Near-Linear Unconditionally-Secure MPC with a Dishonest Minority

Serge Fehr
CWI Amsterdam
www.cwi.nl/~fehr

Eli Ben-Sasson
Technion

Rafail Ostrovsky
UCLA
Multiparty Computation (MPC)

Goal:
Compute function $f$ on private inputs $x_1, \ldots, x_n$, so that

- all learn correct $f(x_1, \ldots, x_n)$
- $x_i$’s remain private

even if adversary corrupts $t$ players.

Classical possibility results:

- computational security for $t < n/2$ [GMW87,CDG88]
- unconditional security for $t < n/2$ (assuming broadcast) [RB89,Bea89]
- perfect security for $t < n/3$ [CCD88,BGW88]

Beyond (im)possibility results: (communication) complexity
**Best known results (binary circuits):**

<table>
<thead>
<tr>
<th>Attack</th>
<th>Resilience</th>
<th>Security</th>
<th>Bits/multiplication</th>
<th>Ref</th>
</tr>
</thead>
<tbody>
<tr>
<td>passive</td>
<td>$t &lt; n/2$</td>
<td>perfect</td>
<td>$O(n \log n)$</td>
<td>[DamNie07]</td>
</tr>
<tr>
<td>active</td>
<td>$t &lt; n/2$</td>
<td>computational</td>
<td>$O(n \log n)$</td>
<td>[DamNie07]</td>
</tr>
<tr>
<td>active</td>
<td>$t &lt; n/2$</td>
<td>unconditional</td>
<td>$O(n^2 k)$</td>
<td>[BerHirt06]</td>
</tr>
<tr>
<td>active</td>
<td>$t &lt; n/3$</td>
<td>perfect</td>
<td>$O(n \log n)$</td>
<td>[BerHirt08]</td>
</tr>
</tbody>
</table>

**Our new result:**

$$O(n \log n + k)$$  

*(actually: $O(n \log n + k/n^c)$ for any $c$ - can probably be removed)*

---

1) *Amortized* complexity: assumes large enough circuits

2) Requires not too large multiplicative depth
Tricks

Protocol makes use of known techniques:

- Shamir secret sharing [Sha79]
- Beaver’s circuit randomization [Bea89]
- dispute control [BerHirt06]
- linear-time passively-secure multiplication [DamNie07]
- ...

and cumbersome fine-tuning, but crucially relies on two new tricks:

1. efficient batch verification for multiplication triples

   (to verify $c = a \cdot b$ for many shared triples $(a, b, c)$ in one go)

2. efficient “mini MPC” for computing authentication tags

---

3) Independent work: similar trick used in [CraDamPas12], in setting of computational interactive proofs
Reconstruction in the Presence of Faults

secret:

$$s$$

shares:

$$s_1 = f(x_1) \quad \ldots \quad s_i = f(x_i) \quad \ldots \quad s_k = f(x_k) \quad \ldots \quad s_n = f(x_n)$$

$$f(X) = s + a_1 X + \ldots + a_t X^t$$

Problem: how to reconstruct $$s$$ if up to $$t$$ shares are faulty?

In case $$n/3 \leq t < n/2$$: impossible (without additional redundancy)

Idea [RB89]: authenticate the shares
Reconstruction in the Presence of Faults

secret:

\[ f(X) = s + a_1 X + \ldots + a_t X^t \]

shares:

\[ s_1 = f(x_1) \ldots s_i = f(x_i) \ldots s_k = f(x_k) \ldots s_n = f(x_n) \]

\[ \tau_{ik} = \alpha_{ki} \cdot s_i + \beta_{ki} \]

Problem #1: Blows up complexity!

Problem #2: Who computes the tag \( \tau_{ik} = \alpha_{ki} s_i + \beta_{ki} \)?
Solving Problem #1

Authenticate large blocks of shares \( s_i^1, \ldots, s_i^L \) (for secrets \( s^1, \ldots, s^L \)) via

\[
\tau = \alpha \cdot s_i + \beta = \sum_\ell \alpha^\ell s_i^\ell + \beta
\]

with key \( \alpha = (\alpha^1, \ldots, \alpha^L) \) and \( \beta \) (actually: \( \tau_{ki}, \alpha_{ki} \) and \( \beta_{ki} \)).

For large \( L \), efficiency loss due to \( \beta \) and \( \tau \) becomes negligible.

Use the same \( \alpha = (\alpha^1, \ldots, \alpha^L) \) for different blocks \( s_i = (s_i^1, \ldots, s_i^L) \).

For many blocks, efficiency loss due to \( \alpha \) becomes negligible.
Solving Problem #2

Problem #2: Who computes tag $\tau = \alpha s_i + \beta$ (actually $\sum_{l} \alpha^l s_i^l + \beta$)?

Recall:
- $P_k$ – who holds $(\alpha, \beta)$ – is not supposed to learn $s_i$
- $P_i$ – who holds $s_i$ – is not supposed to learn $(\alpha, \beta)$
- dealer is not supposed to learn $(\alpha, \beta)$ – as he might be dishonest

Standard approach/solution:
- **do a 2-level sharing**: every $s_i$ is re-shares into $s_{i1}, ..., s_{in}$
- sub-shares $s_{ij}$ are authenticated
- player $P_i$ computes tags for sub-shares $s_{i1}, ..., s_{in}$ of $s_i$
Solving Problem #2

Problem #2: Who computes tag $\tau = \alpha s_i + \beta$ (actually $\sum \alpha^l s_i^l + \beta$)?

Recall:

- $P_k$ – who holds $(\alpha, \beta)$ – is not supposed to learn $s_i$
- $P_i$ – who holds $s_i$ – is not supposed to learn $(\alpha, \beta)$
- dealer is not supposed to learn $(\alpha, \beta)$ – as he might be dishonest

New approach: by means of a MPC

Appears hopeless:

just sharing the input, $s_i$, leads to quadratic complexity

Good news:

- Circuit is very simple: multiplicative depth 1
- Don’t need to worry about other inputs, $\alpha$ and $\beta$
- Dispute control framework => only need passive security
  (correctness can be verified by cut-and-choose)
Solving Problem #2

Solution: To not share the share $s_i$

Instead: use the remaining shares $(s_j)_{j \neq i}$ of $s$ as shares of $s_i$

Fact:
- any $t$ of the shares $(s_j)_{j \neq i}$ give no info on $s_i$
- any $t+1$ of the shares $(s_j)_{j \neq i}$ reveal $s_i$

Thus: $(s_j)_{j \neq i}$ is a sharing of $s_i$, wrt. to a variant of Shamir’s scheme

(where secret is evaluation of $f$ at point $i$, rather than at $0$)
Protocol MINIMPC

- Given: shares $s_1, \ldots, s_i, \ldots, s_n$

- $P_k$ shares $\alpha$ as follows ($P_i$ gets no share)

- $P_k$ shares $\beta$ as follows ($P_i$ gets no share)

- every $P_j$ ($j \neq i$) sends $\tau_j = \alpha_j s_j + \beta_j$ to $P_i$

- $P_i$ reconstructs $\tau = \alpha s_i + \beta$ from $\tau_j$'s
Multiparty-Computing the Tag

Protocol MINIMPC

- Given: shares $s_1, \ldots, s_i, \ldots, s_n$

- $P_k$ shares $\alpha$ as follows ($P_i$ gets no share)

- $P_k$ shares $\beta$ as follows ($P_i$ gets no share)

**Note:**
Adversary can learn $\alpha$ by corrupting $t$ players $P_j \neq P_i$.
But $\alpha$ is of no use, if he does not corrupt $P_i$.

$P_i$ reconstructs $\tau = \alpha s_i + \beta$ from $\tau_j$'s

$\deg(f) = t$
$f(0) = s$

$\deg(g) = t$
$g(i) = \alpha$
$g(0) = 0$

$\deg(h) = 2t$
$h(i) =$
$h(0) = 0$
Conclusion

∃ unconditionally-secure MPC with near-linear complexity

There exist cases where MPC improves efficiency

Open problems:

- Improve circuit-independent part of the complexity: $O(n^7 k)$
- Remove restriction on multiplicative depth of circuit (also present in the simpler $t < n/3$ setting)
- What about non-threshold adversary structures? (Mini MPC crucially relies on Shamir’s secret sharing scheme)