Fully Secure MPC and zk-FLIOP Over Rings: New Constructions, Improvements and Extensions

CRYPTO 2024

Anders Dalskov[1] Daniel Escudero[2] Ariel Nof[3]

August 21, 2024

[1]Partisia, Denmark

[2]**J.P. Morgan AlgoCRYPT CoE + AI Research, U.S.A.**

[3]Bar Ilan University, Israel

Secure Multiparty Computation

A set of n parties $P_1, \ldots, P_n$ wish to compute an arithmetic circuit C on private inputs $x_1, \ldots, x_n$, in such a way that an adversary corrupting t out of the n parties learns nothing about the honest parties' inputs.

Our setting

- Honest majority ($t < n/2$)
- Active security
- (Mostly) statistical security
- Guaranteed output delivery

Our goal

Minimizing communication complexity of active security w.r.t. passively secure protocols.

Known Results

A common paradigm in the design of actively secure protocols is to start from a passive protocol and **compile** it to an **actively secure** one.

- [GMW87; IPS08; DPSZ12; BFO12; GIP15; CDESX18; FLNW17; LN17; Chi+18; FL19; GSZ20]

What is the cost in terms of communication?

Recent work by Boneh et al. [BBCGI19] introduced the notion of *zero-knowledge fully linear interactive oracle proofs* (zk-FLIOP).

With this, a communication of $\texttt{passive} + O(\log |C|)$ is achievable.

Multiple works instantiate this idea to achieve **active security** at the same **communication** cost (asymptotically in $|C|$) as **passive**: [BGIN19; BGIN20; GSZ20; GLOPS21].

Round Complexity[1]

What about the number of interaction rounds required to compute the circuit?

- **Passive security:** $O(\text{depth}(C))$ rounds
- **Active security with abort:** $O(\text{depth}(C))$ rounds
- **Active security with G.O.D.:** $O(\text{depth}(C)) + O(\log |C|)$ rounds

[1] We assume **Fiat-Shamir**. This round-count already takes this into consideration.

rounds

$$\text{Active + GOD} = \texttt{passive} + O(\log|C|)$$

There seems to be a gap between the efficiency (in terms of # rounds) of passive MPC vs active MPC with GOD!

Not only relevant in theory: The extra $O(\log|C|)$ rounds can have detrimental impact in the performance of the final protocol!

- High-latency settings (*e.g.* WAN networks or large # parties): number of rounds affect runtimes more severely.
- "Shallow" circuits where $\log|C| \gg \text{depth}(C)$: the extra term is not negligible

Can full security for an arbitrary number of parties be achieved while incurring in the following additive overheads with respect to state-of-the-art semi-honest protocols: (1) communication overhead that is logarithmic in the circuit size, and (2) ***constant*** *overhead in the round complexity?*

This, assuming Fiat-Shamir.

Our Results

We present an **actively** secure protocol with **G.O.D.**, which has only a **constant number** of additional **rounds** w.r.t the best passive protocol.

For ANY Secret-Sharing Scheme

A factor of $n\times$ more communication for the *offline phase*.

Same communication as passive security for the *online phase*.

For Replicated Secret-Sharing

Same communication as passive security.

	(n,t)	Additive Overhead		Security	Secret sharing scheme
		Communication cost	No. of Rounds		
Boneh et al. [BBCGI19]	(3,1)	$O(\log \lvert C \rvert)$	$O(1)$	with abort	replicated
Boyle et al. [BGIN19]	(3,1)	$O(\log \lvert C \rvert)$	O(1)	Full	replicated
Boneh et al. [BBCGI19]	$(2t+1,t)$	$O(\sqrt{\lvert C \rvert})$	$O(1)$	with abort	replicated
Boyle et al. [BGIN20]	$(2t+1,t)$	$O(\log \lvert C \rvert)$	$O(1)$	with abort	Any linear scheme
Boyle et al. [BGIN20]	$(2t+1,t)$	$O(\log^2 \lvert C \rvert)$	$O(\log \lvert C \rvert)$	Full	replicated
Goyal et al. [GSZ20]	$(2t+1,t)$	$O(\log \lvert C \rvert)$	$O(\log \lvert C \rvert)$	Full	Shamir
This work	$(2t+1,t)$	$O(\log \lvert C \rvert)$	$O(1)$	Full	Any linear scheme, $O(n^2 \lvert C \rvert)$ preprocessing
This work	$(2t+1,t)$	$O(\log \lvert C \rvert)$	$O(1)$	Full	replicated
Furukawa et al. [FL19]	$(3t+1,t)$	$O(1)$	$O(\lvert \text{depth}(C) \rvert)$	with abort	Shamir
Dalskov et al. [DEN22]	$(3t+1,t)$	$O(1)$	$O(1)$	Full	Replicated
This work	$(3t+1,t)$	$O(\log(\lvert C \rvert))$	$O(1)$	Full	Any linear scheme

Features of our approach

Works for any ring (even **non-commutative**!)

In particular, works for fields and the ring $\mathbb{Z}_{2^k}$.

For example, we can compile the non-commutative protocol from [ES21].

Builds on zk-FLIOPs in a **black-box way**.

Improvements on zk-FLIOPs can be immediately be applied to our compiler.

Improvements on zk-FLIOPs for Galois Rings

Galois Rings

Ring extensions of $\mathbb{Z}_{2^k}$.

Useful rings for multiple applications such as ML.

Extensions are useful for enabling polynomial interpolation.

zk-FLIOP over these rings is already known from the work of Boneh et al. [BBCGI19].

We improve zk-FLIOPs over Galois Rings by making use of **Reverse Multiplication-Friendly Embeddings (RMFEs)** [EHLXY23].

See the paper for further details on this front.

Challenges with Prior Works

zk-FLIOPs: A Recipe for Sublinear Overhead

zk-FLIOPs ([BBCGI19]) enable a prover to prove succinctly relations under the following conditions:

- Relation is degree-2
- The values are "committed" in such a way that they can be "robustly opened".

The authors note this can be used to **compile** passive MPC protocols into **active** security at **sublinear** cost.

Two approaches:

- Single prover
- Distributed prover

Single prover approach

Each party proves to the others that the messages they sent during the protocol execution are correct.

Works as long as:

- Messages sent by the parties are "degree-2"
- The values are "robustly-shared" among the parties

Instantiated for **three parties** and **one corruption**:

- [BBCGI19] for security with **abort**
- [BGIN19] for **guaranteed output delivery**

Crucial observation for three parties / one corruption:
Either the prover is corrupt and the (two) verifiers are honest,
or the prover is honest and one of the verifiers is corrupt.

$\Rightarrow$ For a corrupt prover, the values are trivially "robustly shared".

For general $t \geq 2$:
Security with **abort** is possible thanks to the robustness of (say) **Shamir secret-sharing** in the honest majority regime [BGIN20].

G.O.D. remains challenging: not "robust enough".

Distributed prover approach

The parties check that all secret-shared multiplications are correct, distributively emulating the prover in the zk-FLIOP.

Boneh et al. [BBCGI19] use this to obtain an **actively** secure protocol with **abort** for general n using **replicated secret-sharing**.

Challenge to get G.O.D. :
it is difficult to identify who cheated if the verification fails.

Solution in [BGIN20]:

- Perform **binary search** to first identify the exact multiplication that failed the zk-FLIOP
- Develop an "expensive" (*i.e.* non-sublinear) method to detect who cheated in this single multiplication

The extra $\log|C|$ rounds that we want to avoid come from the BINARY SEARCH!

Summary:

- Single-prover approach yields G.O.D. without extra rounds but only for $n = 3$
- Distributed-prover yields G.O.D. for general n but costs $\log|C|$ extra rounds.

Our Approach

General idea

We aim at extending the **single-prover approach** for general n.

Recap: DN07 protocol

Let $[x]$ denote Shamir secret-sharing
Let $\langle x \rangle$ denote additive secret-sharing.

DN07 Multiplication ([DN07]):
Given two secret-shared inputs $[x], [y]$, and a preprocessed $([r], \langle r \rangle)$:

- Compute locally $\langle xy + r \rangle = [x] \cdot [y] + \langle r \rangle$
- Send the shares of $\langle xy + r \rangle$ to P_1, so that P_1 reconstructs $xy + r$
- P_1 sends $xy + r$ to all parties
- Parties compute locally $[xy] = (xy + r) - \langle r \rangle$.

FACT: Active security boils down to ensuring that these multiplications are performed correctly.

Let $\{[x_k], [y_k], [z_k]\}_{k=1}^{|C|}$ be the secret-shared multiplications after executing the aforementioned protocol.

GOAL: Check that the parties sent the correct messages to each other.

Let $x_k^{(i)}$, $y_k^{(i)}$ and $r^{(i)}$ be the shares of $[x]$, $[y]$ and $\langle r \rangle$ held by party P_i.
Let $\mathrm{msg1}_k^{(i)}$ be the message sent by P_i to P_1 (should be equal to $x_k^{(i)} y_k^{(i)} + r_k^{(i)}$).
Let $\mathrm{msg2}_k^{(i)}$ be the message sent by P_1 to P_i, (should be equal to $\sum_{i'=1}^{n} \mathrm{msg1}_k^{(i')}$).

GOAL

Check that, for all $k \in \{1, \ldots, |C|\}$ and $i \in \{1, \ldots, n\}$:

$$\mathrm{msg1}_k^{(i)} - (x_k^{(i)} y_k^{(i)} + r_k^{(i)}) = 0 \quad \text{and} \quad \mathrm{msg2}_k^{(i)} - \sum_{i'=1}^{n} \mathrm{msg1}_k^{(i')} = 0$$

Compress via Random Linear Combinations

Sample $\gamma_1, \ldots, \gamma_{|C|}$ uniformly at random.

Let $\text{msg1}^{(i)} = \sum_{k=1}^{|C|} \gamma_k \cdot \text{msg1}_k^{(i)}$, $\text{msg2}^{(i)} = \sum_{k=1}^{|C|} \gamma_k \cdot \text{msg2}_k^{(i)}$ and $r^{(i)} = \sum_{k=1}^{|C|} \gamma_k \cdot r_k^{(i)}$

GOAL

Check that

$$\text{msg1}^{(i)} - r^{(i)} - \sum_{k=1}^{|C|} \gamma_k \cdot x_k^{(i)} y_k^{(i)} = 0 \quad \text{and} \quad \text{msg2}^{(i)} - \sum_{i'=1}^{n} \text{msg1}^{(i')} = 0$$

for all $i \in \{1, \ldots, n\}$

Let each party "commit" to their compressed messages by broadcasting them.

- Every P_i broadcasts $\mathsf{msg1}^{(i)}$ and $\mathsf{msg2}^{(i)}$.
- P_1 broadcasts $\mathsf{msg2}^{(i)}$ and $\mathsf{msg1}^{(i)}$.
- Parties handle inconsistencies.[a]

[a] This includes checking that $\forall i: \ \mathsf{msg2}^{(i)} - \sum_{i'=1}^{n} \mathsf{msg1}^{(i')} = 0$.
Inconsistencies lead to semi-corrupt pairs.

Now the parties agreed on a "compressed transcript", and they must check that

$$\forall i \in \{1, \ldots, n\} : \quad \mathsf{msg1}^{(i)} - r^{(i)} - \sum_{k=1}^{|C|} \gamma_k \cdot x_k^{(i)} y_k^{(i)} = 0$$

Applying zk-FLIOP in a Black-Box Way

$$\forall i \in \{1, \ldots, n\} : \quad \text{msg1}^{(i)} - r^{(i)} - \sum_{k=1}^{|C|} \gamma_k \cdot x_k^{(i)} y_k^{(i)} = 0$$

zk-FLIOP requires (1) A degree-2 statement ✓, and
(2) Values must be **robustly shared**.

Observation:
$[x_k]$ is secret-shared, hence, the i-th share $x_k^{(i)}$ itself is **also secret-shared**!
(potentially under a *slightly different* secret-sharing scheme).

Let us denote these sharings by $[x_k^{(i)}|_i]$ and $[y_k^{(i)}|_i]$.
$\text{msg1}^{(i)}$ is **public** and hence $[\text{msg1}^{(i)}|_i]$ can be **locally** computed.

$$\forall i \in \{1, \ldots, n\} : \quad [\mathrm{msg1}^{(i)}|_i] - r^{(i)} - \sum_{k=1}^{|C|} \gamma_k \cdot [x_k^{(i)}|_i][y_k^{(i)}|_i] = 0$$

Problem: the parties only have *additive shares* $\langle r_k \rangle$ for $k \in \{1, \ldots, |C|\}$ (and hence $\langle r \rangle = \sum_{k=1}^{|C|} \gamma_k \cdot \langle r_k \rangle$), but we need $[r^{(i)}|_i]$.

We require a **conversion protocol** that transforms $\langle r \rangle$ into $[r^{(i)}|_i]$.

IMPORTANT

This would take place during the **preprocessing phase**.

Converting $\langle r \rangle$ into $[r^{(i)}|_i]$.

For Replicated Secret-Sharing

The PRSS methods to generate random sharings non-interactively already yield the desired sharings! [DEN22].

For Other Schemes

Generate the pair $([r_k], \langle r_k \rangle)$ as follows:

- Each P_i samples $r_k^{(i)}$ and secret-shares as $[r_k^{(i)}]$ (**quadratic communication!**)
- The sampled values constitute $\langle r_k \rangle$.
- Parties add $[r_k] = \sum_{i=1}^{n} [r_k^{(i)}]$.

Recall that $r^{(i)} = \sum_{k=1}^{|C|} \gamma_k \cdot r_k^{(i)}$.

Recall the parties need $[r^{(i)}|_i]$ for the zk-FLIOP.

The parties have $[r^{(i)}]$ (which is similar, but *not* the same).

SOLUTION:

- Let P_i distribute shares $[r^{(i)}|_i]$.
- Check consistency* with respect to $[r^{(i)}]$.

Now the parties can apply the zk-FLIOP!

Other details in the paper

- Description for general secret-sharing schemes and general rings.
- Subtle details in using zk-FLIOP in our setting.
- Improvements for zk-FLIOPs in the concrete case of Galois Rings.
- Proofs and self-contained protocols.

[BBCGI19] D. Boneh et al. **"Zero-Knowledge Proofs on Secret-Shared Data via Fully Linear PCPs".** In: *CRYPTO*. 2019.

[BFO12] E. Ben-Sasson, S. Fehr, and R. Ostrovsky. **"Near-Linear Unconditionally-Secure Multiparty Computation with a Dishonest Minority".** In: *CRYPTO 2012*. 2012.

[BGIN19] E. Boyle et al. **"Practical Fully Secure Three-Party Computation via Sublinear Distributed Zero-Knowledge Proofs".** In: *ACM CCS*. 2019.

[BGIN20] E. Boyle et al. **"Efficient Fully Secure Computation via Distributed Zero-Knowledge Proofs".** In: *ASIACRYPT*. 2020.

References ii

[CDESX18] R. Cramer et al. **"SPD$\mathbb{Z}_{2^k}$: Efficient MPC mod 2^k for Dishonest Majority".** In: *CRYPTO 2018*. 2018.

[Chi+18] K. Chida et al. **"Fast Large-Scale Honest-Majority MPC for Malicious Adversaries".** In: *CRYPTO 2018*. 2018.

[DEN22] A. P. K. Dalskov, D. Escudero, and A. Nof. **"Fast Fully Secure Multi-Party Computation over Any Ring with Two-Thirds Honest Majority".** In: *ACM CCS 2022*. 2022.

[DN07] I. Damgård and J. B. Nielsen. **"Scalable and Unconditionally Secure Multiparty Computation".** In: *CRYPTO*. 2007.

[DPSZ12] I. Damgård et al. **"Multiparty Computation from Somewhat Homomorphic Encryption".** In: *CRYPTO 2012*. 2012.

[EHLXY23] D. Escudero et al. **"Degree-D Reverse Multiplication-Friendly Embeddings: Constructions and Applications".** In: *ASIACRYPT 2023*. 2023.

[ES21] D. Escudero and E. Soria-Vazquez. **"Efficient information-theoretic multi-party computation over non-commutative rings".** In: *CRYPTO 2021*. 2021.

[FL19] J. Furukawa and Y. Lindell. **"Two-Thirds Honest-Majority MPC for Malicious Adversaries at Almost the Cost of Semi-Honest".** In: *ACM CCS 2019*. 2019.

[FLNW17] J. Furukawa et al. **"High-Throughput Secure Three-Party Computation for Malicious Adversaries and an Honest Majority".** In: *EUROCRYPT 2017*. 2017.

[GIP15] D. Genkin, Y. Ishai, and A. Polychroniadou. **"Efficient Multi-party Computation: From Passive to Active Security via Secure SIMD Circuits".** In: *CRYPTO 2015*. 2015.

[GLOPS21] V. Goyal et al. **"ATLAS: Efficient and Scalable MPC in the Honest Majority Setting".** In: *CRYPTO*. 2021.

[GMW87] O. Goldreich, S. Micali, and A. Wigderson. **"How to Play any Mental Game or A Completeness Theorem for Protocols with Honest Majority".** In: *ACM STOC 1987*. 1987.

[GSZ20] V. Goyal, Y. Song, and C. Zhu. **"Guaranteed Output Delivery Comes Free in Honest Majority MPC".** In: *CRYPTO 2020*. 2020.

[IPS08] Y. Ishai, M. Prabhakaran, and A. Sahai. **"Founding Cryptography on Oblivious Transfer - Efficiently".** In: *CRYPTO 2008*. 2008.

[LN17] Y. Lindell and A. Nof. **"A Framework for Constructing Fast MPC over Arithmetic Circuits with Malicious Adversaries and an Honest-Majority".** In: *ACM CCS 2017*. 2017.

- Actively secure MPC with G.O.D. with the same round-count as semi-honest
- Same **online** communication as passive (asymptotically)
- Same **offline** communication as passive for replicated secret-sharing
- A factor of n more communication in the offline phase for other schemes (**can we improve this?**)
- Works for general rings (even non-commutative).
- Improved zk-FLIOP constructions over Galois rings using RMFEs.

Thank you!