Fast Large-Scale Honest Majority MPC for Malicious Adversaries

Koji Chida, Koki Hamada, Dai Ikarashi, Ryo Kikuchi

NTT, Japan

Daniel Genkin

University of Michigan

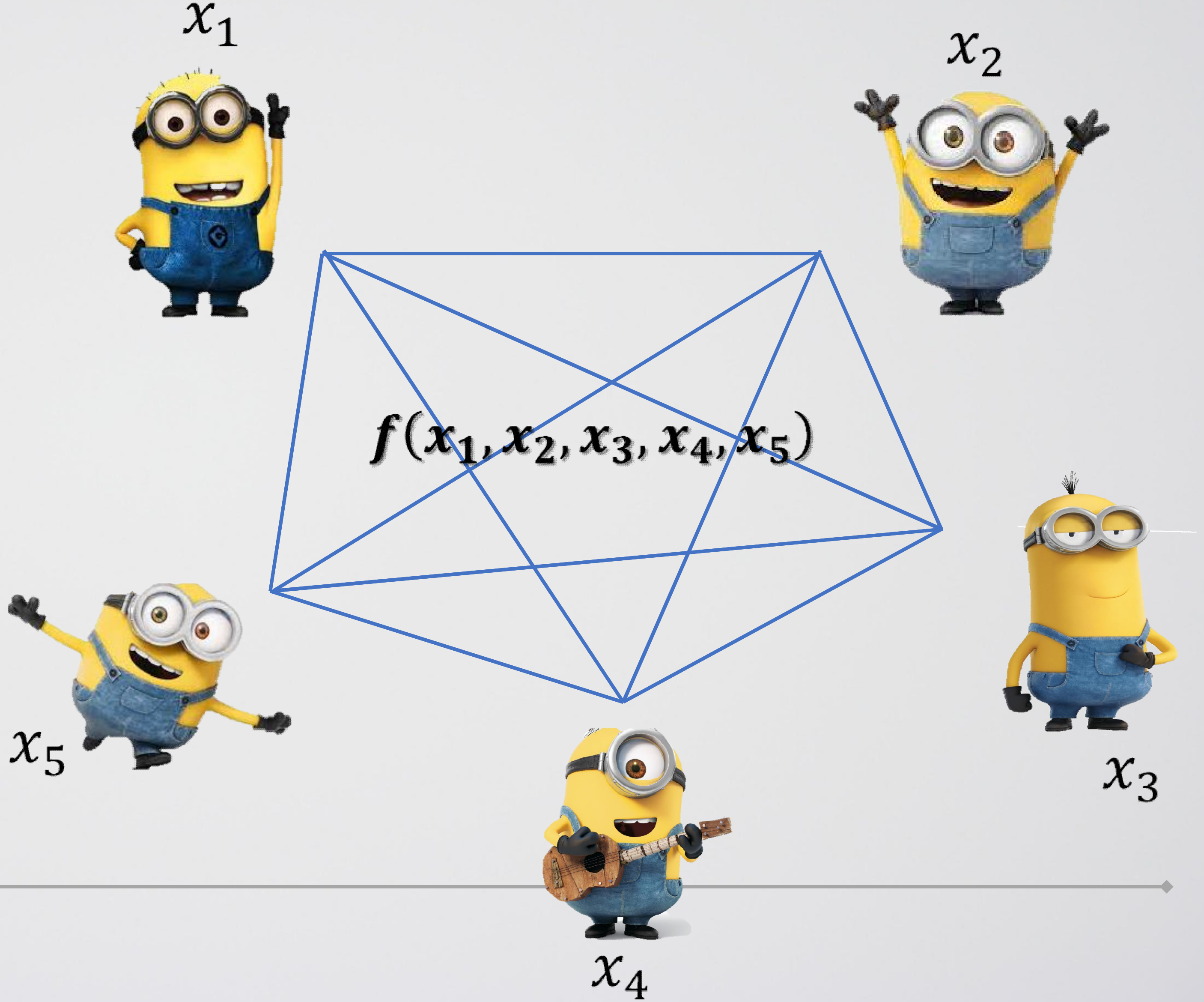
Yehuda Lindell, **Ariel Nof**

Bar-Ilan University Israel

Crypto 2018

The Setting

- n parties wish to compute an **arithmetic** circuit over a field F
- **Malicious** adversary controlling t parties
- **Honest majority** $(t < n/2)$
- **Security with abort**



The Starting Point

1. An observation made by Genkin et al. [GIPST15, GIP16]:
   - In secret-sharing based protocols, many semi-honest multiplication protocols are ***secure up to additive attack*** in the presence of malicious adversaries.

2. For the honest-majority setting, there exists highly efficient semi-honest multiplication protocols with **low and linear communication complexity.**

Our Main Results

- A **information-theoretic** protocol **maliciously secured with abort** at the cost of **running semi-honest protocol δ times**, where δ is such that $\left(\frac{|F|}{3}\right)^{\delta} \geq 2^{\sigma}$ (σ is the security parameter).
  - **For "large" fields, the semi-honest protocol is run only twice!**

- Two instantiations:
  - **3-party with replicated secret sharing**: each party sends ***2 field elements*** per multiplication gate (for large fields).
  - **Multi-party with Shamir's secret sharing**: each party sends ***12 field elements*** per multiplication gate (for large fields).

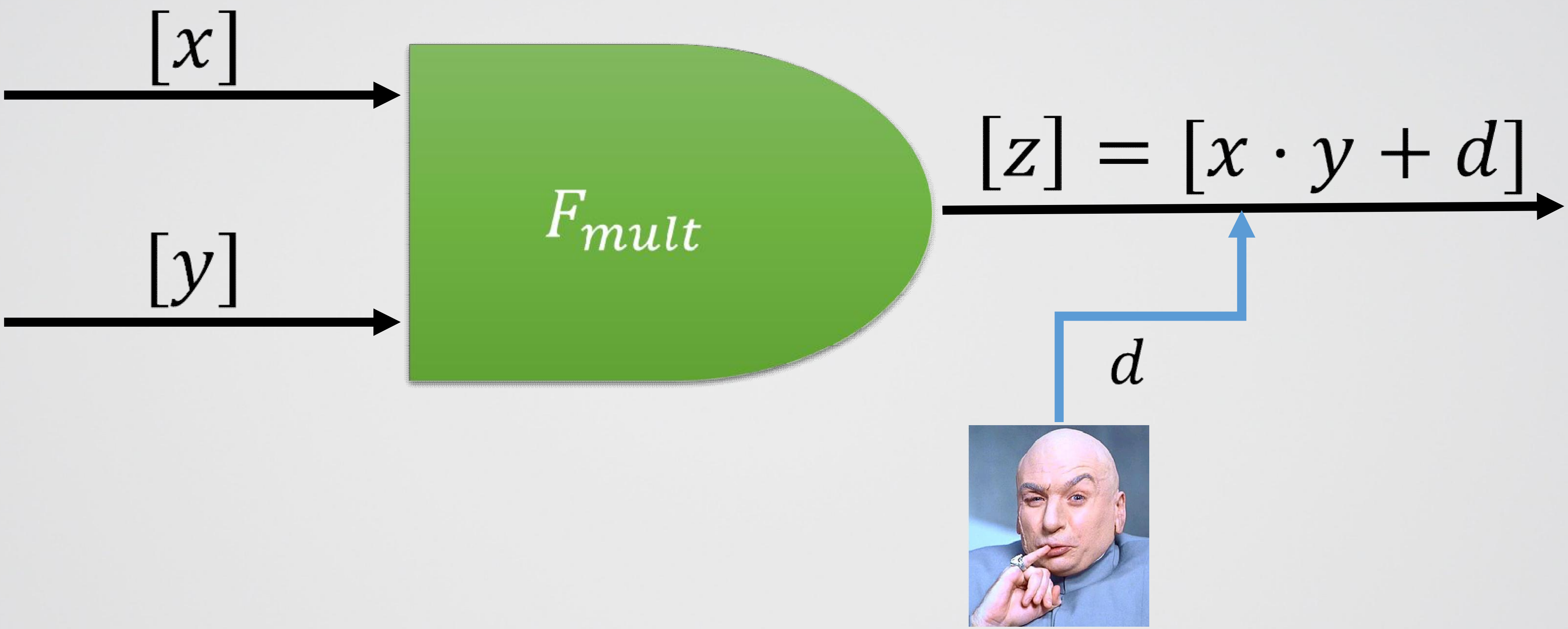
Honest Majority MPC

- **Orders of magnitudes** faster than **dishonest** majority MPC
- $\boldsymbol{t < n/3}$:
  - full security with perfect security and linear complexity can be achieved (HB[08])
  - Concrete efficiency: VIFF[08]
- $\boldsymbol{t < n/2}$:
  - Full security results
    - Computational Model – linear communication complexity using PKC (HN[06])
    - Information-theoretic – best known result: $O(n\log(n))$ (BFO[12])
  - Security with abort
    - Linear complexity and information-theoretic – GIP[15] (no concrete cost)
  - Concrete efficiency:
    - Multi-party: we improve upon the previous best known result (LN[17]) by approximately **twice for a small number of parties** and by up to **10 times for a large number of parties.**

Some Notation

- $[x]$ – a sharing of x.
  - We assume linearity of the secret sharing scheme.
- F_{mult} - a multiplication protocol secure up to additive attack.
- F_{rand} - a sub-protocol to generate random sharings.

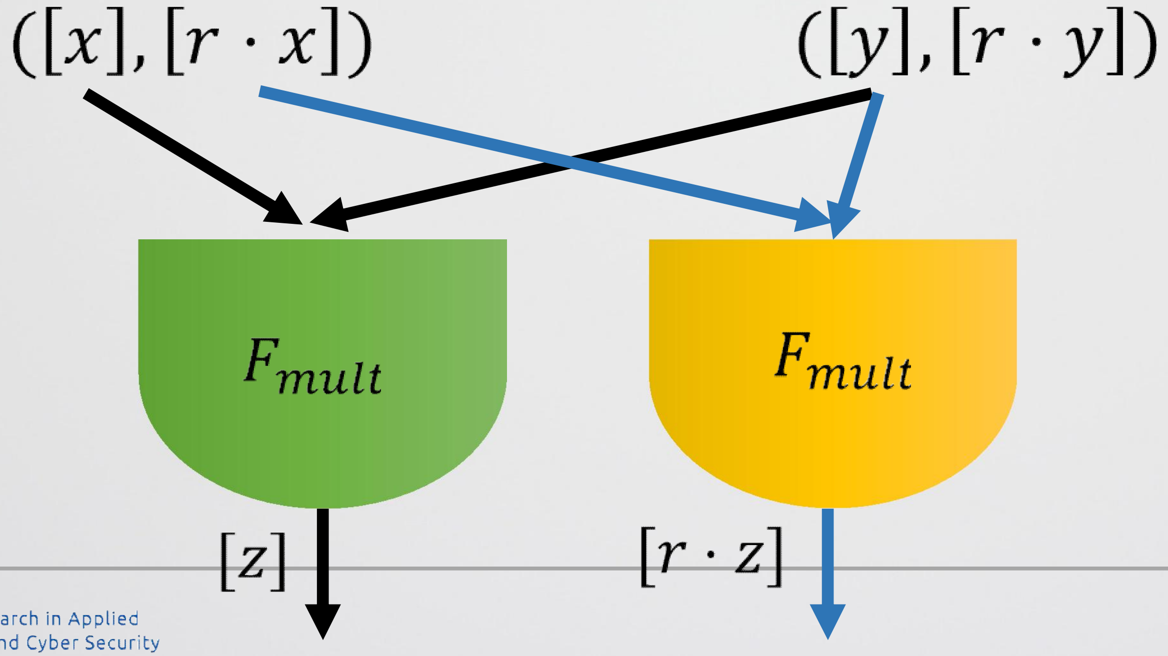
Achieving Malicious Security



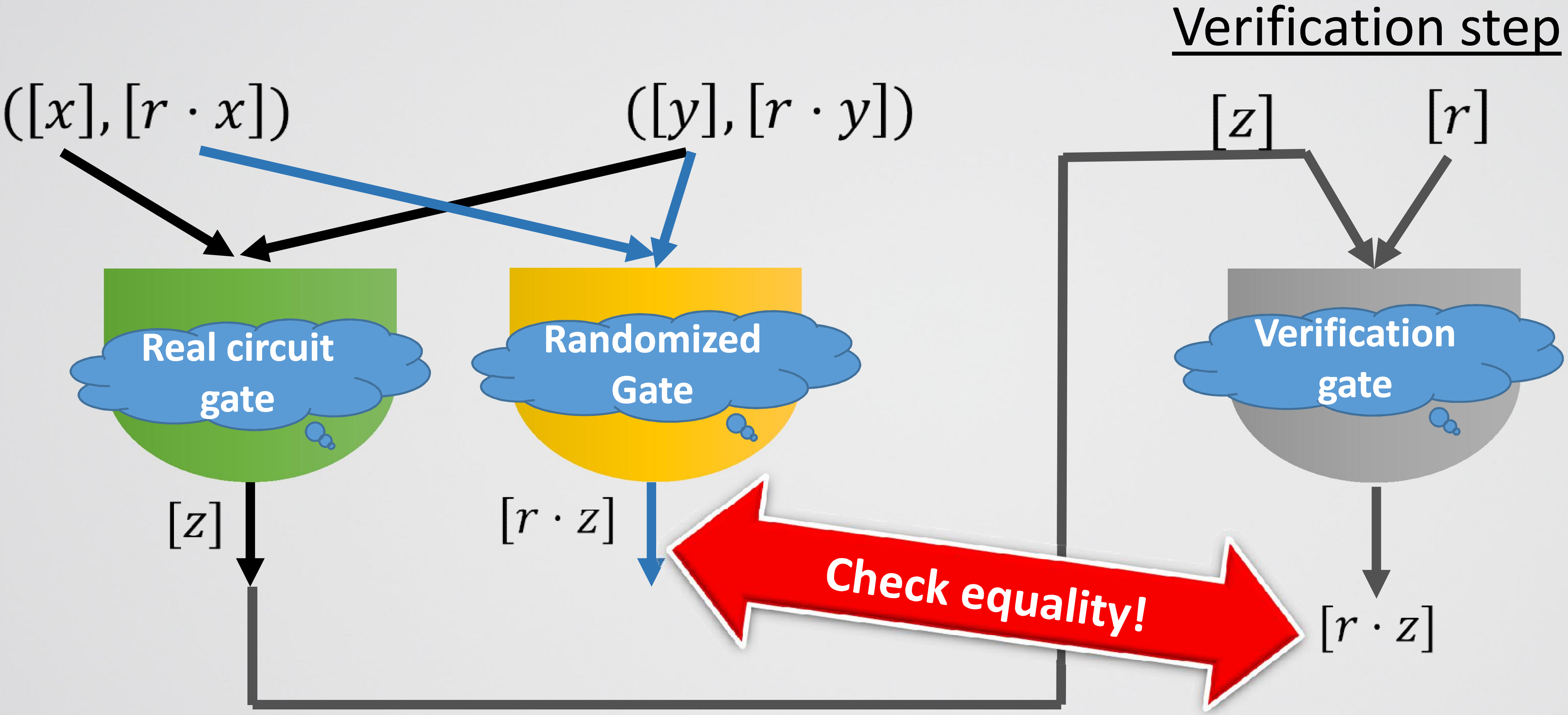
How can the honest parties detect (and abort) when $d \neq 0$?

Cheating Detection - The Main Idea

- Generate a random sharing $[r]$.
- For each wire of the circuit, hold the pair $([x], [r \cdot x])$:
  - Use F_{mult} to randomize the input wires of the circuit
  - For each multiplication gate:

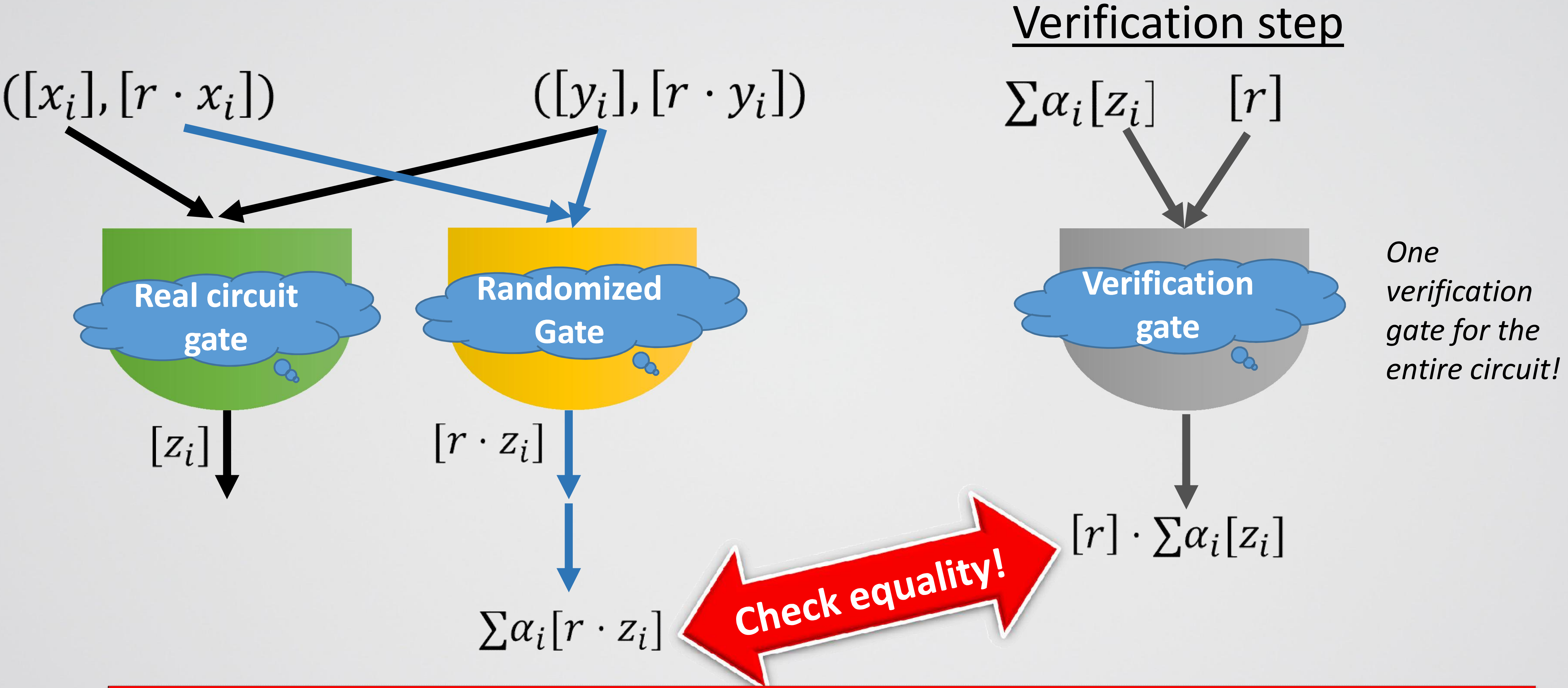


Cheating Detection - The Main Idea



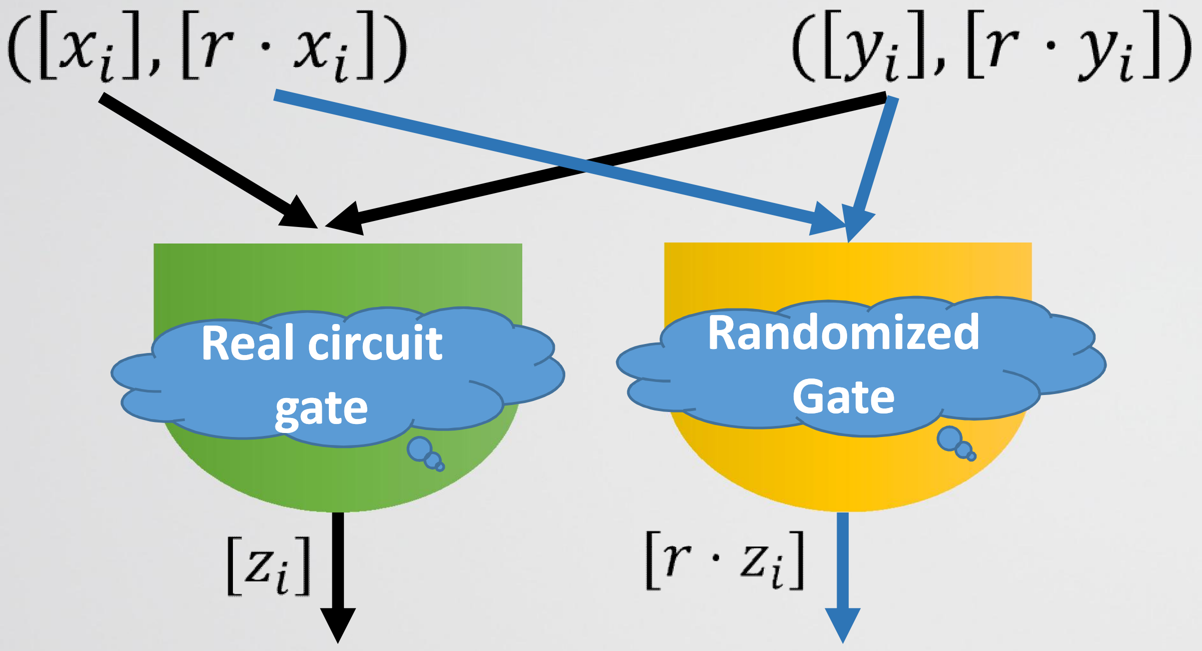
Since r is unknown, then if cheating took place, then the probability that the equality holds is negligible

Cheating Detection - Optimized

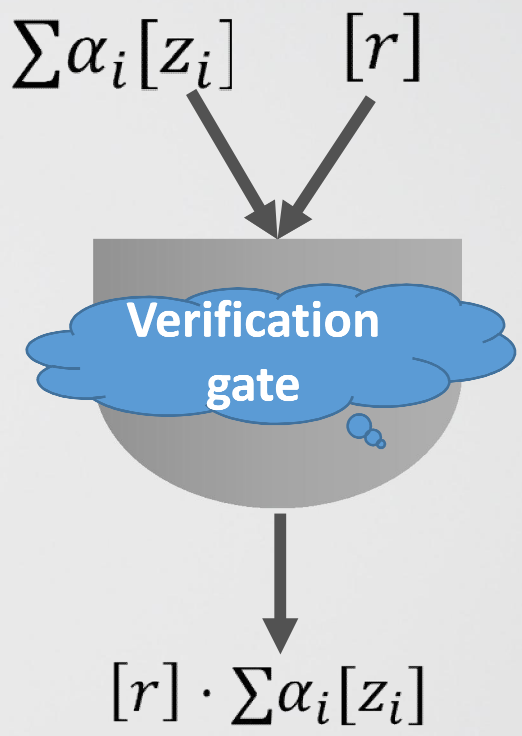


If $z = x\cdot y + d$ for $d \neq 0$, then the honest parties abort w.p. $1-\frac{3}{|F|}$

A security problem!

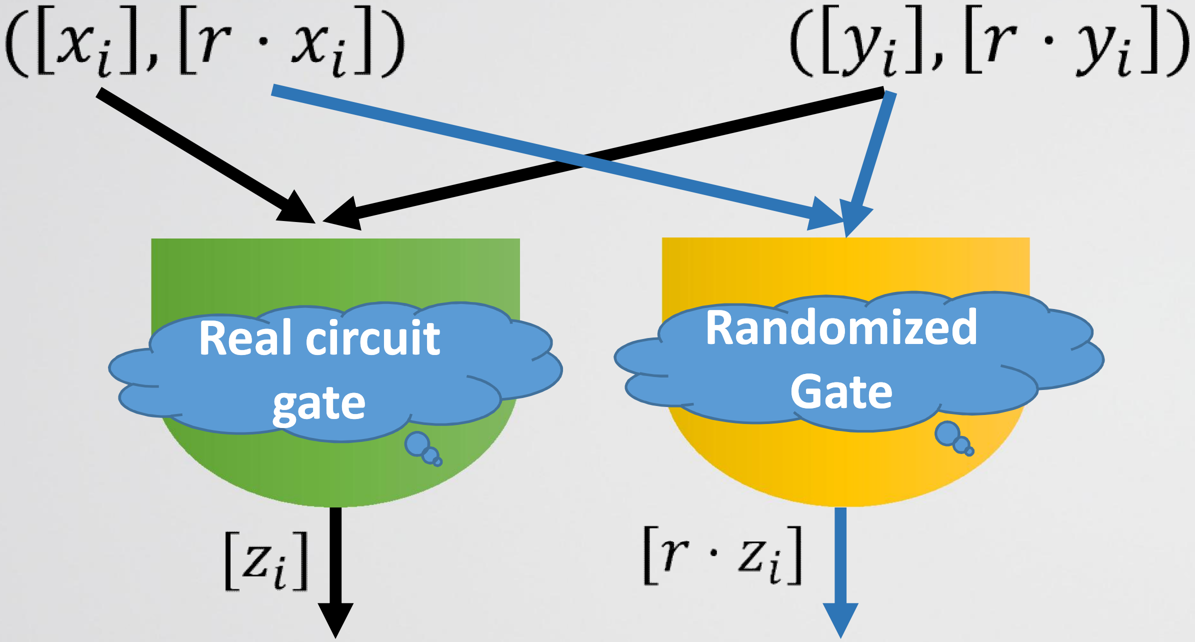


Verification step



This is done **after** the random coefficients have been chosen!!

Cheating Detection - Optimized and Secure



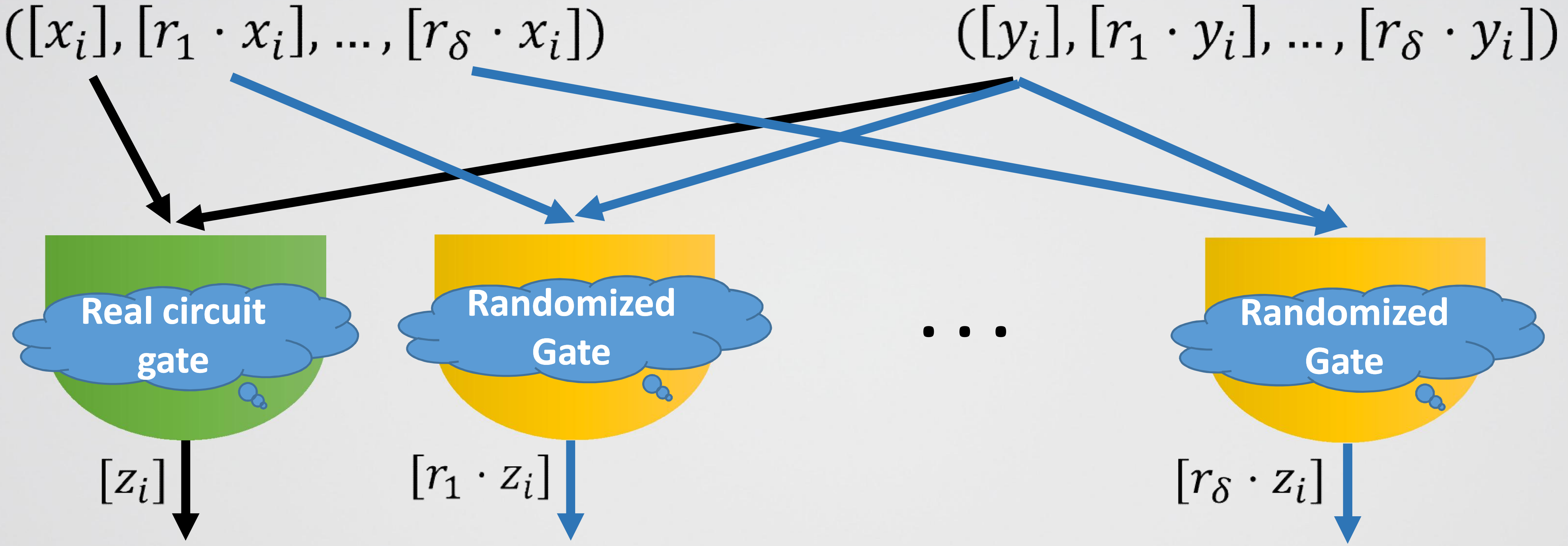
Verification step

Local operation

1. Open $[\mathrm{r}]$
2. Compute $\mathrm{r} \cdot \sum \alpha_i [z_i]$
3. Check that:

$$\sum \alpha_i [r \cdot z_i] = r \cdot \sum \alpha_i [z_i]$$

What about Small fields?



Verification step . . . **Verification step**

$\frac{3}{|F|}$ . . . $\frac{3}{|F|}$ $\leq \left(\frac{3}{|F|}\right)$

Small Fields – Verification

<u>Verification step</u>

1. Open $[\mathrm{r}]$
2. Compute $\mathrm{r} \cdot \sum \alpha_i [z_i]$
3. Check that:

(*) $\sum \alpha_i [r \cdot z_i] = r \cdot \sum \alpha_i [z_i]$

- All α_is and r are publicly known!!
- The values on the wires are known to the **distinguisher** but **not to the simulator**!
- The distinguisher knows whether the equality (*) holds, but the simulator does not!

$$\boldsymbol{Pr[(*) holds\ when\ the\ adversary\ cheats]} \leq \frac{\mathbf{3}}{|\boldsymbol{F}|}$$

Not negligible!

Small Fields – New Verification

Verification step

1. Call F_{rand} to receive $\{[\alpha_i]\}$
2. Open [r]
3. Compute $\mathrm{r} \cdot \sum[\alpha_i] \cdot [z_i]$
4. Check that:

$$\sum[\alpha_i] \cdot [r \cdot z_i] = r \cdot \sum[\alpha_i] \cdot [z_i]$$

Need to call F_{mult} for each gate two more times!!

Computing Sum of Products Efficiently

Example:
Shamir's secret sharing

$$\sum_{i=1}^{m} [\alpha_i]_t \cdot [z_i]_t$$

$[\alpha_1]_t \cdot [z_1]_t$

1. The parties locally multiply their shares

$[\alpha_1 \cdot z_1]_{2t}$

2. Interactive protocol for degree reduction

$[\alpha_1 \cdot z_1]_t$

$[\alpha_2]_t \cdot [z_2]_t$

1. The parties locally multiply their shares

$[\alpha_2 \cdot z_2]_{2t}$

2. Interactive protocol for degree reduction

$[\alpha_2 \cdot z_2]_t$

…

$[\alpha_m]_t \cdot [z_m]_t$

1. The parties locally multiply their shares

$[\alpha_m \cdot z_m]_{2t}$

2. Interactive protocol for degree reduction

$[\alpha_m \cdot z_m]_t$

$$\left[\sum_{i=1}^{m} \alpha_i \cdot z_i\right]_t$$

Computing Sum of Products Efficiently

Example:
Shamir's secret sharing

$$\sum_{i=1}^{m} [\boldsymbol{\alpha}_i]_t \cdot [\boldsymbol{z}_i]_t$$

$[\alpha_1]_t \cdot [z_1]_t$

1. The parties locally multiply their shares

$[\alpha_1 \cdot z_1]_{2t}$

$[\alpha_2]_t \cdot [z_2]_t$

1. The parties locally multiply their shares

$[\alpha_2 \cdot z_2]_{2t}$

$\cdots$

$[\alpha_m]_t \cdot [z_m]_t$

1. The parties locally multiply their shares

$[\alpha_m \cdot z_m]_{2t}$

$$\left[\sum_{i=1}^{m} \alpha_i \cdot z_i\right]_{2t}$$

2. Interactive protocol for degree reduction

Small Fields – New Verification

<u>Verification step</u>

No need to open r!

1. Call F_{rand} to receive $\{[\alpha_i]\}$
2. Open $[\mathrm{r}]$
3. Compute $\mathrm{r} \cdot \sum[\alpha_i] \cdot [z_i]$
4. Check that:

$$\sum[\alpha_i] \cdot [r \cdot z_i] = r \cdot \sum[\alpha_i] \cdot [z_i]$$

Compute this step at the cost of two multiplications for the entire circuit!

Summary

A protocol for large fields

The amortized cost for multiplication gate:
2 calls to $\mathbf{F}_{\mathbf{mult}}$

A protocol for small fields

The amortized cost for multiplication gate:
$(1+\delta)$ calls to $\mathbf{F}_{\mathbf{mult}}$ +
δ calls to $\mathbf{F}_{\mathbf{rand}}$

Experimental Results

- Two instantiations:
  - Replicated secret sharing (3 parties)
  - Shamir’s secret sharing (n parties)

			Open
Replicated	1	0	2
Shamir	6	2	n-1

# of elements sent per party

Experimental Results

- 1,000,000 multiplication gate circuit with different dept
- 61-bi
- **LA** WS region

Can compute 1M gates with 3 parties in **319ms**

Can compute 1M gates with 110 parties in **8.2s**

Circuit Depth	(repli ated)		5	7	9	11	30	50	70	90	10
20	319	826	844	1,058	1,311	1,377	2,769	4,053	5,295	6,586	8,281
100	323	842	989	1,154	1,410	1,477	3,760	6,052	8,106	11,457	15,431
1,000	424	1,340	1,704	1,851	2,243	2,887	12,144	26,310	33,294	48,927	79,728
10,000	1,631	6,883	7,424	8,504	12,238	16,394	61,856	132,160	296,047	411,195	544,525

Execution time in milliseconds

Experimental Results

- 1,000,000 multiplication gate circuit with different depths
- 61-bit Merse
- **WAN conf** gion

Can compute 1M gates with 3 parties in **3s**

Can compute 1M gates with 50 parties in **128s**

Circuit Depth	(repl ted)	3	5	7	9	11	30	0
20	3502	20,492	27,772	28,955	24,482	24,729	87,355	128,366
100	10,712	45,250	53,872	50,719	55,716	56,482	134,860	197,321

Execution time in milliseconds

THANK YOU!

https://eprint.iacr.org/2018/570