Highly optimized 3-party protocols for Sharemind

Margus Niitsoo, Tomas Toft,
Jan Willemson, Sven Laur, Dan Bogdanov

University of Tartu, Cybernetica AS,
University of Aarhus

June 12, 2010

Outline

Sharemind

- Multi-party computation
- Over the ring $\mathbb{Z}_{2^{32}}$.
- 3 parties $\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3$.
- Additive secret sharing $u = u_1 + u_2 + u_3$.

Sharemind Security

Additive secret sharing $u = u_1 + u_2 + u_3$.

- Shares are formed so that $u_1, u_2 \leftarrow \mathbb{Z}_{2^{32}}$ and $u_3 = u - u_1 - u_2$.
- All shares are distributed uniformly in $\mathbb{Z}_{2^{32}}$.
- It takes all three shares to reconstruct the original.
    - Knowing just two will give no information about u.
    - Guarantees information-theoretic security.

Security Model

Additive secret sharing $u = u_1 + u_2 + u_3$.

- When storage is in a passive state, all three need to be corrupted.
- During protocols, tolerate one corrupt party.
- Corruption needs to be passive
  - Adversary follows protocol
  - 'Honest-but-curious'
- We assume secure authenticated channels.
  - Party sees only his own values and what is sent to him

Protocol Design for Sharemind

- Main idea: if $r \leftarrow \mathbb{Z}_{2^{32}}$ is uniformly random, it is safe to send $u - r$ to one peer and r to the other.
  - Both peers receive a uniformly distributed value.
  - Leaks no information as both are independent from u.

Du Atallah Multiplication

- $\mathcal{P}_1$ has u, $\mathcal{P}_2$ has v. End result is a sharing of uv.
  1. The party $\mathcal{P}_3$ uniformly generates $r_1, r_2 \leftarrow \mathbb{Z}_N$ and sends r_1 to $\mathcal{P}_1$ and r_2 to $\mathcal{P}_2$.
  2. $\mathcal{P}_1$ computes $u - r_1$ and sends the result to $\mathcal{P}_2$. At the same time $\mathcal{P}_2$ computes $v - r_2$ and sends the result to $\mathcal{P}_1$.
  3. Now the parties have enough information to compute shares of the product uv:
     - $\mathcal{P}_1$ computes its share $p_1 = (u - r_1)(v - r_2) + u(v - r_2)$
     - $\mathcal{P}_2$ computes its share $p_2 = v(u - r_1)$
     - $\mathcal{P}_3$ computes its share $p_3 = -r_1 r_2$
- To multiply two shared values, note $(u_1 + u_2 + u_3)(v_1 + v_2 + v_3) = \sum_{i,j} u_i v_j$.

Let's simplify

- $\mathcal{P}_1$ has u, $\mathcal{P}_2$ has v. End result is a sharing of $x_1 x_2$.
  1. $\mathcal{P}_1$ generates $r_1 \leftarrow \mathbb{Z}_{2^{32}}$ and $\mathcal{P}_2$ generates $r_2 \leftarrow \mathbb{Z}_{2^{32}}$.
  2. $\mathcal{P}_1$ sends r_1 to $\mathcal{P}_2$ and $u - r_1$ to $\mathcal{P}_3$. At the same time $\mathcal{P}_2$ sends r_2 to $\mathcal{P}_1$ and $v - r_2$ to $\mathcal{P}_3$.
  3. Now the parties have enough information to compute shares of the product uv:
     - $\mathcal{P}_1$ computes its share $p_1 = (u - r_1)r_2$
     - $\mathcal{P}_2$ computes its share $p_2 = vr_1$
     - $\mathcal{P}_3$ computes its share $p_3 = (u - r_1)(v - r_2)$
- Note this works in 1 round instead of 2.
- This scales better to a share multiplication

Share Conversion

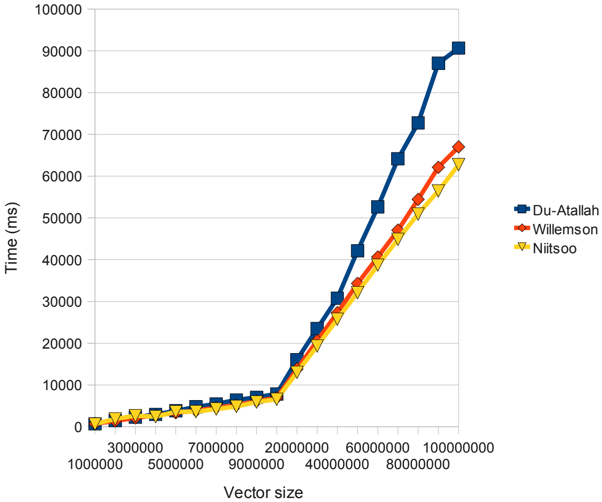
- Parties have a shared bit $u = u_1 \oplus u_2 \oplus u_3$. End result is a sharing of u in $\mathbb{Z}_{2^{32}}$.
  1. $\mathcal{P}_3$ generates a random $m \leftarrow \mathbb{Z}_{2^{32}}$ and $b_1, b_2 \leftarrow \mathbb{Z}_2$. He sends b_1 and $m' = b_1 \oplus b_2 \oplus u_3 - m$ to $\mathcal{P}_1$. In parallel, he sends b_2 and m to $\mathcal{P}_2$.
  2. $\mathcal{P}_1$ sends $b_1 \oplus u_1$ to $\mathcal{P}_2$. $\mathcal{P}_2$ sends $b_2 \oplus u_2$ to $\mathcal{P}_1$.
  3. $\mathcal{P}_1, \mathcal{P}_2$ compute $r = b_1 \oplus b_2 \oplus u_1 \oplus u_2$.
  4. Result $p_1 = m'$ and $p_2 = m$ if $r = 0$ and $p_1 = 1 - m'$ and $p_2 = -m$ otherwise.

Recent Progress

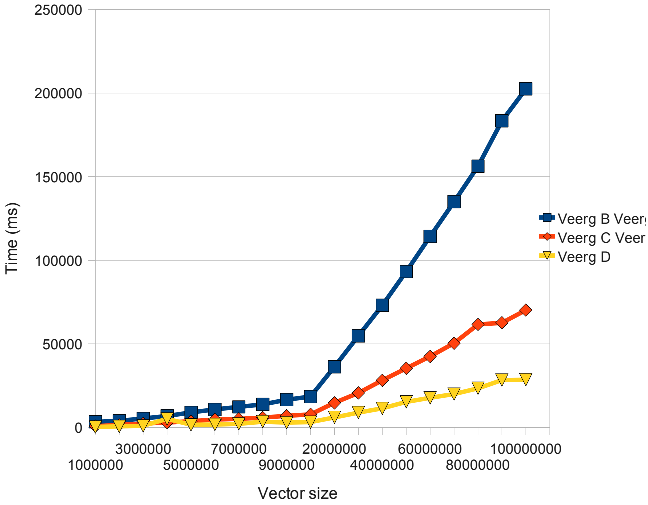
A whole set of new protocols:

- Improved multiplication and share conversion protocols
- Improved Bit Extraction, Comparison and Equality protocols
- New primitives
  - Shift right with a public value
  - Division and Modulo reduction with a public value
- Outline for a division protocol with a secret value.

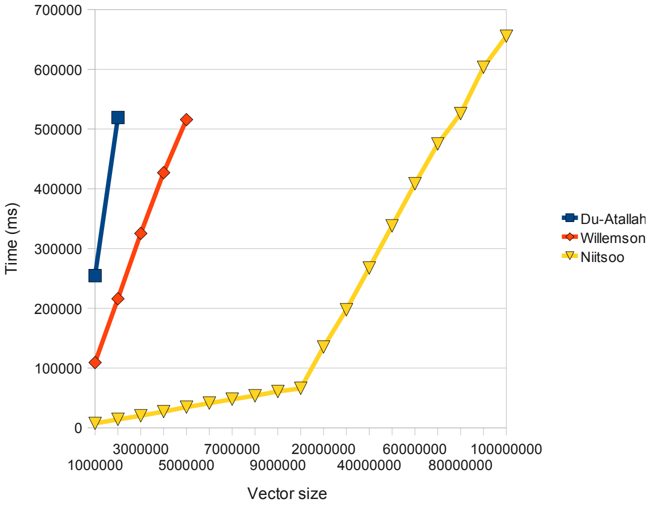
Multiplication



Share Conversion



Integer Comparison



Thank You!

Thank you for attention! Any questions are welcome!