More Efficient Dishonest Majority Secure Computation over $\mathbb{Z}_{2^k}$ via Galois Rings

Read at ia.cr/2022/815

Daniel Escudero[1], Chaoping Xing[2] and Chen Yuan[2]

[1]J.P. Morgan AI Research, New York, USA.

[2]Shanghai Jiao Tong University, Shanghai, China

Introduction

Secure Multiparty Computation (MPC)

- Consider n parties $P_1, \ldots, P_n$ with **private inputs** $x_1, \ldots, x_n$
- They wish to **compute** a function $z = f(x_1, \ldots, x_n)$ **without** a **trusted third party**.
- Even if an adversary **actively corrupts** $n - 1$ out of the n parties (**dishonest majority**), the input of the remaining **honest party** is **kept private**.

The functon f is usually described as an **arithmetic circuit** over some **finite ring**.

Typical choice of ring: $\mathbb{Z}_M$, the ring of integers modulo M.

- If M is large enough, arithmetic over the **integers** $\mathbb{Z}$ can be "emulated"
- **Binary computation** is achieved by taking $M = 2$.

Finite fields (*i.e.* obtained when M is a **prime**) is good for protocol design

- Every **non-zero** element is **invertible**

Taking $M = 2^k$ has **advantages** (some of which have been verified experimentally in (Damgård *et al.*, S&P'19)):

- Reducing modulo 2^k—for certain values of k like 32 or 64—comes **for free** in modern **computer architectures**.
- $\mathbb{Z}_{2^k}$ is "more **compatible**" with **binary computation**.

But $\mathbb{Z}_{2^k}$ has **disadvantages** with respect to **finite fields**:

- **Non-invertible** elements
- **Zero divisors** (*e.g.* $2^{k-1} \cdot 2 = 0$ in this ring)

Previous Protocols for Finite Fields (Dish. Majority)

- BeDOZa (Bendlin *et al.*, EC'11)
- SPDZ (Damgård *et al.*, CRYPTO'12 and ESORICS'13)
- MASCOT (Keller *et al.*, CCS'16)
- Overdrive (Keller *et al.*, EC'18)
- TopGear (Baum *et al.*, SAC'19)
- LeMans (Rachuri and Scholl, CRYPTO'22)

They all use **additive secret-sharing** with **MACs** with preprocessed **multiplication triples**.

Previous Protocols for $\mathbb{Z}_{2^k}$ (Dish. Majority)

For $k \geq 1$:

- SPD$\mathbb{Z}_{2^k}$ (Cramer *et al.*, CRYPTO'18)
- Improved Multiplication Triple Generation (Rathee *et al.*, CANS'19)
- Mon$\mathbb{Z}$a (Catalano *et al.*, PKC'20)
- Overdrive2k (Orsini *et al.*, RSA'22)
- MHz2k (Cheon *et al.*, CRYPTO'21)

They all use the **"SPDZ2k trick"**: work over a larger ring $\mathbb{Z}_{2^{k+s}}$

For $k = 1$ (*i.e.* **binary computation**):

- TinyOT (Nielsen *et al.*, CRYPTO'12)
- MiniMAC (Damgård and Zakarias, TCC'13)
- Committed MPC (Frederiksen *et al.*, PKC'18)
- Secret-Sharing Based MPC Protocol (Cascudo and Gundersen, TCC'20)

Note: Most of the finite-field-based protocols work for $\mathbb{Z}_2 = \mathbb{F}_2$.

Our Work

We present a **dishonest majority** MPC protocol over the **ring** $\mathbb{Z}_{2^k}$ with good **communication trade-offs** with respect to previous works.

Main technical insights:

- Novel use of **Galois rings** in the dishonest majority setting
- Novel use of **RMFEs** that does not require re-encoding rounds
- Novel use of **MFEs** to achieve OLE over ring extensions.

Our protocol, unlike SPD$\mathbb{Z}_{2^k}$, also works for $k = 1$ (*i.e.* **binary computation**)

Communication: Offline Phase

Total **offline communication** divided by $n(n-1)$

Protocol	Offline phase	$s = 64$
This work	$8161.6 \cdot k$	$5142.5 \cdot k$
$\mathrm{SPD}\mathbb{Z}_{2^k}$	$2(k+2s)(4k+9s)$	$2(k+128)(4k+576)$
Cascudo *et al.* (for $k=1$)	$462.21 \cdot \ell$	9706.41

For example, for $k = 32$ and $s = 64$, our protocol uses $\approx 73\%$ the communication of $\mathrm{SPD}\mathbb{Z}_{2^k}$.

Communication: Online Phase

Total **online communication** divided by $(n-1)$

	Online phase	$s = 64$	$s = 128$
This work	$19.68 \cdot k$	$12.4 \cdot k$	$12.84 \cdot k$
SPD$\mathbb{Z}_{2^k}$	$4(k+s)$	$4(k+64)$	$4(k+128)$
Cascudo *et al.* (for $k = 1$)	—	10.2	10.42

For example, for $s = 64$ and $k = 32$ our protocol has around the same complexity as SPD$\mathbb{Z}_{2^k}$, but for $k = 16$ our protocol uses $\approx 62\%$ the communication of SPD$\mathbb{Z}_{2^k}$.

Technical Details

Galois Rings

A **Galois ring** is a degree-d extension of the ring $\mathbb{Z}_{2^k}$, namely $R = \mathbb{Z}_{2^k}[x]/(f(x))$, where $f(x)$ is a degree-d irreducible polynomial over $\mathbb{Z}_{2^k}$.

Previously used in **honest majority** protocols to achieve Shamir secret-sharing over $\mathbb{Z}_{2^k}$.

Ours is the **first use** of **Galois rings** in the **dishonest majority** setting.

Important property of Galois rings

Given uniformly random $\alpha \in R$ and $\delta, \epsilon \in R$ where $\delta \neq 0$, the probability of $\delta \cdot \alpha = \epsilon$ is $\leq 2^{-d}$.

This property enables **MPC over the extension ring** R following templates from previous works:

- **Secret-share** an element $x \in R$ as $\langle x \rangle = (\llbracket x \rrbracket, \llbracket \alpha \rrbracket, \llbracket x \cdot \alpha \rrbracket)$.
  - $\llbracket \cdot \rrbracket$ denotes **additive secret-sharing** over R.
- **Additions** are local.
- For **multiplications**, preprocess **Beaver triples** $(\langle a \rangle, \langle b \rangle, \langle c \rangle)$.
- Use the **AMD code** to verify reconstructions are **correct**.

From MPC over R to MPC over $\mathbb{Z}_{2^k}$

Elements in R can be seen as **polynomials** of degree $\leq d-1$ over $\mathbb{Z}_{2^k}$

We can obtain **MPC over the base ring** $\mathbb{Z}_{2^k}$ by first instantiating MPC over R and then **encoding** $x \in \mathbb{Z}_{2^k}$ as a **constant polynomial** in R.

- **Wasteful** in terms of **communication**!

Reverse Multiplication-Friendly Embeddings (RMFEs)

An **RMFE** is a pair (ϕ, ψ) where $\phi : \mathbb{Z}_{2^k}^{\ell} \to R$ and $\psi : R \to \mathbb{Z}_{2^k}^{\ell}$ are $\mathbb{Z}_{2^k}$-linear maps, such that for every $x, y \in \mathbb{Z}_{2^k}^{\ell}$:

$$x \star y = \psi(\phi(x) \cdot_R \phi(y)).$$

Main idea: Multiplication over R can be used to instantiate ℓ **simultaneous** computations over $\mathbb{Z}_{2^k}$.

- We can take $d \approx 4.92 \cdot \ell$ (Cramer *et al.*, CRYPTO'21), so **no communication overhead** (asymptotically)!

Encoding vectors

Secret-share a vector $\boldsymbol{x} \in \mathbb{Z}_{2^k}^{\ell}$ by secret-sharing $\langle x \rangle$, where $x \in R$ is **any** element such that $\psi(x) = \boldsymbol{x}$.

Proposed in (Abspoel *et al.*, ASIACRYPT'21) in the **honest majority** setting, used in **our work** in the **dishonest majority** scenario.

Better than the approach in (Cascudo and Gundersen, TCC'20)

Fact

Secure multiplication is possible by preprocessing **quintuples**

$$(\langle a \rangle, \langle b \rangle, \langle \phi \circ \psi(a) \rangle, \langle \phi \circ \psi(b) \rangle, \langle \phi \circ \psi(a) \cdot \phi \circ \psi(b) \rangle).$$

OLE over the Galois Ring R

OLE Functionality

(Important **building block** for our **preprocessing**)

Given **two parties** P and Q:

- P **inputs** $a, b \in R$, Q **inputs** $x \in R$
- Q **receives** $y = a \cdot x + b$.

Example: Finite Field Extensions

Consider the case $k = 1$, so R is a **field extension** of degree d of $\mathbb{F}_2$.

- **Write** $x = \sum_{i=1}^{d} x_i \cdot \beta_i$ and $a = \sum_{i=1}^{d} a_i \cdot \beta_i$, where $x_i, a_i \in \mathbb{F}_2$ and $\{\beta_i\}_i$ is a $\mathbb{F}_2$-basis.
- **Write** $a \cdot x = \sum_{i,j} (a_i \cdot x_j) \cdot (\beta_i \cdot \beta_j)$.
- Use **Oblivious Transfer (OT)** for every term $a_i \cdot x_j$.

Requires d^2 calls to OT!

Can be brought down to d calls using the **OT extension** protocol from (Keller *et al.*, CRYPTO'15)

Question: What about the case $k > 1$?

The d^2 approach still works, but it is not clear how to get **OLE** over the **extension ring** R while only using d calls to an **OLE functionality** over the **base ring** $\mathbb{Z}_{2^k}$.

- **OT extension** does **not** help

Multiplication Friendly-Embeddings

An **MFE** is a pair (ρ, μ) where $\rho : \mathbb{Z}_{2^k}^t \to R$ and $\mu : R \to \mathbb{Z}_{2^k}^t$ are $\mathbb{Z}_{2^k}$-linear maps, such that for every $x, y \in R$:

$$x \cdot y = \rho(\mu(x) \star \mu(y)).$$

We can take $t \approx 5.12 \cdot d$ (Cramer *et al.*, CRYPTO'21).

OLE over R from OLE over $\mathbb{Z}_{2^k}$ using MFEs

Recall goal. Given two parties P and Q:

- P inputs $a, b \in R$, Q inputs $x \in R$
- Q receives $y = a \cdot x + b$.

Protocol with MFEs

- P **maps** $\mu(a)$ and $\mu(b)$, and Q **maps** $\mu(x)$.
- Run t **OLE instances** over the **base ring** $\mathbb{Z}_{2^k}$ so that Q **obtains** $\mu(a) \star \mu(x) + \mu(b)$
- Q **applies** ρ to **get** $\rho(\mu(a) \star \mu(x)) + \rho(\mu(b)) = a \cdot x + b$.

Our **OLE protocol** for **extension rings** generalizes approaches that use **OT extension**.

Instead of requiring **quadratic** number of calls d^2 to an **OLE** over the **base ring** $\mathbb{Z}_{2^k}$, a **linear** $t = \Theta(d)$ numbers of calls is needed.

Possible **applications** whenever **OLE** over **ring extensions** is needed.

Offline Phase

Authenticating values

We design a method to **promote** an **unauthenticated** value $[\![x]\!]$ **to authenticated** $\langle x \rangle$ (possibly adding errors).

- Requires **OLE** over the **Galois ring** R
- The **adversary** can **introduce** certain **errors** that may **not** be **detected**
- We show that, even in that case, inputs can be **"extracted"** (which ensures correct **simulation**)

See **full paper** for details.[1]

[1] We also address a bug found in a similar proof in MASCOT and (Cascudo and Gundersen, TCC'20)

Generating $(\langle a \rangle, \langle b \rangle, \langle \tau(a) \rangle, \langle \tau(b) \rangle, \langle \tau(a)\tau(b) \rangle)$

1. Obtain pairs $(\langle a \rangle, \langle \tau(a) \rangle)$ and $(\langle b \rangle, \langle \tau(b) \rangle)$

1. Obtain $([\![a]\!], [\![\tau(a)]\!])$ and $([\![b]\!], [\![\tau(b)]\!])$ **locally**
2. **Promote** them to $(\langle a \rangle, \langle \tau(a) \rangle)$ and $(\langle b \rangle, \langle \tau(b) \rangle)$
3. **Check** for **errors**

Generating $(\langle a \rangle, \langle b \rangle, \langle \tau(a) \rangle, \langle \tau(b) \rangle, \langle \tau(a)\tau(b) \rangle)$

2. Use OLE to compute $[\![\tau(a) \cdot \tau(b)]\!]$ from $[\![\tau(a)]\!]$ and $[\![\tau(b)]\!]$

Use **OLE** for the **cross-terms**

Generating $(\langle a \rangle, \langle b \rangle, \langle \tau(a) \rangle, \langle \tau(b) \rangle, \langle \tau(a)\tau(b) \rangle)$

3. Authenticate $[\![\tau(a) \cdot \tau(b)]\!] \to \langle \tau(a) \cdot \tau(b) \rangle$

And then look for **errors** using a **cut-and-choose**-based check.

Future Directions

Efficient computations over **Galois ring** extensions

Better permutation techniques for **non-amortized computation**

Evaluate practicality of computing **RMFEs** and **MFEs**

Dishonest majority MPC over $\mathbb{Z}_{2^k}$ with good communication trade-offs with respect to previous works.

- Novel use of Galois rings
- Novel use of RMFEs
- Novel use of MFEs for OLE over ring extensions.
- Also works for binary computation

THANK YOU!