Efficient Primitive Protocols for Sharemind

Bingsheng Zhang[1,2]

[1]Cybernetica AS, Estonia
[2]University of Tartu, Estonia

Estonian Theory Days in Elva,
11.06.2010-13.06.2010

Outline

- Division in $\mathbb{Z}^*_{2^{32}}$
- Multiplication in $\mathbb{Z}^*_{2^{32}}$
- High degree Conjunction
- Random Shuffle Protocol

Division in $\mathbb{Z}^*_{2^{32}}$

Division in $\mathbb{Z}^*_{2^{32}}$

Server's input: $[[A]] \in \mathbb{Z}^*_{2^{32}}$ and $[[B]] \in \mathbb{Z}^*_{2^{32}}$

Server's output: $[[C]] \in \mathbb{Z}^*_{2^{32}}$, where $C = A \cdot B^{-1}$

1. Each miner $\mathcal{M}_{p\in\{0,1,2\}}$ generates a random number $R_p \leftarrow_u \{1, 2, \cdots, 2^{31}\}$. Set $R'_p = 2 \cdot R_p - 1$.
2. All miners $\mathcal{M}_{p\in\{0,1,2\}}$ compute and open $[[D]] = [[B]] \cdot [[R']]$.
3. All miners $\mathcal{M}_{p\in\{0,1,2\}}$ compute and set $[[C]] = D^{-1} \cdot [[R']] \cdot [[A]]$.

The total protocol costs 3 rounds.

Multiplication in $\mathbb{Z}^*_{2^{32}}$

Generating Random Invertible Pairs

Server's input: $\perp$

Server's output: Data shares in $\mathbb{Z}_{2^{32}}$: $[[R \leftarrow_u \mathbb{Z}^*_{2^{32}}]]$ and $[[R^{-1}]]$

1. Each miner $\mathcal{M}_{p\in\{0,1,2\}}$ generates two random number $A_p \leftarrow_u \{1,2,\cdots,2^{31}\}$ and $B_p \leftarrow_u \{1,2,\cdots,2^{31}\}$. Set $R_p = 2\cdot A_p - 1$ and $R'_p = 2\cdot B_p - 1$
2. All miners $\mathcal{M}_{p\in\{0,1,2\}}$ compute and open $[[C]] = [[R]]\cdot[[R']]$.
3. Each miner $\mathcal{M}_{p\in\{0,1,2\}}$ computes and sets $[[R^{-1}]] = C^{-1}\cdot[[R']]$.

The total protocol costs 2 rounds.

Unbounded Fan-in Multiplication

Server's input: Data shares in $\mathbb{Z}^*_{2^{32}}$: $[[X_1]], \cdots, [[X_k]]$

Server's output: Data shares in $\mathbb{Z}^*_{2^{32}}$: $[[\prod_{i=1}^{k} X_i]]$

1. All miners $\mathcal{M}_{p\in\{0,1,2\}}$ generate random invertible pairs $([[R_0]], [[R_0^{-1}]]), \cdots, ([[R_k]], [[R_k^{-1}]])$ by using sub-protocol in previous section.
2. For $i \in \{1, \cdots, k\}$, all miners $\mathcal{M}_{p\in\{0,1,2\}}$ compute and open $[[A_i]] = [[R_{i-1}]] \cdot [[X_i]] \cdot [[R_i^{-1}]]$.
3. Each miner $\mathcal{M}_{p\in\{0,1,2\}}$ computes $B = \prod_{i=1}^{k} A_i \quad (= R_0 \cdot \prod_{i=1}^{k} X_i \cdot R_k^{-1})$.
4. All miners $\mathcal{M}_{p\in\{0,1,2\}}$ compute $[[S]] = [[R_0^{-1}]] \cdot B \cdot [[R_k]]$.

The total protocol costs $3 + 2$ rounds.

High degree Conjunction

Outline
High degree Conjunction

Server's input: $[[X_1]], \cdots, [[X_k]]$ $(X_i \in \{0, 1\})$

Server's output: $[[Y]] = [[X_1 \wedge \cdots \wedge X_k)]]$

1. All miners $\mathcal{M}_{p \in \{0,1,2\}}$ computes $[[S]] = \sum_{i=1}^{k} [[X_i]]$.
2. All miners $\mathcal{M}_{p \in \{0,1,2\}}$ call Equal sub-protocol to check if $[[S]] = k$ and return the result bit as $[[Y]]$.

This protocol was improved by Margus Niitsoo's comments. It takes the same rounds as equality check protocol, which is 7 rounds. In theory, it is $O(\log \log k)$ rounds protocol, where k is the degree.

Random Shuffle Protocol

Random Shuffle Protocol

- For $i \in \{1, \cdots, k \log n\}$, all miners $\mathcal{M}_{p\in\{0,1,2\}}$ do:
  - Split documents into shared bits, for each bit do:
    1. Generate a random number with length n, and split R_p to bits, denoting as $R_p[0], \cdots, R_p[n-1]$.
    2. Call one bit share conversion sub-protocol to compute additive shares: $b_p[0], \cdots, b_p[n-1]$.
    3. Call (m, n)-PIR sub-protocol to select 'documents' to set 1 according to bits $b[j]$ (, $b[j] = 1$ means select,) and to select the rest 'documents' to set 0 according to bits $(1 - b[j])$.
  - All miners $\mathcal{M}_{p\in\{0,1,2\}}$ combine the shared documents from shared bits.

$O(k \log^2 n)$ rounds, given that (m, n)-PIR sub-protocol takes $\log n$ rounds. $O(n \log n)$ computation, given that (m, n)-PIR sub-protocol costs $O(n)$ computation regardless m.

Random Shuffle Protocol

(m,n)-PIR Protocol

Client's input: $\overrightarrow{B} = \{b_0, \cdots, b_{n-1}\}$.

- For $j \in \{1, \cdots, n-1\}$, all miners $\mathcal{M}_{p\in\{0,1,2\}}$ do:
  1. Compute $[[Q_j]] = \sum_{w=0}^{j-1}[[b_j]]$.
  2. Call bit decomposition sub-protocol to split $[[Q_j]]$ into bits, denoting as $[[Q_j[0]]], \cdots, [[Q_j[t]]]$, where $t = \lfloor \log n \rfloor$.
  3. Set $[[s_0]] = [[b_0]]$. Compute and set $[[s_j]] = [[b_j]] \cdot \prod_{v=0}^{t}([[Q_j[v]]] \cdot 2^{2^v})$. (i.e. $[[s_j]] = [[b_j]] \cdot 2^{[[Q_j]]}$.)
- Denote the selection vector $[[\overrightarrow{S}]] = \{[[s_0]], \cdots, [[s_{n-1}]]\}$ and document vector $[[\overrightarrow{D[k]}]] = \{[[d_0[k]]], \cdots, [[d_{n-1}[k]]]\}$. For $k \in \{0, \cdots, |D|-1\}$, all miners compute $[[R[k]]] \leftarrow [[\overrightarrow{S}]] \cdot [[\overrightarrow{D[k]}]]^T$. Then split $[[R[k]]]$ to bits.

Random Shuffle Protocol

Perfect Random Shuffle Protocol

1. For set size $i = \{n, n/2, n/2^2, \cdots, 1\}$, all miners $\mathcal{M}_{p\in\{0,1,2\}}$ do:
    1. Create array $A_p[i] = 0$. Pick $i/2$ positions randomly, and set them to 1. Now $A_p[i]$ can be regarded as random number with hamming weight exactly $i/2$.
    2. For $u = 0, 1, 2$, miner $\mathcal{M}_u$ shares $A_u[i]$ bitwisely: $[[b[0]]], \cdots, [[b[i-1]]]$.
        1. Call (m,n)-PIR sub-protocol to select 'documents' to set 1 according to bits $b[j]$ (, $b[j] = 1$ means select,) and to select the rest 'documents' to set 0 according to bits $(1-b[j])$.
    3. All miners $\mathcal{M}_{p\in\{0,1,2\}}$ will execute recursively in a parallel for sets 0 and 1 for next round.

Random Shuffle Protocol

Thank You!

Questions?