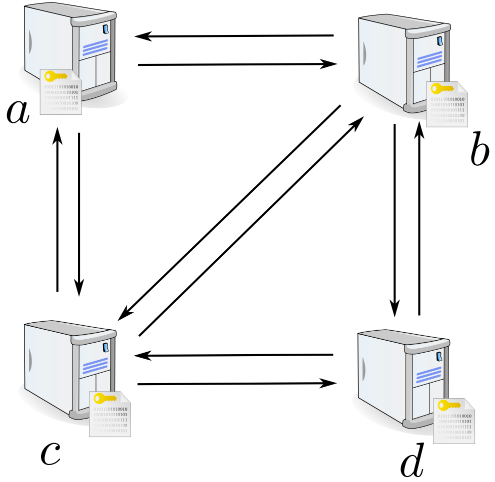
Practical Covertly Secure MPC for Dishonest Majority – or: Breaking the SPDZ Limits

Ivan Damgård, Marcel Keller, Enrique Larraia, Valerio Pastro, *Peter Scholl*, Nigel Smart

University of Bristol, UK
University of Aarhus, Denmark

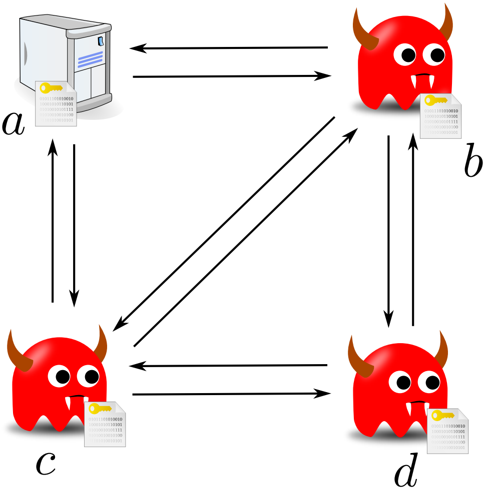
Secure Multi-Party Computation



Goal: compute $f(a, b, c, d)$

Figure: by Pascal Wagler

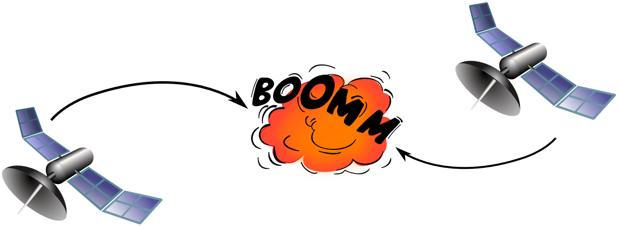
Dishonest majority



Goal: compute $f(a, b, c, d)$

Why? (1)

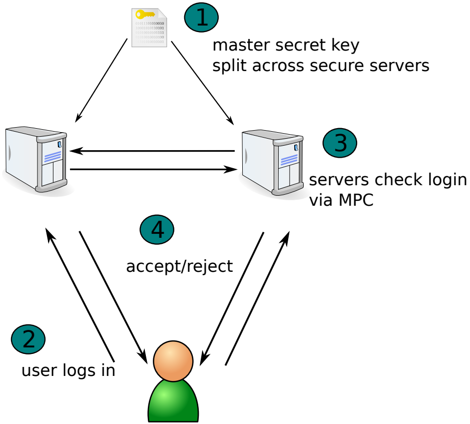
- MPC can replace any scenario where a Trusted Third Party would normally be used to compute on sensitive data.
- There should be mutual benefit in one or more parties learning result.
- Example: satellites – detect collisions without revealing location.



Why? (2)

- MPC can be used to enhance security of stored data.
- Sensitive data is split across multiple servers.
- When data needs to be used, perform computation with MPC.
- Secret data and result of the computation are never known by any one server.
  - Attacker must compromise every server to gain entry

Example: one-time password verification (e.g. RSA tokens)



Overview

- Build on SPDZ ('SPeedZ') protocol (Damgaard *et al.* Crypto '12)
- Practical, actively secure against $n - 1$ corrupted parties (UC secure)
- Various improvements to protocol, aimed at practical scenarios (integer + floating point arithmetic, reactive computation)
- Implementation

Previous work

Active secure, dishonest majority MPC:

- Early construction [CLOS02]
- "MPC in the Head" [IKOS07, IPS08]
- [BDOZ11], [SPDZ12], [DKLMS12]

Boolean circuit approach (2-parties, active):

- Garbled circuits [KSS12, Lin13, HKE13, ...]
- Tiny-OT [NNOB12]

SPDZ-1 implementation (SCN '12 [DKLMS]):

- Focus on AES, $\mathbb{F}_{2^8}$
- Covert security was ad-hoc, no security proofs
- Also benefits from our improvements

SPDZ protocol

Preprocessing ('offline') stage

- Parties interact to generate 'raw data'
- Computation is independent of function inputs
- Uses public key crypto (FHE)

Online stage

- Parties interact to perform the computation (on secret shared data)
- Doesn't need PK crypto, much more efficient
- Information-theoretically secure

Data representation

All data $\in \mathbb{F}_p$, prime $p \approx 2^{64}$ or 2^{128}

$\alpha := \alpha_1 + \cdots + \alpha_n$ is the long-term MAC key.

x is shared across n parties such that:

$$x = x_1 + x_2 + \cdots + x_n$$
$$\alpha \cdot x = \gamma(x)_1 + \gamma(x)_2 + \cdots + \gamma(x)_n \qquad \text{(MAC on } x\text{)}$$

$\boxed{P_1}$ x_1 $\gamma(x)_1$ $\boxed{P_2}$ x_2 $\gamma(x)_2$

$\boxed{P_3}$ x_3 $\gamma(x)_3$ $\boxed{P_4}$ x_4 $\gamma(x)_4$

Write $\langle x \rangle := ((x_1, \ldots, x_n), \gamma(x)_1, \ldots, \gamma(x)_n)$

Online phase

Given $\langle x \rangle, \langle y \rangle$:

Addition

P_i computes $\langle z \rangle$ by:

- $z_i = x_i + y_i$
- $\gamma(z)_i = \gamma(x)_i + \gamma(y)_i$

$\Rightarrow z = x + y$ and $\alpha \cdot z = \alpha \cdot (x + y)$
Addition, and any linear function, is a local operation.

Online phase

Open

$x = \text{open}(\langle x \rangle)$ (1 round of communication)
Note: only the data share is revealed here, not the MAC.

Multiplication

Using a pre-computed multiplication triple $\langle a \rangle, \langle b \rangle, \langle c \rangle$ such that $a \cdot b = c$, compute:

$$
\begin{aligned}
d &= \text{open}(\langle x \rangle - \langle a \rangle \\
e &= \text{open}(\langle y \rangle - \langle b \rangle) \\
\langle x \cdot y \rangle &= d \cdot e + e \cdot \langle a \rangle + d \cdot \langle b \rangle + \langle c \rangle
\end{aligned}
$$

N.B. any computation can be expressed with just add/multiply in $\mathbb{F}_p$

MAC checking

SPDZ 1: reveal α to check MACs.
$\Rightarrow$ once a MAC is checked, cannot continue computation!
Given:

- a: opened value
- MAC γ and MAC key α: secret shared

Want to check:

$$\gamma = \alpha \cdot a$$

- Locally compute shares of $\gamma - \alpha \cdot a$
- Reveal and check $= 0$
- Batching: check random linear comb. of many MACs

Allows reactive computation

SPDZ: preprocessing

- Use FHE scheme to generate multiplication triples
- Parties have common public key, and shares of secret key.

Want $\langle a \rangle, \langle b \rangle, \langle c \rangle$ such that $a \cdot b = c$.

- P_i generates random a_i, b_i
- Broadcasts $\mathsf{Enc}(a_i)$, $\mathsf{Enc}(b_i)$
- Compute ciphertexts
  - $\mathbf{a} = \sum_i \mathsf{Enc}(a_i)$
  - $\mathbf{b} = \sum_i \mathsf{Enc}(b_i)$

  using homomorphic addition

SPDZ: preprocessing

- Compute $\mathbf{c} = \mathrm{Enc}(\mathbf{a}) \cdot \mathrm{Enc}(\mathbf{b})$ via homomorphic multiplication
- Distributed decryption to get $\langle c \rangle = \mathrm{DistDec}(\mathbf{c})$

Only needs one multiplication, so FHE is efficient.

SPDZ 2 offline: improvements

- Distributed key generation protocol:
  - SPDZ 1 assumed 'magic' setup for FHE keys
  - Generic MPC techniques for this are expensive
  - Shares of $\mathsf{sk}, \mathsf{sk}^2$ for key/modulus switching (BGV scheme)
  - Ciphertexts 50% smaller
- Preprocessing data:
  - Extended to generate shared bits, squaring tuples, random values

Security

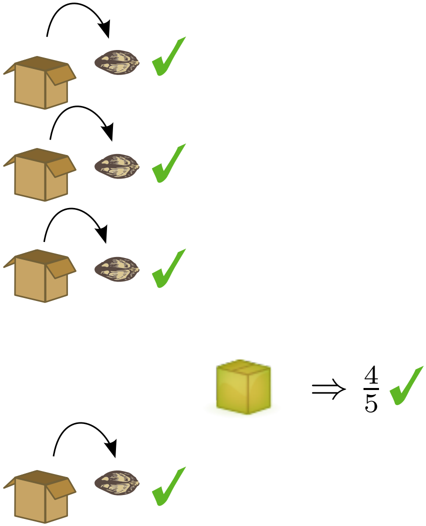
- Offline phase as outlined: only passively secure.
- (Online phase: active security)
- Want to prevent adversaries from tampering with the protocol.
- Ensure a cheating player is detected with probability $1 - 1/c$

Covert: c small, e.g. 5, 20

Active: c tiny, e.g. $2^{40}, 2^{80}$

- Previous approaches:
  - Active with zero knowledge
  - Covert with ZK, only $c = 2$
- This work:
  - Faster covert and active variants using cut and choose

Covert security: cut and choose



- Run c instances of protocol
- Commit to random seed for each instance
- Open all bar one of the seeds
  - Check random data for correctness
- Use data from final, unopened seed

$\Rightarrow$ adversary can cheat with probability $1/c$

Active security

New approach to active security:

- Variant of cut-and-choose
- Can be run in small batches (unlike ZK)
- Gives smaller FHE parameters

Implementation

Security	n	KeyGen (ms)	Offline (ms/triple)	Online (ms/mult)
Covert (5)	2	5900	1.94	–
	3	7700	2.67	–
Active (2^{40})	2	–	19.5	0.002
	3	–	28.7	0.0035

Table: Runtimes for key generation and secure 64-bit multiplication

MPC vs Computing

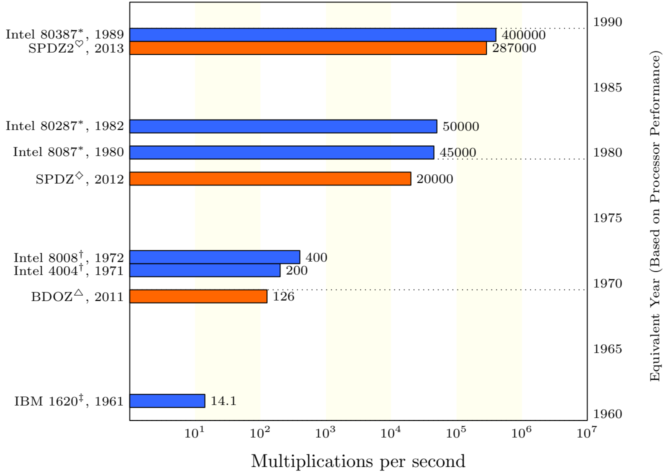


Figure: Blue bars: historical CPUs (vertically spread by year)
Red bars: MPC protocols (3 players, LAN)

Summary: SPDZ 2

- Fully reactive computation
- Complete protocol inc. key generation
- Improved covert/active secure protocols with cut-and-choose
- Implementation

http://eprint.iacr.org/2012/642

MPC is practical and moving fast.