Threshold Cryptography in MP-SPDZ

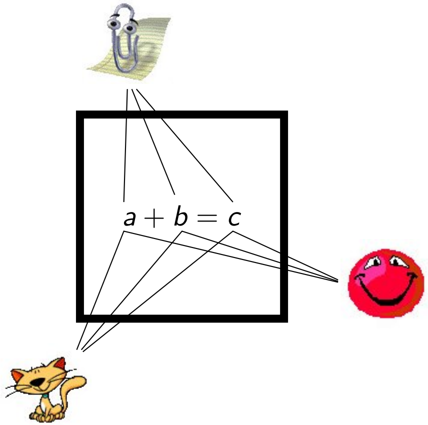
MPTS 2023: NIST Workshop on Multi-Party Threshold Schemes 2023

Marcel Keller

CSIRO's Data61

26 September 2023

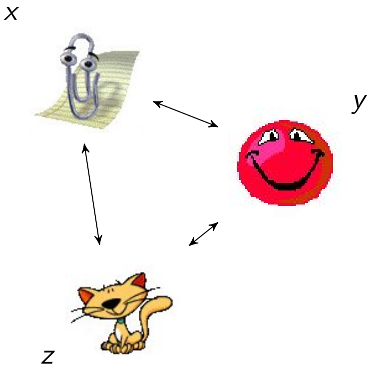
Imagine a Magic Black Box Between a Set of Parties



Parties

- Have handles to values
- Don't know the values
- Can input values
- Can agree on computations creating new values
- Can agree on outputting values

Secure Multiparty Computation: Black Box as Protocol



Wanted: $f(x, y, z)$

- Computation on secret inputs
- Replace black box
- Central questions in MPC
  - How many honest parties?
  - Dishonest parties still follow the protocol?
- MP-SPDZ supports $>$ 40 protocol variants across all properties

Unifying MPC: Basic Operations

	Addition	Multiplication
Communication	✗	✓
Shamir/Replicated	Add shares	Reshare
SPDZ/TinyOT	Add shares/MACs	Beaver

Unified C++ Interface

```
for (int i = 0; i < n; i++)
  sum[i] = a[i] + b[i];

protocol.init_mul();
for (int i = 0; i < n; i++)
  protocol.prepare_mul(a[i], b[i]);
protocol.exchange();
for (int i = 0; i < n; i++)
  product[i] = protocol.finalize_mul();
```

- Addition is straightforward
- Similar for multiplication would lead to sequential execution
- Prepare/exchange/finalize minimal interface for parallel execution

C++ Templating

```
Rep3<Rep3Share<Z2<64>>> proto;
Rep3<Rep3Share<gfp_<0, 2>>> proto;
Shamir<ShamirShare<gfp_<0, 2>>> proto;
Shamir<ShamirShare<gf2n>>> proto;
Beaver<SemiShare<Z2<64>>> proto;
Beaver<SemiShare<gfp_<0, 2>>> proto;
Beaver<LowGearShare<gfp_<0, 2>>> proto;
Beaver<HighGearShare<gfp_<0, 2>>> proto;
```

- Share type defines protocol variant
- Share types are templated on domain
- Maximal code reuse across variants

Threshold ECDSA with Black-Box MPC

ECDSA Signature

$$s = k^{-1}(H(M) + \mathsf{sk} \cdot r_x)$$

where

- k secret randomness in $\mathbb{Z}_p$
- r_x a coordinate of kG in group of order p

Black-Box MPC

- Use black box for secret key sk and k
- Need to publish kG but not k
- Secret sharing scheme over $\mathbb{Z}_p$ implies one over the group with local conversion

MP-SPDZ Domain Interface for EC Group

- Uses OpenSSL for EC functionality
- 200 lines of code
- 7 static members, 10 overloaded operators, 4 constructors, (de)serialization

```
P256Element P256Element::operator +(const P256Element& other) const
{
    P256Element res;
    assert(EC_POINT_add(curve, res.point, point, other.point, 0) != 0);
    return res;
}
```

ECDSA in MP-SPDZ (Simplified)

$$s = k^{-1}(H(M) + \mathsf{sk} \cdot r_x)$$

```
Scalar hash = hash_to_scalar(message);
Share<Scalar> k, b, c;
get_random_triple(k, b, c);
Share<Scalar> k_inv = b / open(c);
Scalar r_x = open(Share<P256Element>(k)).x();
Scalar s = open(mul(k_inv, hash + sk * rx));
```

Supported Protocols

Name	Honest Majority	Malicious	
Rep3	Y	N	`https://ia.cr/2016/768`
Mal-Rep3	Y	Y	`https://ia.cr/2017/816`
Shamir	Y	N	`https://ia.cr/2000/037`
Mal-Shamir	Y	Y	`https://ia.cr/2017/816`
Semi	N	N	`https://ia.cr/2016/505`
MASCOT	N	Y	`https://ia.cr/2016/505`
ATLAS	N	N	`https://ia.cr/2021/833`
Rep4	N	Y	`https://ia.cr/2020/1330`
SPDZ-wise Rep3	N	Y	`https://ia.cr/2018/570`

Links

https://github.com/data61/MP-SPDZ
https://ia.cr/2020/521
https://ia.cr/2019/889
https://mp-spdz.readthedocs.io/en/latest/ecdsa.html
https://twitter.com/mkskeller