,0,\ldots,0)$ (from A[x])
First $c+1$ bits: $1,0,\ldots,0,$ Last $n-c-1$ bits: $0,0,\ldots,0$	$P(1,0,\ldots,0,0,0,\ldots,0)$ (from A[x])
First $c+1$ bits: $\infty,0,\ldots,0,$ Last $n-c-1$ bits: $0,0,\ldots,0$	$P(\infty,0,\ldots,0,0,0,\ldots,0)$ (computed from $P(0,0,\ldots)+P(1,0,\ldots)$)

This table has $3^{c+1}\cdot 2^{n-c-1}$ rows, representing all combinations of elements from $\{0,1,\infty\}^{c+1} \times \{0,1\}^{n-c-1}$.

Phase 1 in details

Notes on computing $P(x)$:

- If the index is a pure bitstring (without ∞), the evaluation is directly copied from table $A[x]$.
- If the index contains ∞, the new value is computed by summing (or XORing in $\mathbb{F}_2$) two rows in the extended table.
- The two rows to be summed are determined as follows:
  1. Locate the rightmost ∞ in the index.
  2. Replace this position with 0 and 1 while keeping all other positions unchanged.
  3. The resulting two indices correspond to the rows to be summed.
- For example:

$$P(\infty,0,0,\dots,0,0,\dots,0) = P(0,0,0,\dots,0,0,\dots,0) + P(1,0,0,\dots,0,0,\dots,0)$$

$$P(\infty,0,0,\dots,\infty,0,\dots,0) = P(\infty,0,0,\dots,0,0,\dots,0) + P(\infty,0,0,\dots,1,0,\dots,0)$$

$Q(x)$ values are computed similarly to $P(x)$, but using $B[x]$ instead of $A[x]$.

Constructing the extended table requires $3^{c+1} \cdot 2^{n-c-1} - 2^n$ additions (or XOR) and 2^n copy-paste operations.

Phase 1 Details

3. Compute $P \wedge Q$ based on the $P(x)$ and $Q(x)$ columns. The extended table is structured as follows:

Index	**P(x)**	**Q(x)**	**P $\wedge$ Q**
...	...	...	...

Calculating $P \wedge Q$ **requires** $3^{c+1} \cdot 2^{n-c-1}$ $\wedge$ **operations.**

Phase 1 Details

4. Prover enters Round 1:
   - Compute valuations of $(P \wedge Q)(t, x_{>0})$ for $t \in \{0, 1, \infty\}$:
     - fix t
     - look up extending table to get $(P \wedge Q)(t, x_{>0})$ for all $x \in 0, 1^{n-1}$
     - multiplied by corresponding $eq(x_{>0}, q_{>0})$
     - add them up

   This requires $3 * 2^{n-1}$ multiplications
   - Obtain challenge r_0 from verifier
   - Update table according to r_0 Update table A according to r_0:

$$A[x] \leftarrow (1 - r_0) \cdot A[0, x] + r_0 \cdot A[1, x] = A[0, x] + r_0 \cdot (A[1, x] - A[0, x])$$

   Similarly, update table B according to r_0:

$$B[x] \leftarrow (1 - r_0) \cdot B[0, x] + r_0 \cdot B[1, x] = B[0, x] + r_0 \cdot (B[1, x] - B[0, x])$$

   Update the extended table based on the updated A and B. **This step, including updates to A, B, and the extended table, requires $3^{c+1} \cdot 2^{n-c-1}$ multiplications in total.**

Trick 2: Perform in Two Phases using different strategy

- In the last $n - c$ rounds, the strategy changes to maintain polynomials:

$$P_i(r_0, \ldots, r_c, x_{>c}), \quad Q_i(r_0, \ldots, r_c, x_{>c}).$$

- In the final round, when opening $U(r)$, Frobenius Twists are used to compute $P_i(r)$ and $Q_i(r)$. Additionally, the **4 Russians method** is employed for the $eq(r, q)$ multi-open operation:

$$U(r) = (P \wedge Q)(r) \cdot eq(r, q).$$

Appendix

Proof: Frobenius Mapping is a Linear Transformation

Let $F : \mathbb{F}_{q^n} \to \mathbb{F}_{q^n}$ be the Frobenius mapping defined by $F(x) = x^q$, where $\mathbb{F}_{q^n}$ is a finite field of characteristic p and order q^n. We aim to prove that F is a linear transformation over the field $\mathbb{F}_q$.

- **Additivity:** For any $a, b \in \mathbb{F}_{q^n}$,

  $$F(a + b) = (a + b)^q.$$

  Using the Binomial Theorem,

  $$(a + b)^q = \sum_{k=0}^{q} \binom{q}{k} a^k b^{q-k}.$$

  Since $q = p^m$ for some $m \geq 1$ and the characteristic of the field is p, all binomial coefficients $\binom{q}{k}$ are divisible by p for $1 \leq k \leq q - 1$. Thus,

  $$F(a + b) = a^q + b^q = F(a) + F(b).$$

Proof: Frobenius Mapping is a Linear Transformation

- **Homogeneity:** For any $c \in \mathbb{F}_q$ and $a \in \mathbb{F}_{q^n}$,

$$F(ca) = (ca)^q = c^q a^q.$$

  Since $c \in \mathbb{F}_q$, we have $c^q = c$ (Fermat's Little Theorem). Hence,

$$F(ca) = cF(a).$$

Conclusion: F satisfies additivity and homogeneity, proving that F is a linear transformation over $\mathbb{F}_q$.

Proof: $F^i(P_j(F^{-i}(r))) = P_j(r)$

- Let $P_j(x)$ be a polynomial defined as:

$$P_j(x) = c_0 + c_1x + c_2x^2 + \cdots + c_nx^n, \quad c_i \in \mathbb{F}_p.$$

- Start by evaluating $P_j(F^{-i}(r))$:

$$P_j(F^{-i}(r)) = c_0 + c_1\left(F^{-i}(r)\right) + c_2\left(F^{-i}(r)\right)^2 + \cdots + c_n\left(F^{-i}(r)\right)^n.$$

- Apply the Frobenius map F^i to $P_j(F^{-i}(r))$:

$$F^i\left(P_j(F^{-i}(r))\right) = F^i\left(c_0 + c_1F^{-i}(r) + c_2\left(F^{-i}(r)\right)^2 + \cdots + c_n\left(F^{-i}(r)\right)^n\right).$$

- Using the linearity of the Frobenius map $F^i(x+y) = F^i(x) + F^i(y)$ and its action on powers $F^i(x^k) = (F^i(x))^k$:

$$F^i\left(P_j(F^{-i}(r))\right) = F^i(c_0) + F^i(c_1)F^i(F^{-i}(r)) + F^i(c_2)\left(F^i(F^{-i}(r))\right)^2 + \cdots + F^i(c_n)\left(F^i(F^{-i}(r)\right.$$

Proof: $\mathrm{F}^i(P_j(F^{-i}(r))) = P_j(r)$

- Since $F^i(F^{-i}(x)) = x$, this simplifies to:

$$F^i\left(P_j(F^{-i}(r))\right) = F^i(c_0) + F^i(c_1)r + F^i(c_2)r^2 + \cdots + F^i(c_n)r^n.$$

- Now, note that $c_i \in \mathbb{F}_p$, and for any $c_i \in \mathbb{F}_p$, $F^i(c_i) = c_i$. Therefore:

$$F^i\left(P_j(F^{-i}(r))\right) = c_0 + c_1 r + c_2 r^2 + \cdots + c_n r^n.$$

- This is exactly $P_j(r)$.

Conclusion: We have shown that:

$$F^i(P_j(F^{-i}(r))) = P_j(r).$$

The End