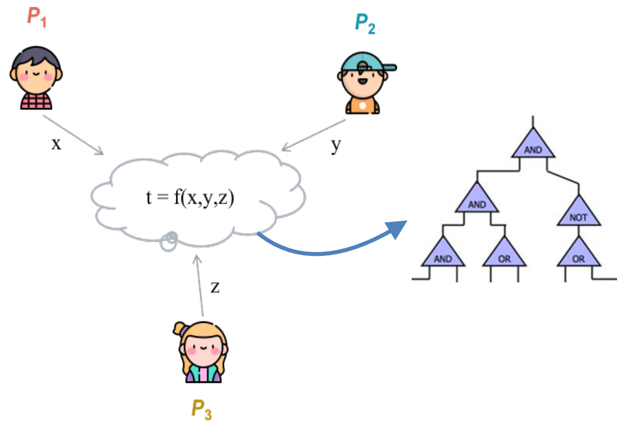
Efficient 3PC for Binary Circuits with Application to Maliciously-Secure DNN Inference

Yun Li, Yufei Duan, Zhicong Huang, Cheng Hong, Chao Zhang, Yifan Song

Problem Setting

Three-Party Computation (on Binary Circuits)



ADVERSARIAL BEHAVIOR

Malicious

CORRUPTION THRESHOLD

Honest-Majority

COMMUNICATION NETWORK

Authenticated

Secure

Synchronized

Previous Works

Cut-and-choose + Beaver Triples

- Use cut-and-choose to generate valid Beaver triples and use them to check online AND triples

- **Low computational cost**
- **High communication complexity**
  - ***9 bits*** per AND gate per party [FLNW17]
  - ***7 bits*** per AND gate per party [ABFL+17]

Distributed Zero-knowledge Proofs

- Use distributed zero-knowledge proofs to verify local semi-honest AND gate computations

- **High computational cost**
  - rely on ***binary extension fields*** [BGIN19]
- **Low communication complexity**
  - 1 bit per AND gate per party (amortized)

Question

Can we achieve **the same communication complexity as [BGIN19]**

and also **a comparable concrete efficiency as [FLNW17, ABFL+17] ?**

Our Results

- **A Maliciously-Secure 3PC Protocol for Binary Circuits**
  - **Communication complexity**
    - the same as [BGIN19]
    - 9× lower than [FLNW17], 7× lower than [ABFL+17]
  - **Computational cost**
    - comparable with [FLNW17]
    - 3.5× faster than [BGIN19]

- **Application to DNN Inference**
  - SqueezeNet
  - DenseNet
  - ResNet50

Starting Point: [BGIN19]

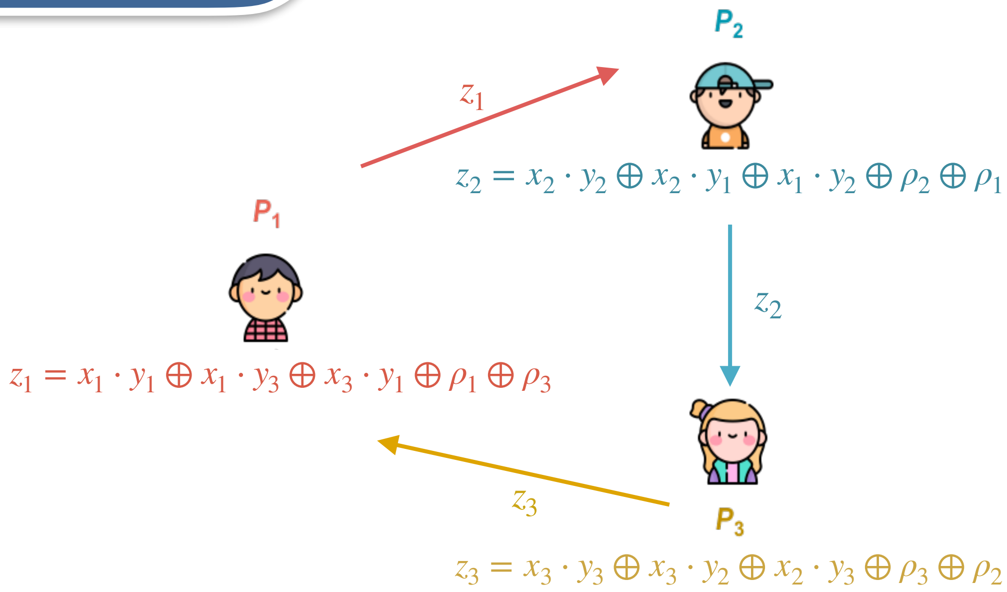
Key to **sublinear verification communication cost**:

distributed zero-knowledge proofs

Review: [BGIN19]

1. **Semi-honest**: Replicated Secret Sharing

Computing AND gate: $[z] = [x] \cdot [y]$



Starting Point: [BGIN19]

Key to **sublinear verification communication cost**:

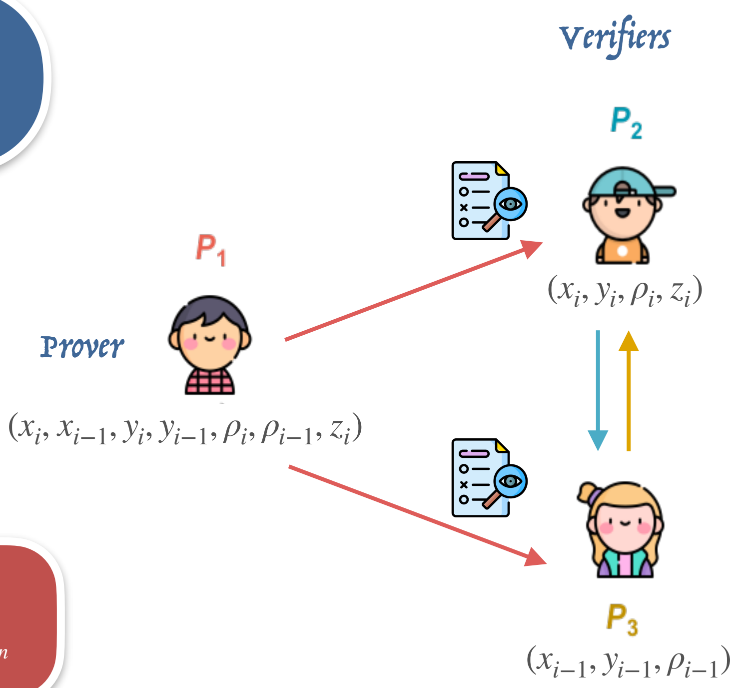
distributed zero-knowledge proofs [BBCG+19]

Review: [BGIN19]

2. Verification: Verify semi-honest AND computations

$$z_i = x_i \cdot y_i \oplus x_i \cdot y_{i-1} \oplus x_{i-1} \cdot y_i \oplus \rho_i \oplus \rho_{i-1}$$

Lift $\mathbb{F}_2$ computations up to its extension field $\mathbb{F}_{2^n}$ to apply distributed zero-knowledge proofs over $\mathbb{F}_{2^n}$



Our Techniques: Basic Construction

Lift $\mathbb{F}_2$ computations up to **prime fields** $\mathbb{F}_p$, and exploit the **algebraic structure** of prime fields to further reduce the computational cost of distributed zkps

1. Reduce the relation

$$z_i = x_i \cdot y_i \oplus x_i \cdot y_{i-1} \oplus x_{i-1} \cdot y_i \oplus \rho_i \oplus \rho_{i-1}$$

$$\Downarrow$$

$$a \cdot b \oplus c \cdot d \oplus e \oplus f = 0$$

P_2

$a := x_i$

$c := y_i$

$e := x_i \cdot y_i \oplus z_i \oplus \rho_i$

P_1

$b := y_{i-1}$

$d := x_{i-1}$

$f := \rho_{i-1}$

P_3

Our Techniques: Basic Construction

Lift $\mathbb{F}_2$ computations up to **prime fields** $\mathbb{F}_p$,
and exploit the **algebraic structure** of prime fields to
further reduce the computational cost of dzkps

2. Transform to Prime Fields

$$\alpha \oplus \beta = \alpha + \beta - 2\alpha\beta \mod p = \alpha(1-2\beta) + \beta \mod p$$

$$\begin{aligned} a \cdot b \oplus c \cdot d \oplus e \oplus f \quad = \quad & -2(a \cdot c \cdot (1-2e)) \cdot (b \cdot d \cdot (1-2f)) \\ & +(c \cdot (1-2e)) \cdot (d \cdot (1-2f)) + (a \cdot (1-2e)) \cdot \\ & (b \cdot (1-2f)) - \frac{1}{2}((1-2e) \cdot (1-2f)) + \frac{1}{2} \mod p \end{aligned}$$

Our Techniques: Basic Construction

Lift $\mathbb{F}_2$ computations up to **prime fields** $\mathbb{F}_p$,
and exploit the **algebraic structure** of prime fields to
further reduce the computational cost of dzkps

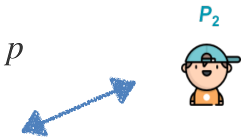
2. Transform to Prime Fields

$$a \cdot b \oplus c \cdot d \oplus e \oplus f = \cdots\cdots$$

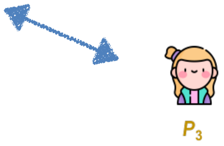
$$\Downarrow$$

$$\sum_{k=1}^{4} g_k \cdot h_k + 1/2 = 0 \mod p$$

$$g_1 := -2a \cdot c \cdot (1-2e) \mod p$$
$$g_2 := c \cdot (1-2e) \mod p$$
$$g_3 := a \cdot (1-2e) \mod p$$
$$g_4 := -(1-2e)/2 \mod p$$



$$h_1 := b \cdot d \cdot (1-2f) \mod p$$
$$h_2 := d \cdot (1-2f) \mod p$$
$$h_3 := b \cdot (1-2f) \mod p$$
$$h_4 := 1-2f \mod p$$



Our Techniques: Basic Construction

Lift $\mathbb{F}_2$ computations up to **prime fields** $\mathbb{F}_p$, and exploit the **algebraic structure** of prime fields to further reduce the computational cost of dzkps

3. Batch Verifying AND Gates

For some Incorrect AND triple: $\sum_{k=1}^{4} g_k \cdot h_k + 1/2 = a \cdot b \oplus c \cdot d \oplus e \oplus f = 1$

Batch checking m AND triples:

$$\sum_{\ell=1}^{m} (\sum_{k=1}^{4} g_k^{(\ell)} \cdot h_k^{(\ell)} + 1/2) = 0 \iff \text{all AND triples are correct (given } m < p\text{)}$$

No need for random linear combination

Our Techniques: Optimizations

Lift $\mathbb{F}_2$ computations up to **prime fields** $\mathbb{F}_p$, and exploit the **algebraic structure** of prime fields to further reduce the computational cost of dzkps

Use the Powers of 2 as Coefficients for Linear Combination

Batch checking m AND triples:

Enable to use **native CPU computations** for batch computing $\mathbb{F}_p$ data directly from $\mathbb{F}_2$ data (without first transforming them to $\mathbb{F}_p$)

$$\sum_{\ell=1}^{m}(\sum_{k=1}^{4} g_k^{(\ell)} \cdot h_k^{(\ell)} + 1/2) = 0 \quad \Rightarrow \quad \sum_{i=1}^{m} 2^{(i-1) \bmod 32}(\sum_{k=1}^{4} g_k^{(i)} \cdot h_k^{(i)} + 1/2) = 0$$

requirement: $< p$

Choosing $p = 2^{61} - 1$ is sufficient for any $m \leq 2^{33}$

Our Techniques: Optimizations

Lift $\mathbb{F}_2$ computations up to **prime fields** $\mathbb{F}_p$,
and exploit the **algebraic structure** of prime fields to further reduce the computational cost of dzkps

Other Optimizations

Lookup Table for Polynomial Evaluation

Use Mersenne Fields for Fast Arithmetic Computation

Speed up Inner-product Operations

Performance

Performance on Pure Binary Circuits

Depth		Semi	Ours	BGIN19	FLNW17
1	LAN Time	0.12	1.15	3.42	0.95
	WAN Time	2.97	3.31	5.67	11.74
10	LAN Time	0.12	1.14	3.78	0.90
	WAN Time	2.18	2.58	4.61	11.41
100	LAN Time	0.12	1.14	3.84	0.91
	WAN Time	5.18	5.85	6.78	14.94
1000	LAN Time	0.18	1.16	3.87	0.96
	WAN Time	41.05	41.83	42.76	51.05
10000	LAN Time	0.70	1.36	4.05	1.50
	WAN Time	401.60	402.47	403.35	412.35
	Comm.	24.00	24.80	24.57	224.16

Table 2: Time (s), communication (MB) for computing circuits of 64 million AND gates with different depths.

Comparison with [BGIN19]

- **Communication**: almost the same
- **End-to-end time**: $3 \sim 3.4\times$ faster in LAN, $1 \sim 1.8\times$ faster in WAN

Performance

Performance on Pure Binary Circuits

Depth		Semi	Ours	BGIN19	FLNW17
1	LAN Time	0.12	1.15	3.42	0.95
	WAN Time	2.97	3.31	5.67	11.74
10	LAN Time	0.12	1.14	3.78	0.90
	WAN Time	2.18	2.58	4.61	11.41
100	LAN Time	0.12	1.14	3.84	0.91
	WAN Time	5.18	5.85	6.78	14.94
1000	LAN Time	0.18	1.16	3.87	0.96
	WAN Time	41.05	41.83	42.76	51.05
10000	LAN Time	0.70	1.36	4.05	1.50
	WAN Time	401.60	402.47	403.35	412.35
	Comm.	24.00	24.80	24.57	224.16

Table 2: Time (s), communication (MB) for computing circuits of 64 million AND gates with different depths.

Comparison with [FLNW17]

- **Communication**: 9× lower
- **End-to-end time**: 1 ~ 1.3× slower in LAN, 2.6 ~ 4.2× faster in WAN

Performance

Performance on DNN Inference

Model	# of Threads		Semi + Semi	SW + Semi[1]	SW + Ours	SW + BGIN19	SW + FLNW17
ResNet-50	1	LAN Time	89.68	336.71	372.97	582.55	391.23
	32	LAN Time	18.66	56.64	63.33	81.48	87.48
	32	WAN Time	544.15	1969.48	2048.89	2096.17	2786.22
		Comm.	7537.86	27791.9	27846.10	27830.40	41114.30
DenseNet	1	LAN Time	63.42	305.89	375.72	622.83	371.124
	32	LAN Time	12.98	57.07	66.13	84.60	94.39
	32	WAN Time	713.42	1994.98	2070.69	2096.17	2842.19
		Comm.	8919.85	31924.50	31993.40	31973.5	48709.60
SqueezeNet	1	LAN Time	13.61	49.13	58.89	106.20	63.09
	32	LAN Time	2.23	9.93	11.28	14.17	15.70
	32	WAN Time	200.19	432.05	448.96	455.29	674.26
		Comm.	1403.22	4803.36	4816.58	4812.76	8047.35

Time cost for achieving malicious security for the binary part

- SW+Ours is 15% ~ 33% of SW+BGIN19 in LAN, and 7% ~ 10% of SW+FLNW17 in WAN

SW: SpdzWise Protocol [ADEN21]

An Extra Finding

A hidden security issue in probabilistic truncation protocols

prepare $[r], [r/2^d]$

$\Downarrow$

compute and open $[x+r] = [x] + [r]$

$\Downarrow$

output $(x+r)/2^d - [r/2^d]$

The same randomness is used for both **protecting the privacy** of the secret value, and **sampling the 1-bit rounding error** probabilistically

An Extra Finding

A hidden security issue
in probabilistic truncation protocols

prepare $[r], [r/2^d]$

$\Downarrow$

compute and open $\underline{[x+r]} = [x] + [r]$

$\Downarrow$

output $\underline{(x+r)/2^d - [r/2^d]}$

Introduced by

- [CH10]

Affected Papers

- [KPPS21]
- [DEK21]
- [PS20]
- [EGK+20]
- [MR18]
- [MZ17]
- ……

Thanks !

liyun19@mails.tsinghua.edu.cn

References

- [BGIN19] Elette Boyle, Niv Gilboa, Yuval Ishai, and Ariel Nof. Practical fully secure three-party computation via sublinear distributed zero-knowledge proofs. In Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, pages 869–886, 2019.
- [FLNW17] Jun Furukawa, Yehuda Lindell, Ariel Nof, and Or Weinstein. High-throughput secure three-party computation for malicious adversaries and an honest majority. In Annual international conference on the theory and applications of cryptographic techniques, pages 225–255. Springer, 2017.
- [ABFL+17] Toshinori Araki, Assi Barak, Jun Furukawa, Tamar Lichter, Yehuda Lindell, Ariel Nof, Kazuma Ohara, Adi Watzman, and OrWeinstein. Optimized honest-majority mpc for malicious adversaries — breaking the 1 billiongate per second barrier. In 2017 IEEE Symposium on Security and Privacy (SP), pages 843–862, 2017.
- [BBCG+19] Dan Boneh, Elette Boyle, Henry Corrigan-Gibbs, Niv Gilboa, and Yuval Ishai. Zero-knowledge proofs on secret-shared data via fully linear pcps. In Annual International Cryptology Conference, pages 67–97. Springer, 2019.
- [ADEN21] Mark Abspoel, Anders Dalskov, Daniel Escudero, and Ariel Nof. An efficient passive-toactive compiler for honest-majority mpc over rings. In International Conference on Applied Cryptography and Network Security, pages 122– 152. Springer, 2021.[CH10] Octavian Catrina and Sebastiaan de Hoogh. Improved primitives for secure multiparty integer computation. In International Conference on Security and Cryptography for Networks, pages 182–199. Springer, 2010.
- [CH10] Octavian Catrina and Sebastiaan de Hoogh. Improved primitives for secure multiparty integer computation. In International Conference on Security and Cryptography for Networks, pages 182–199. Springer, 2010.
- [KPPS21] Nishat Koti, Mahak Pancholi, Arpita Patra, and Ajith Suresh. {SWIFT}: Super-fast and robust {Privacy-Preserving} machine learning. In 30th USENIX Security Symposium (USENIX Security 21), pages 2651–2668, 2021.
- [DEK21] Anders P. K. Dalskov, Daniel Escudero, and Marcel Keller. Fantastic four: Honest-majority four-party secure computation with malicious security. In Michael Bailey and Rachel Greenstadt, editors, 30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021, pages 2183–2200. USENIX Association, 2021.
- [PS20] Arpita Patra and Ajith Suresh. BLAZE: blazing fast privacy-preserving machine learning. In 27th Annual Network and Distributed System Security Symposium, NDSS 2020, San Diego, California, USA, February 23-26, 2020. The Internet Society, 2020.
- [EGK+20] Daniel Escudero, Satrajit Ghosh, Marcel Keller, Rahul Rachuri, and Peter Scholl. Improved primitives for mpc over mixed arithmetic-binary circuits. In Annual International Cryptology conference, pages 823–852. Springer, 2020.
- [MR18] Payman Mohassel and Peter Rindal. Aby3: A mixed protocol framework for machine learning. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018, pages 35–52. ACM, 2018.
- [MZ17] Payman Mohassel and Yupeng Zhang. Secureml: A system for scalable privacy-preserving machine learning. In 2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017, pages 19–38. IEEE Computer Society, 2017.