Efficient Secure Computation with an Honest Majority

Yuval Ishai
Technion
MPC with an Honest Majority

Several potential advantages
- Unconditional security
- Guaranteed output and fairness
- Universally composable security
- This talk: efficiency

Main feasibility results
- Perfect security with $t < n/3$ [BGW88, CCD88]
- Statistical security with $t < n/2$ (assuming broadcast) [RB89]

Goal: minimize complexity
- Communication
- Computation
What can we hope for?

Communication
- Match insecure communication complexity?
  - Possible (in theory, up to poly(k) overhead) using FHE
  - Big open question in information-theoretic setting
- A more realistic goal
  - Allow communication for each gate
  - Minimize amortized cost as a function of n
    - Ignore additive terms that do not depend on circuit size
    - Ideally, $O(1)$ bits per gate

Computation
- $O(1)$ computation per gate?
What can we get?

- Essentially what we could hope for
  - At most polylog(n) overhead
  - Work per party decreases with number of parties!
  - Small price in resilience
  - O(depth) rounds
    - or O(1) rounds with poly(k) overhead and comp. security

- This talk: several simplifying assumptions
  - Inputs originate from a constant number of “clients”
  - Security with abort
  - Statistical security against static malicious adversary
  - Small fractional resilience
  - Broadcast

- Assumptions can be removed
The model

- \( m \geq 2 \) clients, \( n \) servers
  - Only clients have inputs and outputs
  - Assume \( m = O(1) \) in most of this talk
  - Motivated by next talk

![Diagram of model with cloud representing servers and multiple clients.]
The model

- Synchronous secure point-to-point channels + broadcast
  - Servers only talk to clients

- Malicious adversary corrupting:
  - at most $cn$ servers for some constant $0 < c < 1/2$
  - any subset of the $m$ clients

- Statistical security with abort
Functionality represented by a circuit $C$
- Arithmetic circuit over $F$ (with + and $\times$ gates)
- Assume $n \ll |C|$, $\text{depth}(C) \ll |C|$
- Ignore low-order additive terms

**Goal 1: Minimize communication**
- Initial protocols $[BGW88, CCD88]$: $|C| \cdot \text{poly}(n)$
- Best unconditional protocols (this talk): $|C| \cdot O(1)$
- Using FHE: $|\text{input}| + \text{poly}(k) \cdot |\text{output}|$

**Goal 2: Minimize computation**
- Best one can hope for: $|C| \cdot \text{field ops.}$
- Best known (this talk): $|C| \cdot O(\log n)$
  - Assumes large $F$ ($|F| > 2^k$)
  - Polylog($n$) overhead possible for any $F$
Some historical credits

- **Franklin–Yung 92**
  - Run several parallel instances of BGW roughly for price of one
  - Small penalty in security threshold
  - Reduces complexity of BGW for some tasks

- **Hirt–Maurer 01, Cramer–Damgård–Nielsen 01, Damgård–Nielsen 06**
  - Improved overhead of MPC with optimal resilience

- **Damgård–I 06, I–Prabhakaran–Sahai 09**
  - Extend scope of Franklin–Yung technique to general tasks
  - Optimize computational complexity using technique from Groth 09
Some historical credits

- **Damgård–I–Kroigaard–Nielsen–Smith 08**
  efficiency with many clients, boosting resilience using technique of Bracha 87

- **Beerliova–Hirt 08**, **Damgård–I–Kroigaard 10**
  perfect security

- **Beaver–Micali–Rogaway 90**, **Damgård–I 05**
  constant-round protocols

- **Chen–Cramer 06**
  using constant-size fields
Starting point: BGW

- Secret-share inputs
- Evaluate C on shares
  - Non-interactive addition
  - Interactive multiplication
- Recover outputs

- Secure with \( t < n/2 \) (semi-honest) or \( t < n/3 \) (malicious)
- Complexity: \(|C| \cdot O(n^2)\) (semi-honest)
  \(|C| \cdot \text{poly}(n)\) (malicious)
Sources of overhead

- Each wire value is split into n shares
  - Use “packed secret sharing” to amortize cost

- Multiplication involves communication between each pair of servers
  - Reveal blinded product to a single client

- Expensive consistency checks
  - Efficient batch verification
Share packing

- Handle block of \( w \) secrets for price of one.
- Security threshold degrades from \( d \) to \( d-w+1 \)
- \( w=n/10 \) \( \Rightarrow \) \( \Omega(n) \) savings for small security loss
- Compare with error correcting codes

Denote shared block by \([x_1, \ldots, x_w]_d\)
YES: evaluate a circuit on multiple inputs in parallel

NO: evaluate a circuit on a single input

3 inputs

5 blocks
Warmup: Semi–honest, depth 1

Client A

Client B

Client C

- Extends to constant-depth circuits
- Still 2 rounds, \( t = \Omega(n) \)

A→S: \( p_A = [a_1, a_2, a_2]_d \)
\( q_A = [a_1, a_1, a_2]_d \)
\( z_A = [0, 0, 0]_2d \)

B→S: \( p_B = [b_1, b_2, b_1]_d \)
\( q_B = [b_2, b_1, b_2]_d \)
\( z_B = [0, 0, 0]_2d \)

S→C: \( p_A p_B + z_A + z_B \)
\( q_A + q_B \)
Semi–honest, any depth

- Assume circuit is composed of layers 1,...,H.
- Clients share inputs into $[\text{left}^1]_d$ and $[\text{right}^1]_d$
- For $h=1$ to $H-1$:
  - Clients generate random blocks $[r]_{2d}$, $[\text{left}_r]_d$ and $[\text{right}_r]_d$ replicated according to structure of layer $h+1$
  - Servers send masked output shares of layer $h$ to Client A:
    $$[y]_{2d} = [\text{left}^h]_d*[\text{right}^h]_d + [r]_{2d} \ (\ast \in \{x,+,-\})$$
  - A decodes, rearranges and reshares $y$ into $[\text{left}_y]_d$, $[\text{right}_y]_d$
  - Servers let
    - $[\text{left}^{h+1}]_d = [\text{left}_y]_d - [\text{left}_r]_d$
    - $[\text{right}^{h+1}]_d = [\text{right}_y]_d - [\text{right}_r]_d$

- Servers reveal output shares
  $$[\text{left}^H]_d*[\text{right}^H]_d + [0]_{2d}$$
Example

Secure Computation and Efficiency
Bar-Ilan University, Israel 2011
Malicious model

- Need to protect against $t = \Omega(n)$ malicious servers and $t’ < m$ malicious clients.
- Malicious servers handled via error correction
  - Valid shares form a good error-correcting code
  - Error detection sufficient for security with abort
- Malicious clients handled via efficient VSS procedures (coming up)
Efficient statistical VSS

- Recall: only shoot for security with abort
- Two types of verification procedures
  - Verify that shares lie in a linear space
    - E.g., degree-d polynomials
  - Verify that shared blocks satisfy a given replication pattern
    - E.g., \([r_1, r_1, r_2, r_1] [r_2, r_3, r_1, r_2]\)
- Cost is amortized over multiple instances
Verifying membership in a linear space

- Suppose Client A distributed a vector $v$ between servers.
  - $S_i$ holds the $i$-th entry of $v$
  - Can be generalized to an arbitrary partition of entries
- **Goal**: Prove in zero-knowledge to Client B that $v$ is in some (publicly known) linear space $L$.
- **Protocol**:
  - A distributes a random $u \in rL$
  - B picks and broadcasts $c \in rF$
  - Servers jointly send $w = cv + u$ to B
  - B checks that $w \in L$
- **ZK**: $w$ is a random vector in $L$
- **Soundness** (static corruption):
  - consider messages from honest servers
  - $cv + u, c'v + u \in L \Rightarrow (c - c')v \in L \Rightarrow v \in L$
  - soundness error $\leq 1/|F|$
Amortizing cost

- Can be jointly generated by clients
- Can be pseudorandom (\(\varepsilon\)-biased)

\[
\begin{array}{cccc}
\text{c}_1 & \times & v_1 \\
\text{c}_2 & \times & v_2 \\
\text{c}_3 & \times & v_3 \\
\text{c}_4 & \times & v_4 \\
\text{c}_5 & \times & v_5 \\
+ & & u \\
\end{array}
\]

\[w \in L?\]
Verifying replication pattern

secret

inner product

public

\[
\begin{array}{cccc}
\text{a b c d} & \text{e f g h} \\
\text{r_1 r_2 s_1 s_2} & \text{r_3 s_3 r_4 r_5} \\
\text{r_2 r_3 s_2 s_3} & \text{r_4 s_1 r_5 r_1}
\end{array}
\]
Asymptotic efficiency

Communication
- $O(|C|)$ field elements ($|F| > n$) + “low order terms”
- Low order terms include:
  - Additive term of $O(\text{depth} \cdot n)$ for layered circuits
  - depth $\Rightarrow$ # “communicating layer pairs” for general circuits
  - Multiply by $k/\log|F|$ for small fields
    ($k = \text{statistical security parameter}$)

Computation
- Communication $\times O(\log n)$
  - Uses FFT for polynomial operations
  - Multiply by $k/\log|F|$ for small fields
Goal: small fractional resilience $\Rightarrow$ nearly optimal resilience
- without increasing asymptotic complexity!

Solution: server virtualization
- Example: $0.01n$-secure $\Pi \Rightarrow 0.33n$-secure $\Pi'$
- Pick $n$ committees of servers such that
  - Each committee is of size $s=O(1)$
  - If $0.33n$ servers are corrupted, then $>99\%$ of the committees have $<s/3$ corrupted members
- Choose committees at random, or use explicit constructions

$\Pi'$ uses $s$–party BGW to simulate each server in $\Pi$ by a committee
- Overhead $\text{poly}(s)=O(1)$
Using constant-size fields

- Consider a boolean circuit $C$ with $|C| \gg$ depth
- Previous protocol requires $|F| > n$
  - $O(|C| \log n)$ bits of communication
- Can we get rid of the $\log n$ term?
- Yes, using algebraic-geometric codes
  - Field size independent of $n$
  - Small fractional loss of resilience
  - Asymptotically optimal protocols for natural classes of circuits
Other extensions

- Many clients
  - Previous protocol required generating secret blocks
  - Easy to implement by summing blocks generated by all clients
  - Overhead can be amortized if only a constant fraction of clients are corrupted
    - Requires converting circuit into a “repetitive” form
  - Gives protocols with polylog(n) overhead in standard n-party setting with \( t = \Omega(n) \).

- Perfect security
  - Use efficient variant of BGW VSS with share packing
**Constant-round protocols**

- **BMR90: Constant-round version of BGW**
  - Uses garbled circuit technique
  - Black-box use of PRG in semi-honest model (Benny’s talk)
  - Non-black-box use of PRG in malicious model
    - Required for zero-knowledge proofs involving “cryptographic relations”
    - In BMR paper: distributed ZK proofs of consistency of seed with PRG output

- **DI05: Black-box use of PRG in malicious model**
  - Uses threshold symmetric encryption
Conclusions

- An honest majority can be useful
  - Unconditional, composable security
  - Fairness
  - Efficiency

- Open efficiency questions
  - Break circuit size communication barrier for unconditional security
  - Constant computational overhead
  - Improve additive terms
  - Better constant-round protocols
    - $O(1)$ PRG invocations per gate?
  - Practical efficiency