SPD$\mathbb{Z}_{2^k}$: Efficient MPC mod 2^k for Dishonest Majority

To appear at CRYPTO 2018

Ronald Cramer[1] Ivan Damgård[2] Daniel Escudero[2] Peter Scholl[2] Chaoping Xing[3]

May 29, 2018

[1]CWI, Amsterdam

[2]Aarhus University, Denmark

[3]Nanyang Technological University, Singapore

Introduction

MPC

- Given a function $f : D^n \to D$ (where D is some set) and n parties $P_1, \ldots, P_n$, where party P_i holds some input $x_i \in D$,

Secure Multiparty Computation
Obtain a protocol Π such that, at the end of its execution, the parties learn $z = f(x_1, \ldots, x_n)$ **and nothing else**.

- Comparable to an ideal world where inputs are sent to a third trusted party who computes the output and does not reveal anything else

Many different approaches

- Garbled Circuits ($D = \mathbb{F}_2$)
- BMR ($D = \mathbb{F}_2$)
- GMW ($D = \mathbb{F}_2$)
- BGW ($D = \mathbb{F}_p$ for $p > n$)
- BeDOZa ($D = \mathbb{F}_p$)
- SPDZ ($D = \mathbb{F}_p$)
- MASCOT ($D = \mathbb{F}_p$)

Few works address the case $D = \mathbb{Z}_{2^k}$.[1]

[1] $\mathbb{Z}_M$ denotes the ring of integers modulo M

Why should we care about this case?

In many scenarios, it is desirable to let D be the ring of integers modulo 2^k.

- Computation modulo 2^k matches closely what happens on standard CPUs and hence protocol designers can take advantage of the tricks found in this domain;
  - It simplifies implementations by avoiding the need for modular arithmetic,
  - It reduces the complexity of compiling existing programs into arithmetic circuits.
- Functions containing comparisons and bitwise operations are typically easier to implement using arithmetic modulo 2^k;
- Operations modulo 2^k are expensive to emulate with finite field arithmetic.

Some works on this direction

- (Cramer et al, EUROCRYPT 2003) showed how to contruct actively secure MPC over black-box rings.
    - Mostly a feasibility result
    - Concrete efficiency is not clear
- (Bogdanov et al, ESORICS 2008, aka Sharemind) allows for computation over this ring.
    - Assumes $n = 3$, $t = 1$;
    - Provides only passive security.
- (Damgård, Orlandi, Simkin, CRYPTO 2018) show a compiler from passive to active security for arbitrary rings.
    - Small number of corrupt players.

Why is it so difficult?

Most practical secret-sharing-based multiparty protocols like SPDZ and MASCOT require authentication mechanisms to avoid cheating which only work over fields.

- It has been an open problem to design an efficient homomorphic authentication scheme modulo 2^k

Many problems appear when working over $\mathbb{Z}_{2^k}$ in contrast to $\mathbb{F}_p$:

- Zero-divisors!
- Non-invertible elements!
- Taking dot product with random vectors is not a 2-universal function!

Our contributions i

1. A new additively homomorphic authentication scheme that works in $\mathbb{Z}_{2^k}$ and is as efficient as the standard solution over a field.
    - New number-theoretic tricks to overcome the difficulties of working over a ring like $\mathbb{Z}_{2^k}$.
    - A new method for checking large batches of MACs with a communication complexity that does not depend on the size of the batch.
2. As a corollary, we obtain a SPDZ-style online protocol that securely computes an arithmetic circuit over $\mathbb{Z}_{2^k}$ with statistical security (assuming a preprocessing functionality).
    - Total computational work $\leq O(|C|n)$ operations over $\mathbb{Z}_{2^{k+s}}$
    - Amortized communication complexity $\leq O(|C|k)$ bits

Our contributions ii

3. An implementation of the preprocessing functionality to generate multiplication triples.
    - Roughly twice the communication cost of MASCOT

SPDZ

Additive Secret sharing with MACs

We write $[x]$ to denote the following situation[2]

- Each party P_i holds a random value x^i such that $\sum x^i = x$.
- There is a (global) random value α for which each party P_i has a share α^i such that $\sum \alpha^i = \alpha$.
- Each party P_i holds a random value m^i such that $\sum m^i = \alpha \cdot x$.

Important!
$[x+y] = [x] + [y]$, $[c \cdot x] = c \cdot [x]$ and $[x+c] = [x] + c$ can be computed <u>locally</u>.

[2] In this setting D is a finite field of size p (a big prime)

Secure computation in a nutshell

Input phase

$$[x_i] = (x_i - r_i) + [r_i]$$

where x_i are the inputs and $(r_i, [r_i])$ is preprocessed.

Addition gates

$$[x + y] = [x] + [y]$$

Multiplication gates

$$[x \cdot y] = [c] + (x - a) \cdot [b] + (y - b) \cdot [a] + (x - a)(y - b)$$

where $([a], [b], [c])$ is preprocessed with $c = a \cdot b$.

MAC Checking

Consider a shared value $[x]$ ($x = \sum x^i$, $x \cdot \alpha = \sum m^i$)

- To (partially) open it, each party P_i announces its share x^i and the parties reconstruct $x = \sum x^i$
- To check that this value is correct, each party computes, commits to and announces $z^i = m^i - \alpha^i x$.
- Then the parties check that $\sum z^i = 0$.

Security Analysis

Some corrupt parties may lie about their shares and open an incorrect value $x' = x + \delta$ with $\delta \neq 0$.

- It can be shown that in this case the adversary knows Δ and δ such that $\delta \cdot \alpha = \Delta$.
- Since $\alpha = \delta^{-1} \cdot \Delta$ and α is random, this happens only with probability at most $1/p$.

This does not work modulo 2^k: the equation $\Delta \equiv \alpha \cdot \delta \bmod 2^k$ can be satisfied with high probability (e.g. $\delta = 2^{k-1}$ and $\Delta = 0$)

- Main problem: δ may not be invertible modulo 2^k.

SPD$\mathbb{Z}_{2^k}$

Our solution

The circuit to be computed is in $\mathbb{Z}_{2^k}$, but the computation is performed modulo 2^{k+s}.

Same type of sharing $[x]$ than SPDZ.[3]

- Each party P_i holds a random value $x^i \in \mathbb{Z}_{2^{k+s}}$ such that $x' \equiv_k x$ where $x' \equiv_{k+s} \sum x^i$.
- There is a (global) random value α for which each party P_i has a share $\alpha^i \in \mathbb{Z}_{2^s}$ such that $\sum \alpha^i \equiv_{k+s} \alpha$.
- Each party P_i holds a random value $m^i \in \mathbb{Z}_{2^{k+s}}$ such that $\sum m^i \equiv_{k+s} \alpha \cdot x'$.

[3] $x \equiv y \bmod 2^\ell$ will be abbreviated by $x \equiv_\ell y$

Security Analysis[4]

What if the check passes ($\alpha \cdot \delta \equiv \Delta \bmod 2^{k+s}$) and there is an error $\delta \not\equiv 0 \bmod 2^k$.

- Let v be the largest integer such that $2^v | \delta$ (we have that $v < k$), then $\alpha \cdot \frac{\delta}{2^v} \equiv \frac{\Delta}{2^v} \bmod 2^{k+s-v}$
- But $\delta/2^v$ is odd! So we can invert: $\alpha \equiv \left(\frac{\delta}{2^v}\right)^{-1} \cdot \frac{\Delta}{2^v} \bmod 2^{k+s-v}$
- Therefore, the adversary knows the last $k+s-v$ bits of α, which happens with probability at most $2^{v-k-s} < 2^{-s}$.

[4] The actual MAC checking protocol is a bit more complicated due to some random masks that are required for the upper s bits

SPD$\mathbb{Z}_{2^k}$: Full protocol

Offline phase (preprocessing)

1. Random authenticated values
2. Multiplication triples
3. Generate shares of MAC key and shares of MACked values

Online phase

1. Distribute inputs
2. Compute shares of the values on the circuit
3. Check correctness of the opened values using their MACs
    - Checking individual MACs
    - Batch MAC-checking

SPD$\mathbb{Z}_{2^k}$: Full protocol

Offline phase (preprocessing)

1. Random authenticated values
2. Multiplication triples
3. Generate shares of MAC key and shares of MACked values

Online phase

1. Distribute inputs
2. Compute shares of the values on the circuit
3. Check correctness of the opened values using their MACs
    - Checking individual MACs
    - Batch MAC-checking

Batch MAC-checking

Motivation

During the execution of the protocol many values are partially opened (e.g. on the multiplication gates)

- Checking correctness for each one of these individually incurs in a large overhead
- These are only the means towards the final goal: ensuring correctness of the output

Instead, we perform only <u>one</u> check at the end of the execution that takes into account all previously opened values at once.

Typical solution over fields

Take a random linear combination of the partially opened values and check correctness of this combination.

Batch MAC-checking in SPD$\mathbb{Z}_{2^k}$

Let $x_1 + \delta_1, \ldots, x_t + \delta_t \in \mathbb{Z}_{2^{k+s}}$ be the partially opened values where the $x_i \in \mathbb{Z}_{2^k}$ are the "correct" values.

Key idea

Compute, open and check $[x] = \sum_i \chi_i \cdot [x_i]$

- The argument over a field relies on the fact that if $(\chi_1, \ldots, \chi_t) \in \mathbb{F}^t$ is random and $(\delta_1, \ldots, \delta_n) \in \mathbb{F}^t$ is non-zero, then $\delta = \sum_i \chi_i \cdot \delta_i$ is non-zero with low probability
- This does not work modulo 2^k (same invertibility issues as before)
- Using the same solution as before naively would require us to add yet another register (i.e. work in $\mathbb{Z}_{2^{k+2s}}$)

This is not actually required if we do a more fine-grained analysis!

- Let E be the event in which the check of $[x]$ passes, i.e. the equation $\delta \cdot \alpha \equiv_{k+s} \Delta$ is satisfied with $\delta \equiv_{k+s} \sum_i \chi_i \cdot \delta_i$
- Let w be the largest integer such that 2^w divides δ.

Theorem

$$\Pr[E] = \overbrace{\Pr[E | 0 \leq w \leq k]}^{\leq 2^{-s}} \cdot \overbrace{\Pr[0 \leq w \leq k]}^{\leq 1} + \sum_{c=1}^{s} \underbrace{\Pr[E | w = k + c]}_{\leq 2^{c-s}} \cdot \underbrace{\Pr[w = k + c]}_{\leq 2^{-c-1}} \leq 2^{-s} + 2^{-s-1+\log s}$$

Multiplication Triples

General Idea (high level) i

The parties need to preprocess triples $([a], [b], [c])$ such that a, b are random and $c \equiv_k a \cdot b$.

Similar to the MASCOT triple generation protocol (Keller et al, CCS 2016). Based on Oblivious Transfer.

1. Each party P_i chooses $b^i \in \mathbb{Z}_{2^{k+s}}$ and $\boldsymbol{a}^i \in (\mathbb{Z}_2)^\tau$. Let $\boldsymbol{a} = (\sum_i \boldsymbol{a}^i) \bmod 2^{k+s}$ and $b = (\sum_i b^i) \bmod 2^{k+s}$, notice that

$$\boldsymbol{c} \equiv_{k+s} \boldsymbol{a} \cdot b \equiv_{k+s} \sum_i \boldsymbol{a}^i \cdot b^i + \sum_{i \neq j} \boldsymbol{a}^i \cdot b^j$$

General Idea (high level) ii

2. Every ordered pair of parties (P_i, P_j) runs OT to get

$$\boldsymbol{c}^i_{i,j} + \boldsymbol{c}^j_{i,j} \equiv_{k+s} \boldsymbol{a}^i \cdot b^j,$$

   where P_i has $\boldsymbol{c}^i_{i,j}$, P_j has $\boldsymbol{c}^j_{i,j}$, and the modulo congruence is performed component-wise.

3. Each party P_i computes:

$$\boldsymbol{c}^i = \boldsymbol{a}^i \cdot b^i + \sum_{j \neq i} (\boldsymbol{c}^i_{i,j} + \boldsymbol{c}^i_{j,i}) \mod 2^{k+s}.$$

   Notice that $\sum_i \boldsymbol{c}^i \equiv_{k+s} \boldsymbol{a} \cdot b$.

Combine:

1. Sample $\boldsymbol{r}, \hat{\boldsymbol{r}} \in (\mathbb{Z}_{2^{k+s}})^{\tau}$.
2. Each party P_i sets

$$a^i = \sum_{h=1}^{\tau} r_h \boldsymbol{a}_h^i \mod 2^{k+s}, \qquad c^i = \sum_{h=1}^{\tau} r_h \boldsymbol{c}_h^i \mod 2^{k+s}$$

$$\hat{a}^i = \sum_{h=1}^{\tau} \hat{r}_h \boldsymbol{a}_h^i \mod 2^{k+s}, \qquad \hat{c}^i = \sum_{h=1}^{\tau} \hat{r}_h \boldsymbol{c}_h^i \mod 2^{k+s}$$

It holds that $a \cdot b \equiv_{k+s} c$ and $\hat{a} \cdot b \equiv_{k+s} \hat{c}$ where $a \equiv_{k+s} \sum_i a^i$ and similarly for b, c, $\hat{a}$ and $\hat{c}$.

- At this point the shares are authenticated (using a MAC functionality) and the triple $([\hat{a}], [b], [\hat{c}])$ is sacrificed to check correctness of $([a], [b], [c])$

Conclusions

We develop an efficient dishonest majority MPC protocol for computation over $\mathbb{Z}_{2^k}$.

- New number-theoretic tricks introduced to overcome the difficulties of working over a ring as $\mathbb{Z}_{2^k}$:
  - Zero-divisors!
  - Non-invertible elements!
  - Taking dot product with random vectors is not a 2-universal function!
- First efficient, information-theoretic secure, homomorphic authentication scheme modulo 2^k.

Future work

- Implementation and performance test[5]
    - Preprocessing is theoretically slower than MASCOT
    - We expect SPD$\mathbb{Z}_{2^k}$'s online phase to be faster in practice since each individual operation is faster
- Develop sub-protocols for basic primitives like inequality and equality tests, bit comparisons, bit decomposition, shifting, etc.
    - This is not trivial since, for example, shifting down means dividing by 2, which is not possible directly.
- Constructing a information-theoretic secure protocol modulo 2^k in the honest majority setting.[6]

---

[5] This is ongoing joint work with Alexandra Institute, Denmark

[6] This is ongoing joint work with the Cryptology group at CWI

Thank you!