Points for presentation:

What is SMPC?

Scale-Mamba

* Online/Offline (calculate TRIPLES)
* Abort/How to fight back adversaries?
* Secret Sharing? MACs?
* ScaleMamba as suit for SMPC that allows to choose all the state of the art technologies

Usecases:

<https://www.researchgate.net/publication/328040632_A_New_Approach_to_Privacy-Preserving_Clinical_Decision_Support_Systems_for_HIV_Treatment>

<https://arxiv.org/ftp/arxiv/papers/1810/1810.01107.pdf>

Useage?:

<http://www.bu.edu/hic/files/2018/06/2018-06-05-Anand.Swarte.pdf>

Q2 access structures: Scale Mamba can use all?

Q2 / full threshold

Monotone Span Programs (MSP)

Qualified/unqualified set

Universal Composability?

Beaver triple?

Replicated secret sharing?

N.P.: Practical

covertly secure MPC for dishonest majority - or: Breaking the SPDZ limits. I

MASCOT

2018 - 467

Computationally secure vs malicious adversaries with ABORT

MPC: two categories: garbled circus & linear secret sharing schema based (LSSS-based)

By contrast, the LSSS-based approach involves using a so-called linear

secret-sharing scheme, in which the parties: \share" a secret into several shares

which are distributed to different parties, perform computations on the shares,

and then reconstruct the secret at the end by combining the shares to determine

the output

although many modern practical protocols that

realise LSSS-based MPC often make use of computationally-secure primitives

such as somewhat-homomorphic encryption (SHE) [25] or oblivious transfer

(OT)

seen significant, efficient instantiations

for full-threshold access structures [7, 23, 25, 32], is where the protocol is

secure if at least one party is honest,

Most LSSS-based MPC protocols split the computation into two parts: an of-

ine phase, in which parties interact using \expensive" public-key cryptography

to lay the groundwork for an online phase in which only \cheap" informationtheoretic

primitives are required. The online phase is where the actual circuit

evaluation takes place. For the access structures considered in this work, namely

Q2 structures, the offline phase is almost as fast as the online phase. Thus the

goal here is to minimize the cost of communication in both phases.

In the full-threshold SPDZ protocol [25] and its successors, e.g. [23, 32], authentication

is achieved with additively homomorphic message authentication

codes (MACs). For each secret that is shared amongst the parties, the parties

also share a MAC on that secret. Since the authentication is additively homomorphic

and the sharing scheme is linear, this means that the sum (and consequently

scalar multiple) of authenticated shares is authenticated \for free" by performing

the addition (or scalar multiplication) on the associated MACs. More work

is required for multiplication of secrets, but the general methodology for doing

these operations on shared secrets is now generally considered \standard" for

MPC in this setting

One important branch of this authentication methodology contributing significantly to their practical performance is the amortization of verification costs

by batch-checking MACs, a technique developed in [6,25], amongst other works.

Linear Secret Sharing Schemes: An LSSS is a method of sharing secret data

amongst parties. It consists of three multi-party algorithms: Input, Open, and

ALF (affine linear function), allowing parties to provide secret inputs, reveal (or

6

open) secrets, and compute an affine linear function on shared secrets. In a practical

sense, this means that the parties can add secrets, multiply by scalars, and

add public constants to a shared secret, all by local computations. In this work

we consider, as examples, the three most well-known secret-sharing schemes:

Shamir; replicated, also known as CNF-based (conjunctive-normal-form-based);

and DNF-based (disjunctive-normal-form-based).

Pre-processing: Many modern MPC protocols split computation into two

phases, the offline or pre-processing phase and the online phase. In the offline

phase, the parties engage in several rounds of communication to produce data

which can then be used in the online phase. The purpose of doing this is that

the pre-processing can be done at any time prior to the execution of the online

phase, can be **made independent of the function to be computed**, and may

use expensive public-key primitives, in order to allow the online phase to use

only fast information-theoretic primitives. In our protocol design, we follow this

model, although we only require symmetric-key primitives throughout since the

access structure is Q2.

This process ensures that the shares held by all parties are consistent,

even though each party need only send one share to one party per opening. If

many shares are opened in the execution of the protocol (as is the case in SPDZlike

protocols, **since every multiplication requires two secrets to be opened**),

this significantly reduces communication overhead, at the cost of cryptographic

assumptions for the existence of a collision-resistant hash function. This was

generalised to any replicated scheme Q2 LSSS by Keller et al. [33].

To achieve this, in our protocol each party will receive enough shares from

other parties to determine \optimistically" all shares held by all parties { that

is, reconstruct the entire share vector { and then all parties will compare their

reconstructed share vectors. To amortise the cost of comparison, the parties will

actually update a local collision-resistant hash function each time they reconstruct

a new share vector and will then compare the final output of the hash

function at the end of the computation, when output is required. This, in essence,

is the idea behind the protocols of Furakawa et al. [27] and Keller et al. [33] that

are tailored to replicated secret-sharing.

In this section we give an offline subprotocol. Recall that the offline phase of

MPC protocols involves the generation of so-called Beaver triples: triples of

share vectors**, (a; b; c) such that c encodes the product of the secrets encoded**

**by a and b**. As is relatively standard practice for MPC protocols in the preprocessing

model, we provide a semi-honest multiplication procedure, which is

then bootstrapped to active security using the standard technique known as

**sacrificing to catch if errors were introduced**.

The Beaver triples must be share-vectors

Method 2 Use [33] to do all of the pre-processing, and then use the technique by Cramer et al. [18] to convert the replicated shares into shares under any

LSSS computing the same access structure by local computations (after an

inexpensive one-time set-up phase).

{ Method 3 Generalise the procedure from [33] to convert additive sharings

to replicated.

We would expect the second method to be extremely efficient in many situations,

since after setting up keys for generating random secrets efficiently, every

party only needs to send \_(n) \_field elements to different parties for each passive

multiplication. The main problem is that the required number of PRF (pseudorandom

function) keys grows linearly with the number of maximally unqualified

sets. Indeed, for a (84; 41)-threshold access structure using the former method,

the parties would need to agree on􀀀8443\_> 280 keys.We note that the optimisation

in [33], requiring PRF keys to generate some shares of a product deterministically,

can be instantiated using keys already set up for generating the PRSS

shares. The computational cost is asymptotically O(2n=pn) PRF evaluations

per party; nevertheless this computational cost reduces the communication cost,

so there is a trade-off here.

Table 1