DIMACS Workshop 2019

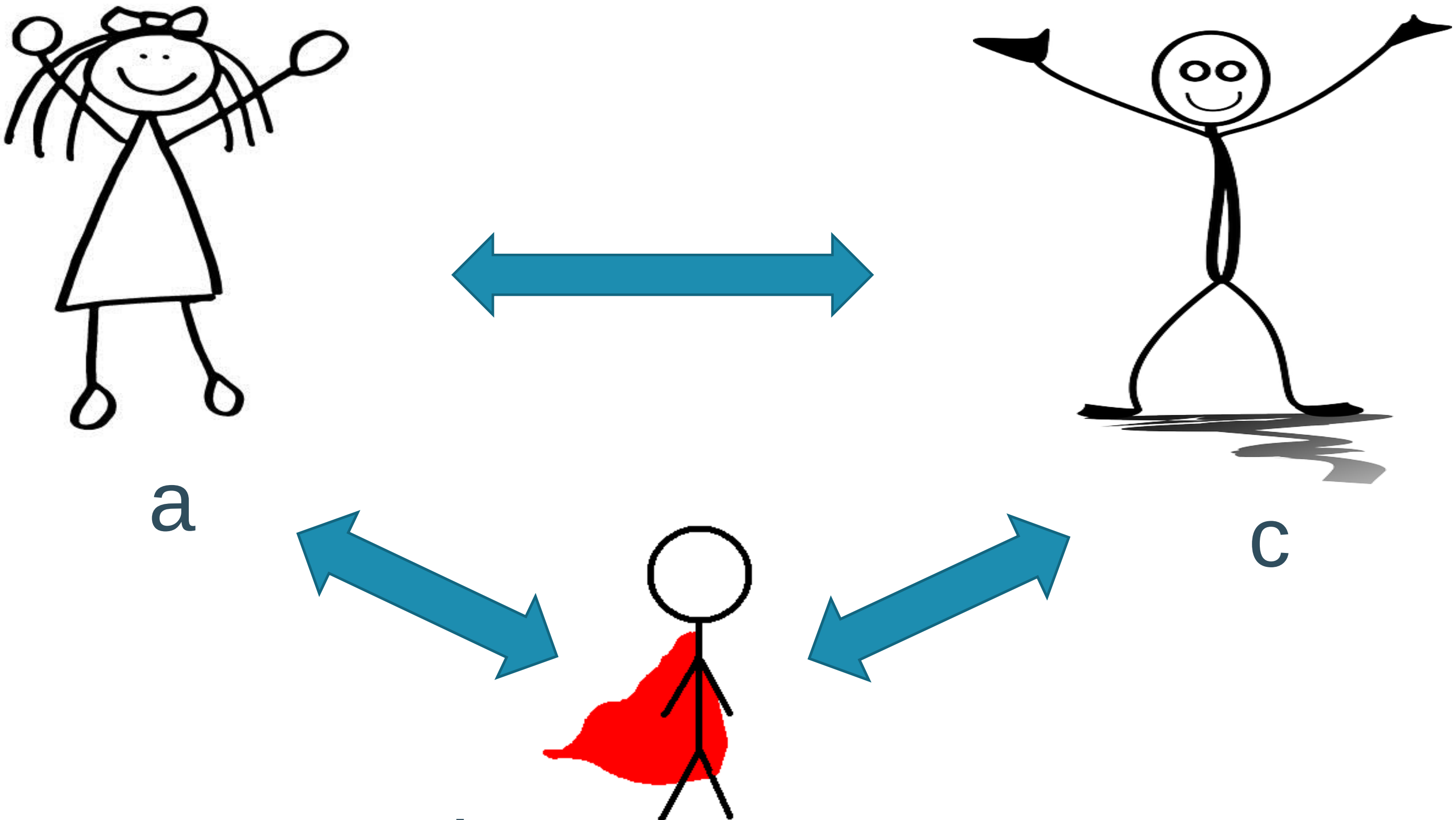
A universal MPC machine*

Dragoș Rotaru

University of Bristol, KU Leuven

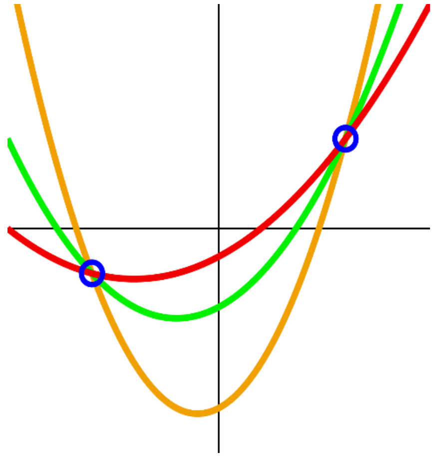
*MArBled Circuits: Mixing Arithmetic and Boolean Circuits with Active Security;
- Joint work with Tim Wood.
- ***https://ia.cr/2019/207***

What is multiparty computation?



Goal: Compute F(a, b, c)

How can we achieve MPC?



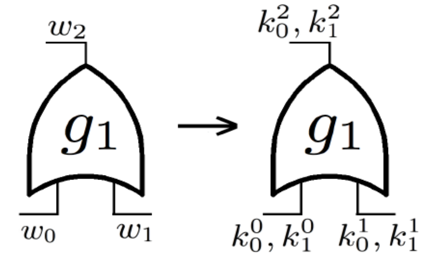


Figure 1: Garbling a single gate

w_0	w_1	w_2
0	0	0
0	1	1
1	0	1
1	1	1

(a) Original Values

w_0	w_1	w_2	garbled value
k_0^0	k_1^0	k_2^0	$H(k_0^0 \| \| k_1^0 \| \| g_1) \oplus k_2^0$
k_0^0	k_1^1	k_2^1	$H(k_0^0 \| \| k_1^1 \| \| g_1) \oplus k_2^1$
k_0^1	k_1^0	k_2^1	$H(k_0^1 \| \| k_1^0 \| \| g_1) \oplus k_2^1$
k_0^1	k_1^1	k_2^1	$H(k_0^1 \| \| k_1^1 \| \| g_1) \oplus k_2^1$

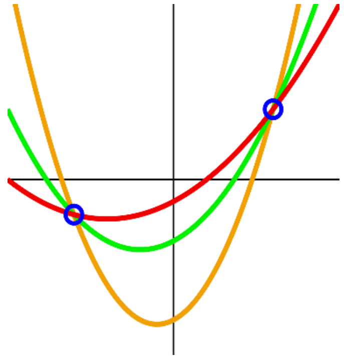
(b) Garbled Values

Figure 2: Computation table for g_1^{OR}

Secret Sharing	Garbled Circuits
Fast networks (LAN)	Slow Networks (WAN)
Arithmetic/Boolean circuits	Boolean circuits
Low depth, many AND gates	Large depth, few AND gates

Dragoș Rotaru
imec-Cosic, Dept. Electrical Engineering
KU LEUVEN

Can we switch between?



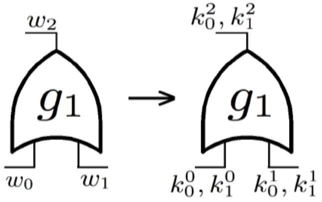


Figure 1: Garbling a single gate

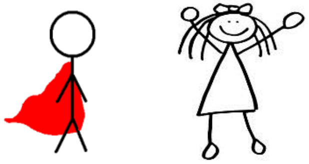
w_0	w_1	w_2
0	0	0
0	1	1
1	0	1
1	1	1

(a) Original Values

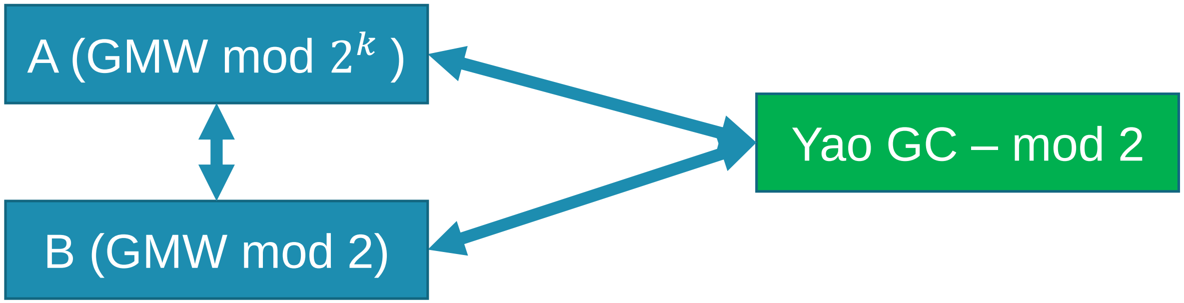
w_0	w_1	w_2	garbled value
k_0^0	k_1^0	k_2^0	$H(k_0^0\|\|k_1^0\|\|g_1) \oplus k_2^0$
k_0^0	k_1^1	k_2^1	$H(k_0^0\|\|k_1^1\|\|g_1) \oplus k_2^1$
k_0^1	k_1^0	k_2^1	$H(k_0^1\|\|k_1^0\|\|g_1) \oplus k_2^1$
k_0^1	k_1^1	k_2^1	$H(k_0^1\|\|k_1^1\|\|g_1) \oplus k_2^1$

(b) Garbled Values

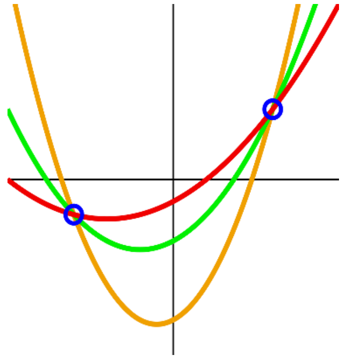
Figure 2: Computation table for g_1^{OR}

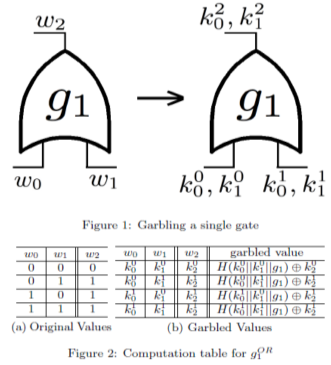


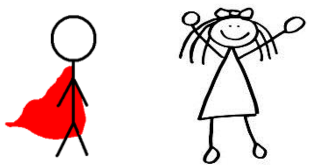
ABY [DMZ'15]



Can we switch between?





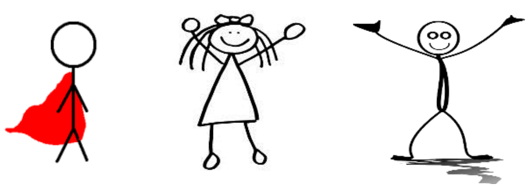


A (GMW mod 2^k)

B (GMW mod 2)

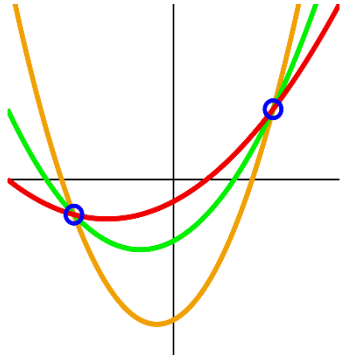
Yao GC – mod 2

ABY [DMZ'15]



ABY3 [MR'18]

Can we switch between?



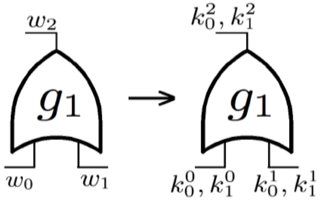


Figure 1: Garbling a single gate

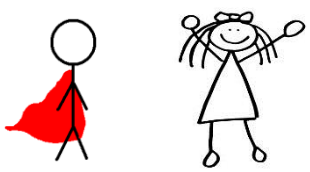
w_0	w_1	w_2
0	0	0
0	1	1
1	0	1
1	1	1

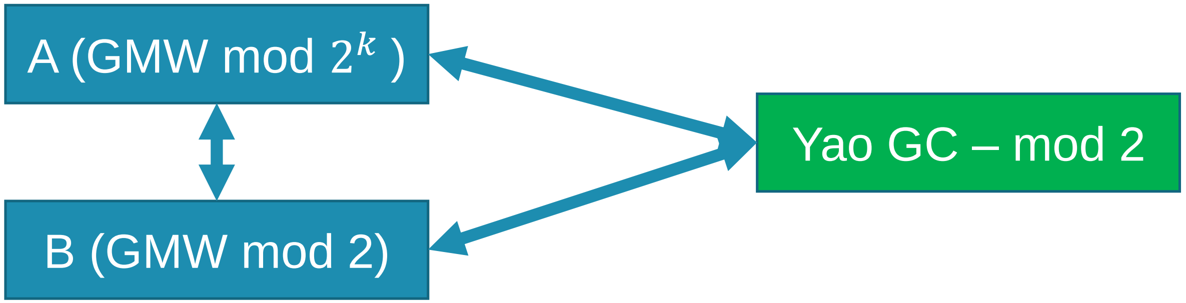
(a) Original Values

w_0	w_1	w_2	garbled value
k_0^0	k_1^0	k_2^0	$H(k_0^0 \| \| k_1^0 \| \| g_1) \oplus k_2^0$
k_0^0	k_1^1	k_2^1	$H(k_0^0 \| \| k_1^1 \| \| g_1) \oplus k_2^1$
k_0^1	k_1^0	k_2^1	$H(k_0^1 \| \| k_1^0 \| \| g_1) \oplus k_2^1$
k_0^1	k_1^1	k_2^1	$H(k_0^1 \| \| k_1^1 \| \| g_1) \oplus k_2^1$

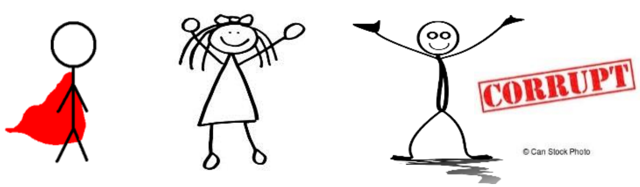
(b) Garbled Values

Figure 2: Computation table for g_1^{OR}



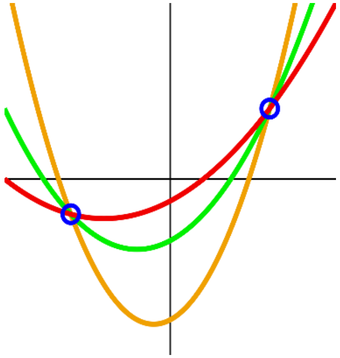


ABY [DMZ'15]



ABY3 [MR'18]

What about dishonest majority?



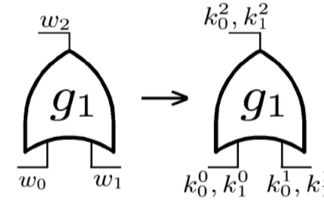


Figure 1: Garbling a single gate

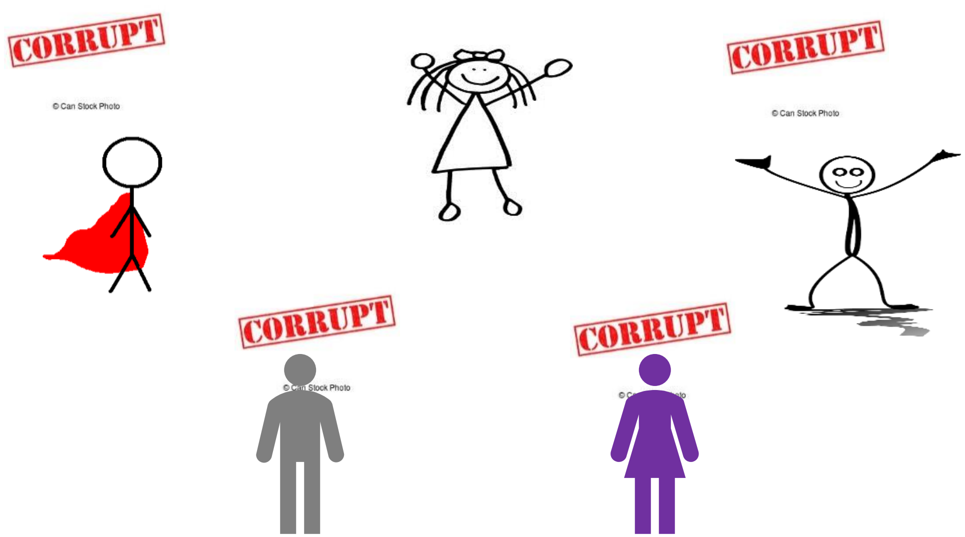
w_0	w_1	w_2
0	0	0
0	1	1
1	0	1
1	1	1

(a) Original Values

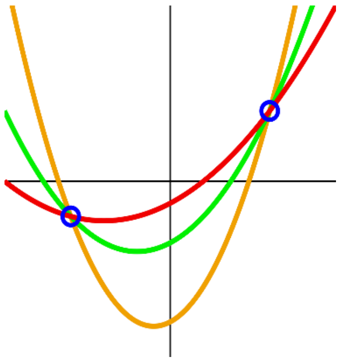
w_0	w_1	w_2	garbled value
k_0^0	k_1^0	k_2^0	$H(k_0^0\|\|k_1^0\|\|g_1) \oplus k_2^0$
k_0^0	k_1^1	k_2^1	$H(k_0^0\|\|k_1^1\|\|g_1) \oplus k_2^1$
k_0^1	k_1^0	k_2^1	$H(k_0^1\|\|k_1^0\|\|g_1) \oplus k_2^1$
k_0^1	k_1^1	k_2^1	$H(k_0^1\|\|k_1^1\|\|g_1) \oplus k_2^1$

(b) Garbled Values

Figure 2: Computation table for g_1^{OR}



What about dishonest majority?



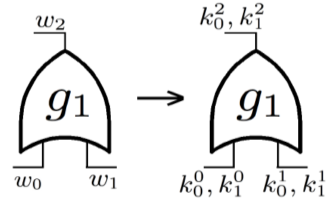


Figure 1: Garbling a single gate

w_0	w_1	w_2
0	0	0
0	1	1
1	0	1
1	1	1

(a) Original Values

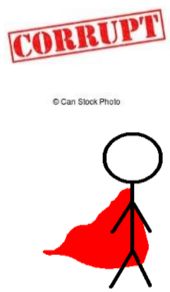
w_0	w_1	w_2	garbled value
k_0^0	k_1^0	k_2^0	$H(k_0^0 \| k_1^0 \| g_1) \oplus k_2^0$
k_0^0	k_1^1	k_2^1	$H(k_0^0 \| k_1^1 \| g_1) \oplus k_2^1$
k_0^1	k_1^0	k_2^1	$H(k_0^1 \| k_1^0 \| g_1) \oplus k_2^1$
k_0^1	k_1^1	k_2^1	$H(k_0^1 \| k_1^1 \| g_1) \oplus k_2^1$

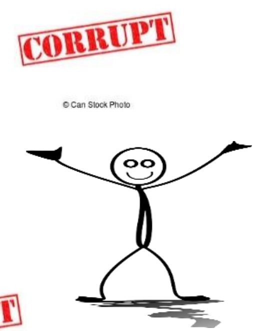
(b) Garbled Values

Figure 2: Computation table for g_1^{OR}

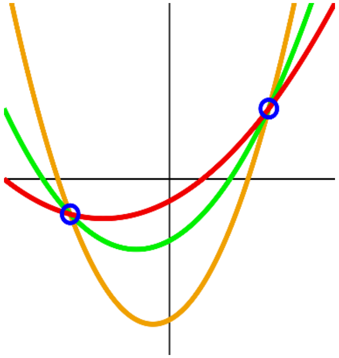
SPDZ

SPDZ-BMR





What about dishonest majority?



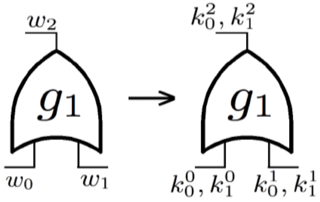


Figure 1: Garbling a single gate

w_0	w_1	w_2
0	0	0
0	1	1
1	0	1
1	1	1

(a) Original Values

w_0	w_1	w_2	garbled value
k_0^0	k_1^0	k_2^0	$H(k_0^0 \| k_1^0 \| g_1) \oplus k_2^0$
k_0^0	k_1^1	k_2^1	$H(k_0^0 \| k_1^1 \| g_1) \oplus k_2^1$
k_0^1	k_1^0	k_2^1	$H(k_0^1 \| k_1^0 \| g_1) \oplus k_2^1$
k_0^1	k_1^1	k_2^1	$H(k_0^1 \| k_1^1 \| g_1) \oplus k_2^1$

(b) Garbled Values

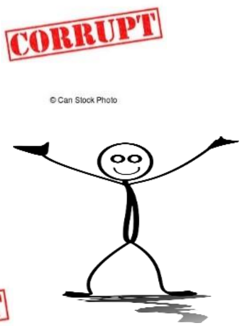
Figure 2: Computation table for g_1^{OR}

Naive

>110K ANDs

SPDZ

SPDZ-BMR

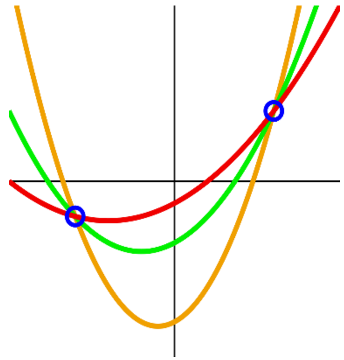


Dragoș Rotaru

imec-Cosic, Dept. Electrical Engineering

KU LEUVEN

What about dishonest majority?



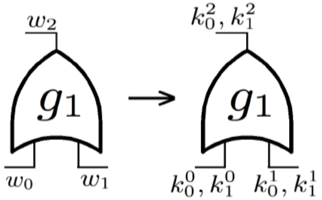


Figure 1: Garbling a single gate

w_0	w_1	w_2
0	0	0
0	1	1
1	0	1
1	1	1

(a) Original Values

w_0	w_1	w_2	garbled value
k_0^0	k_1^0	k_2^0	$H(k_0^0 \| k_1^0 \| g_1) \oplus k_2^0$
k_0^0	k_1^1	k_2^1	$H(k_0^0 \| k_1^1 \| g_1) \oplus k_2^1$
k_0^1	k_1^0	k_2^1	$H(k_0^1 \| k_1^0 \| g_1) \oplus k_2^1$
k_0^1	k_1^1	k_2^1	$H(k_0^1 \| k_1^1 \| g_1) \oplus k_2^1$

(b) Garbled Values

Figure 2: Computation table for g_1^{OR}

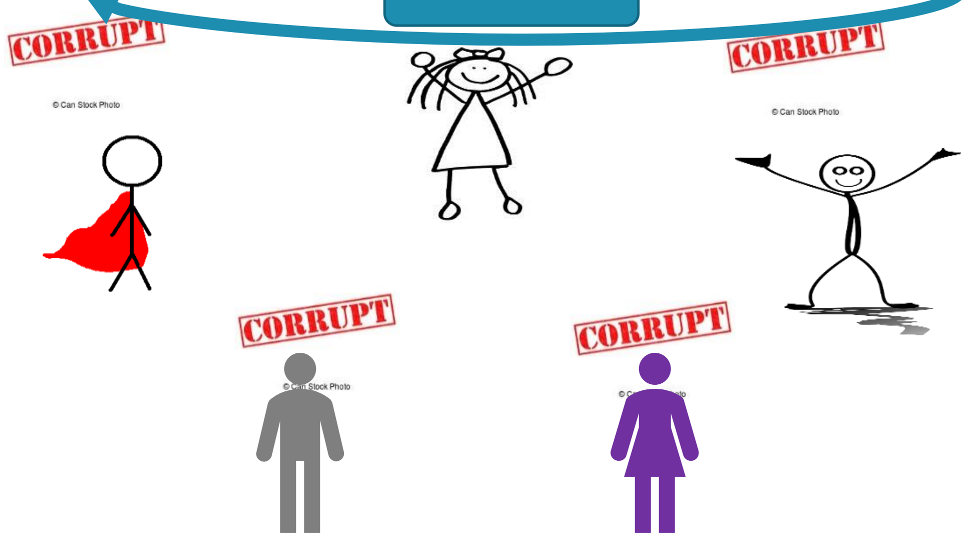
Naive

SPDZ

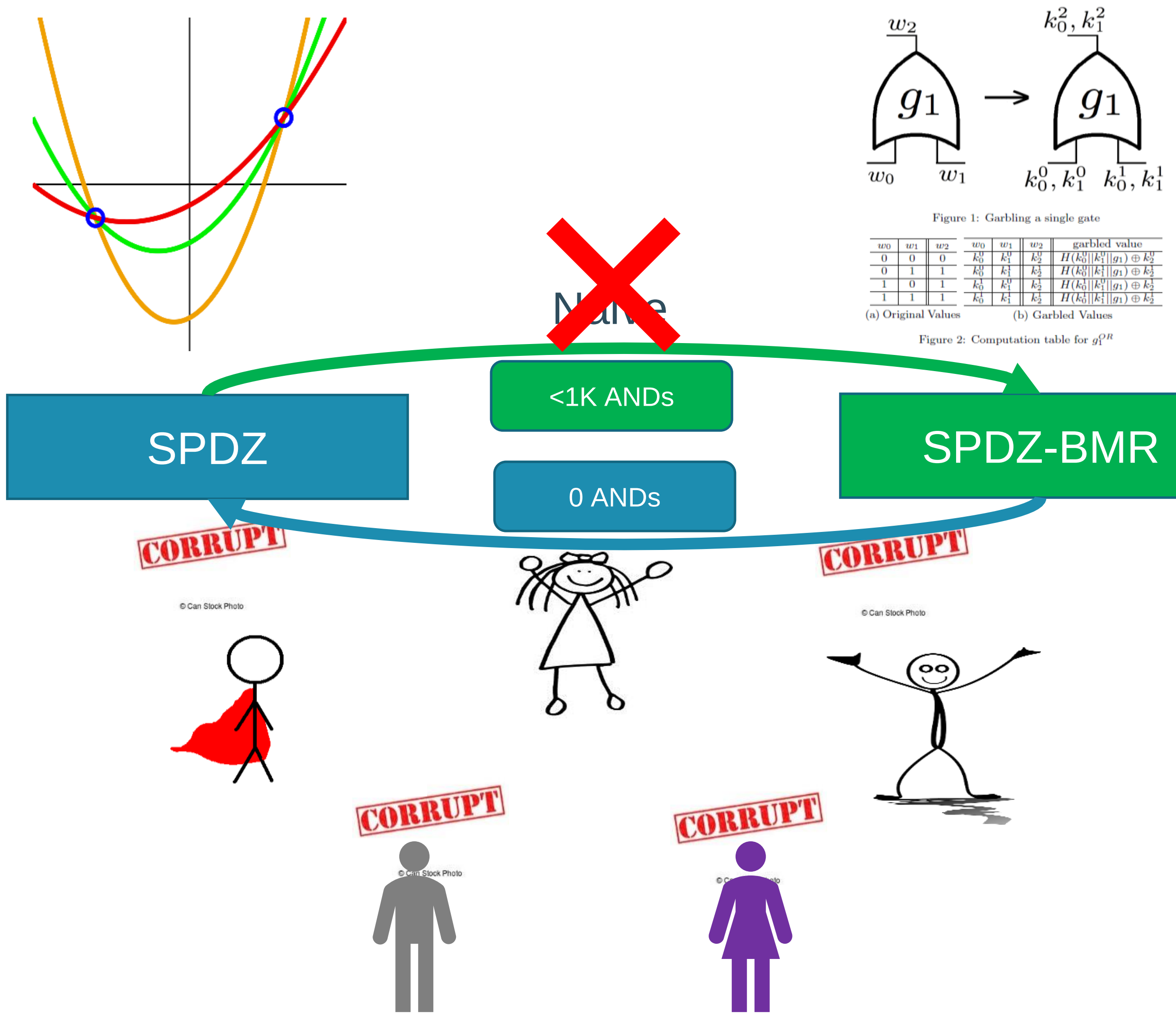
>110K ANDs

>110K ANDs

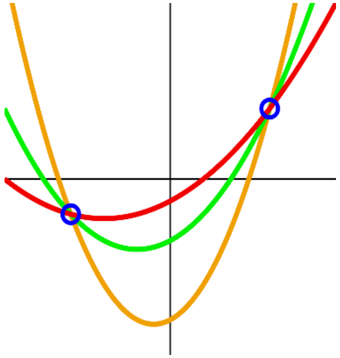
SPDZ-BMR



What about dishonest majority?



How general is this?



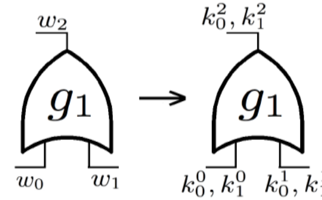


Figure 1: Garbling a single gate

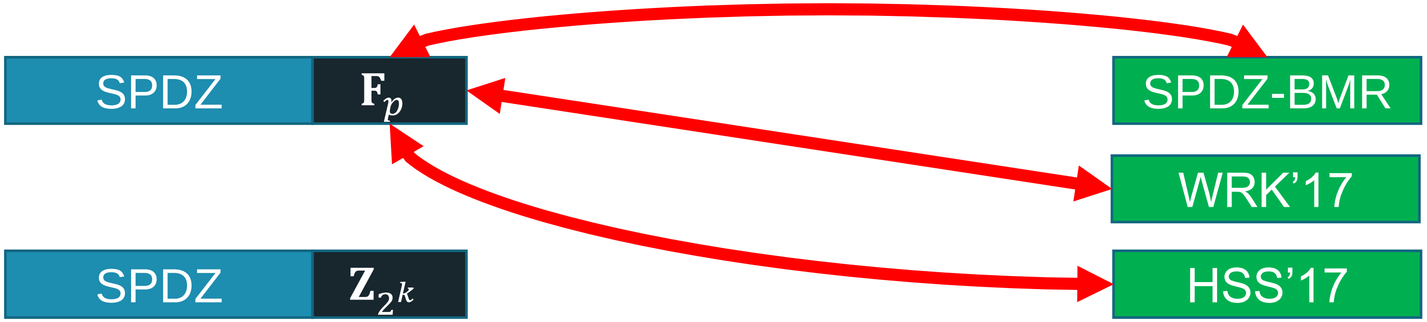
w_0	w_1	w_2
0	0	0
0	1	1
1	0	1
1	1	1

(a) Original Values

w_0	w_1	w_2	garbled value
k_0^0	k_1^0	k_2^0	$H(k_0^0 \| k_1^0 \| g_1) \oplus k_2^0$
k_0^0	k_1^1	k_2^1	$H(k_0^0 \| k_1^1 \| g_1) \oplus k_2^1$
k_0^1	k_1^0	k_2^1	$H(k_0^1 \| k_1^0 \| g_1) \oplus k_2^1$
k_0^1	k_1^1	k_2^1	$H(k_0^1 \| k_1^1 \| g_1) \oplus k_2^1$

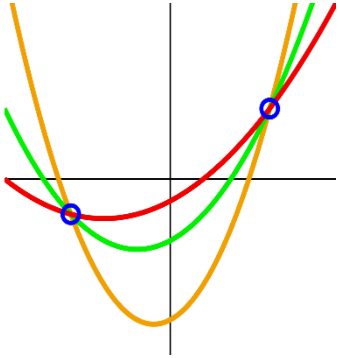
(b) Garbled Values

Figure 2: Computation table for g_1^{OR}



Dragoș Rotaru
imec-Cosic, Dept. Electrical Engineering

How general is this?



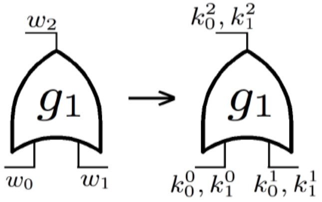


Figure 1: Garbling a single gate

w_0	w_1	w_2
0	0	0
0	1	1
1	0	1
1	1	1

(a) Original Values

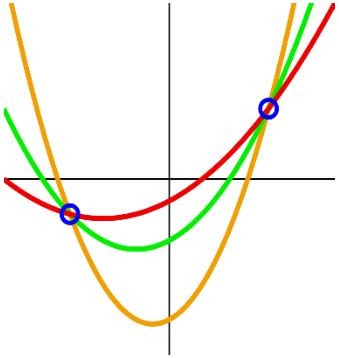
w_0	w_1	w_2	garbled value
k_0^0	k_1^0	k_2^0	$H(k_0^0 \| k_1^0 \| g_1) \oplus k_2^0$
k_0^0	k_1^1	k_2^1	$H(k_0^0 \| k_1^1 \| g_1) \oplus k_2^1$
k_0^1	k_1^0	k_2^1	$H(k_0^1 \| k_1^0 \| g_1) \oplus k_2^1$
k_0^1	k_1^1	k_2^1	$H(k_0^1 \| k_1^1 \| g_1) \oplus k_2^1$

(b) Garbled Values

Figure 2: Computation table for g_1^{OR}

Dragoș Rotaru
imec-Cosic, Dept. Electrical Engineering
KU LEUVEN

How general is this?



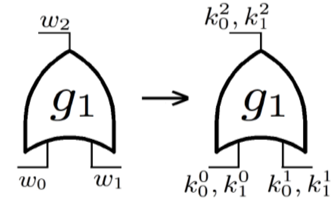


Figure 1: Garbling a single gate

w_0	w_1	w_2
0	0	0
0	1	1
1	0	1
1	1	1

(a) Original Values

w_0	w_1	w_2	garbled value
k_0^0	k_1^0	k_2^0	$H(k_0^0 \| \| k_1^0 \| \| g_1) \oplus k_2^0$
k_0^0	k_1^1	k_2^1	$H(k_0^0 \| \| k_1^1 \| \| g_1) \oplus k_2^1$
k_0^1	k_1^0	k_2^1	$H(k_0^1 \| \| k_1^0 \| \| g_1) \oplus k_2^1$
k_0^1	k_1^1	k_2^1	$H(k_0^1 \| \| k_1^1 \| \| g_1) \oplus k_2^1$

(b) Garbled Values

Figure 2: Computation table for g_1^{OR}

SPDZ | $\mathbf{F}_p$

↕

SPDZ | $\mathbf{Z}_{2^k}$

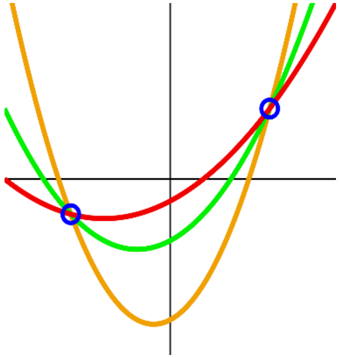
SPDZ-BMR

WRK'17

HSS'17

Dragoș Rotaru
imec-Cosic, Dept. Electrical Engineering
KU LEUVEN

How general is this?



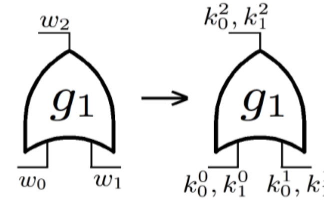


Figure 1: Garbling a single gate

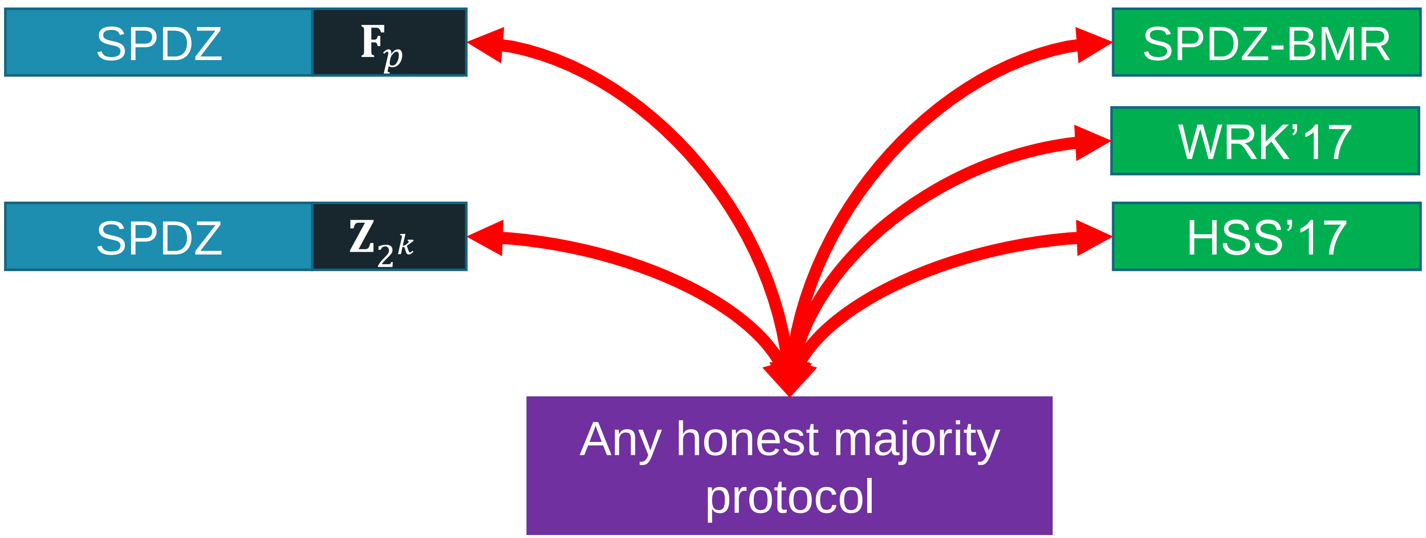
w_0	w_1	w_2
0	0	0
0	1	1
1	0	1
1	1	1

(a) Original Values

w_0	w_1	w_2	garbled value
k_0^0	k_1^0	k_2^0	$H(k_0^0 \| k_1^0 \| g_1) \oplus k_2^0$
k_0^0	k_1^1	k_2^1	$H(k_0^0 \| k_1^1 \| g_1) \oplus k_2^1$
k_0^1	k_1^0	k_2^1	$H(k_0^1 \| k_1^0 \| g_1) \oplus k_2^1$
k_0^1	k_1^1	k_2^1	$H(k_0^1 \| k_1^1 \| g_1) \oplus k_2^1$

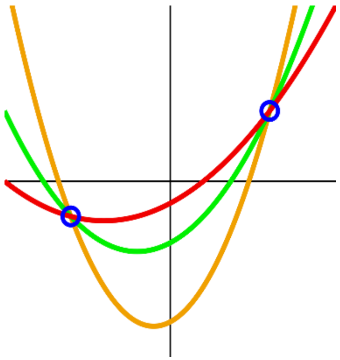
(b) Garbled Values

Figure 2: Computation table for g_1^{OR}



Dragoș Rotaru
imec-Cosic, Dept. Electrical Engineering
KU LEUVEN

Our focus



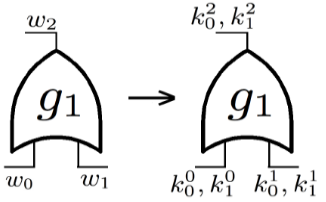


Figure 1: Garbling a single gate

w_0	w_1	w_2
0	0	0
0	1	1
1	0	1
1	1	1

(a) Original Values

w_0	w_1	w_2	garbled value
k_0^0	k_1^0	k_2^0	$H(k_0^0 \| k_1^0 \| g_1) \oplus k_2^0$
k_0^0	k_1^1	k_2^1	$H(k_0^0 \| k_1^1 \| g_1) \oplus k_2^1$
k_0^1	k_1^0	k_2^1	$H(k_0^1 \| k_1^0 \| g_1) \oplus k_2^1$
k_0^1	k_1^1	k_2^1	$H(k_0^1 \| k_1^1 \| g_1) \oplus k_2^1$

(b) Garbled Values

Figure 2: Computation table for g_1^{OR}

SPDZ $\mathbf{F}_p$

SPDZ-BMR

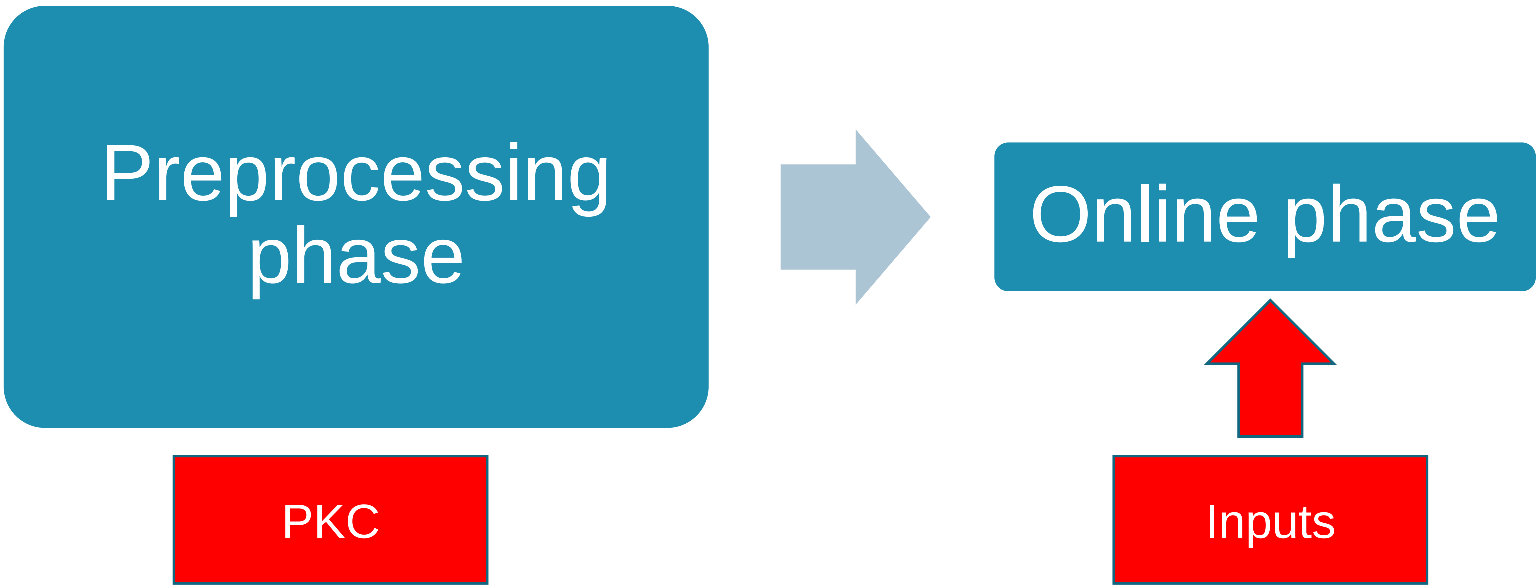
WRK'17

SPDZ $\mathbf{Z}_{2^k}$

HSS'17

Dragoș Rotaru
imec-Cosic, Dept. Electrical Engineering
KU LEUVEN

Malicious MPC protocols

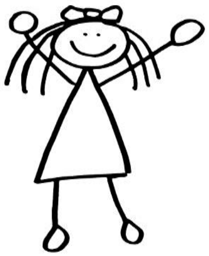


SPDZ, TinyOT, BDOZa, MASCOT, WRK'17, HSS'17, …

Let's talk about

SPDZ	$\mathbf{F}_p$

SPDZ $\mathbf{F}_p$ online phase

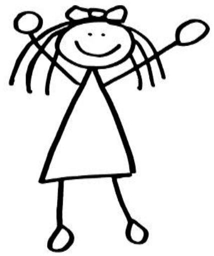


$$\alpha_1 + \alpha_2 + \alpha_3 = \alpha$$

$$x_1 + x_2 + x_3 = x$$

$$\gamma(x)_1 + \gamma(x)_2 + \gamma(x)_3 = \alpha x$$

SPDZ $\mathbf{F}_p$ online phase

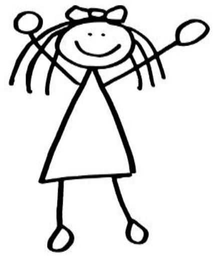
  

$$\alpha_1 + \alpha_2 + \alpha_3 = \alpha$$

$$x_1 + y_1 + x_2 + y_2 + x_3 + y_3 = x + y$$

$$\gamma(\mathrm{x})_1 + \gamma(\mathrm{y})_1 + \gamma(\mathrm{x})_2 + \gamma(\mathrm{y})_2 + \gamma(\mathrm{x})_3 + \gamma(\mathrm{y})_3 = \alpha(x + y)$$

SPDZ $\mathbf{F}_p$ online phase



Input

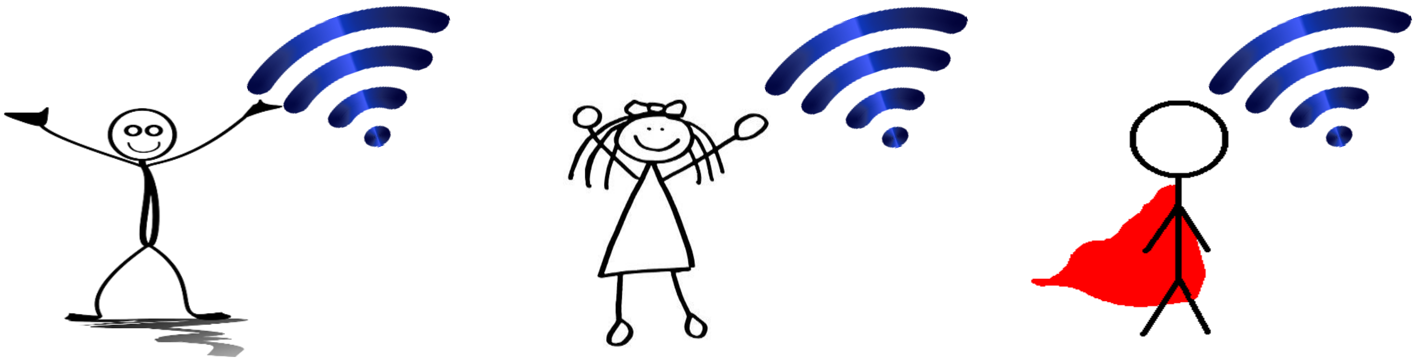
Retrieve a random mask

$X_A \leftarrow X_A$

SPDZ $\mathbf{F}_p$ online phase

Input $\boxed{X_A} \longleftarrow X_A$

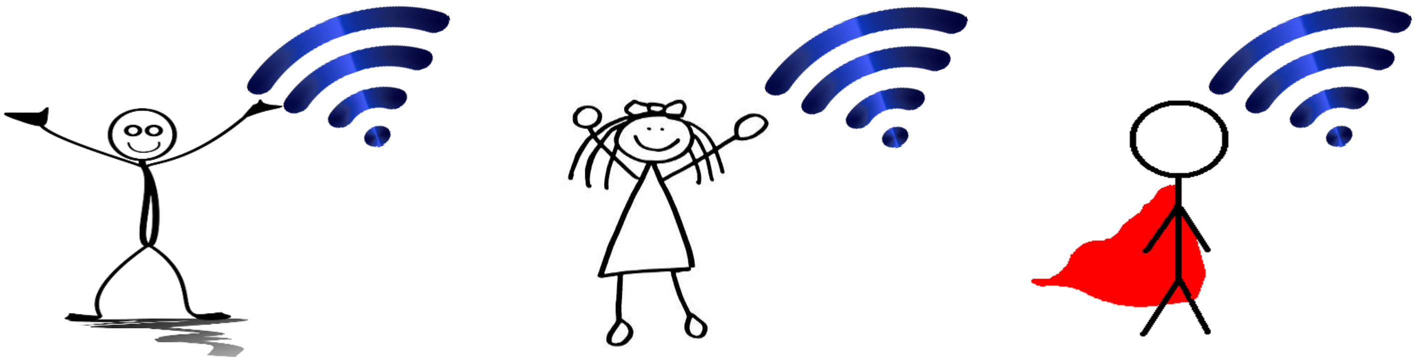
SPDZ $\mathbf{F}_p$ online phase



Input X_A $\leftarrow$ X_A

Open X $\leftarrow$ X

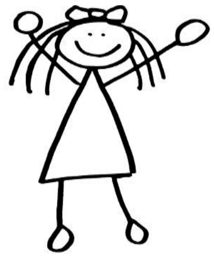
SPDZ $\mathbf{F}_p$ online phase



Input X_A ← X_A

Open MAC Check X ← X

SPDZ $\mathbb{F}_p$ online phase

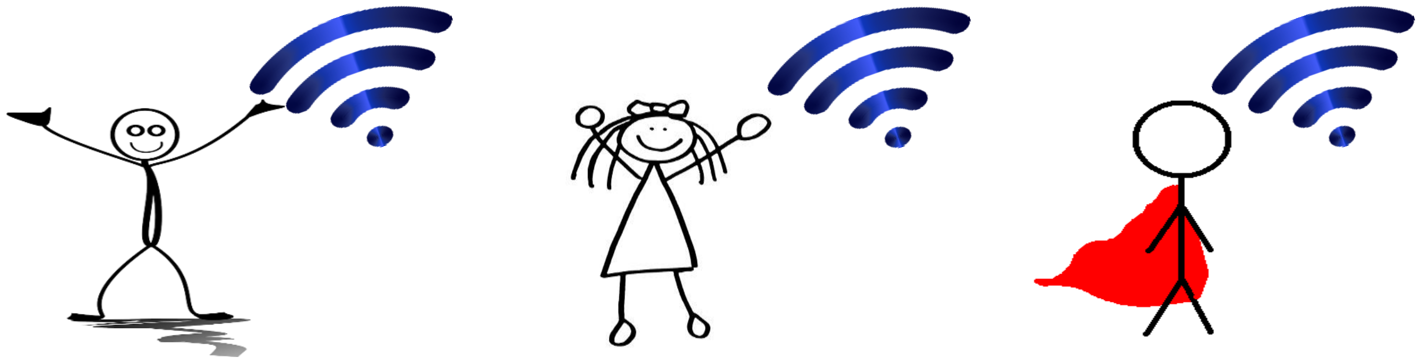


Input $\quad X_A \leftarrow X_A$

Open $\quad x \leftarrow x$

XOR $\quad$ Retrieve a Beaver triple $\quad z \leftarrow x \oplus y$

SPDZ $\mathbf{F}_p$ online phase



Input		$x_A \leftarrow x_A$
Open	MAC Check	$x \leftarrow x$
XOR		$z \leftarrow x \oplus y$

Dragoş Rotaru

imec-Cosic, Dept. Electrical Engineering

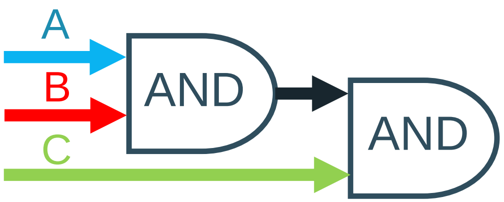
KU LEUVEN

Let’s talk about

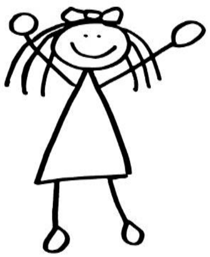
SPDZ-BMR	$\mathbf{F}_2$

Dragoș Rotaru
imec-Cosic, Dept. Electrical Engineering

SPDZ-BMR $\mathbf{F}_2$ online phase



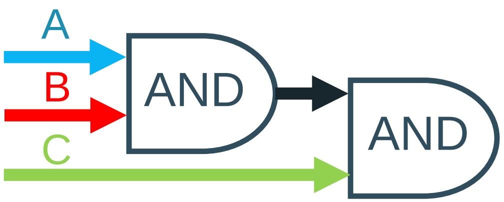
C

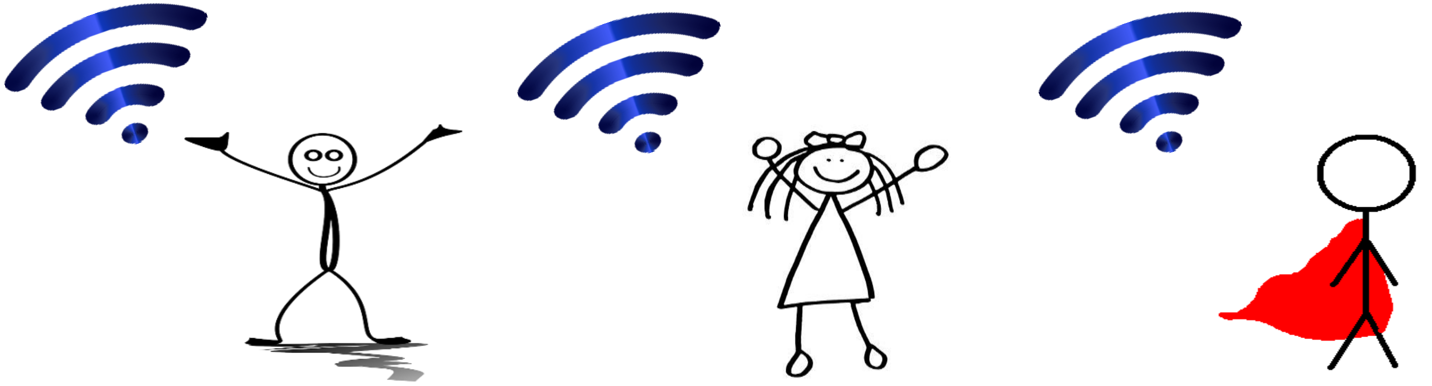


A

B

SPDZ-BMR $\mathbb{F}_2$ online phase

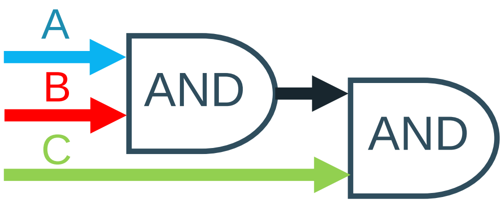


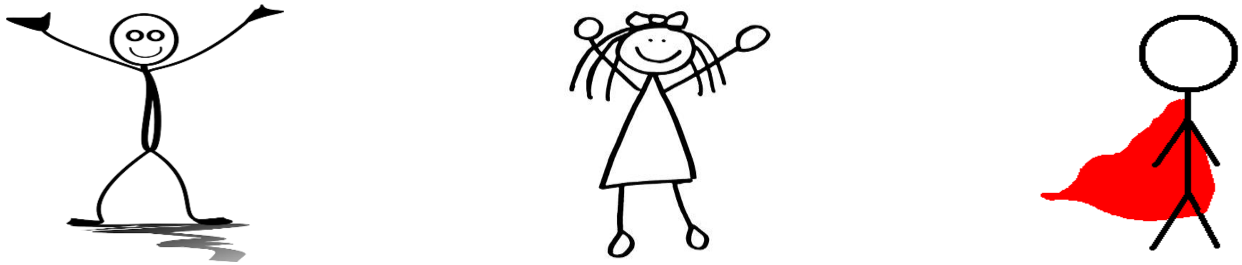


MAC Check

$\Lambda_c \leftarrow C + \lambda_c$ $\Lambda_A \leftarrow A + \lambda_a$ $\Lambda_B \leftarrow B + \lambda_b$

SPDZ-BMR $\mathbb{F}_2$ online phase





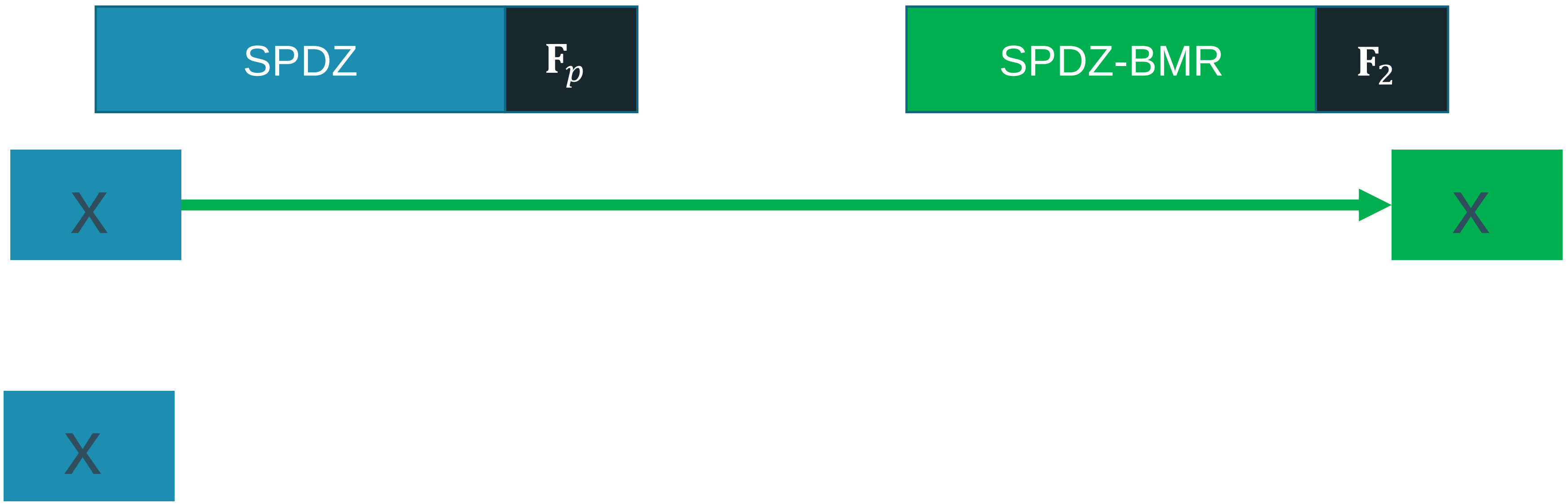
$\Lambda_c \leftarrow C + \lambda_c$ $\Lambda_A \leftarrow A + \lambda_a$ $\Lambda_B \leftarrow B + \lambda_b$

Inputs - cheap

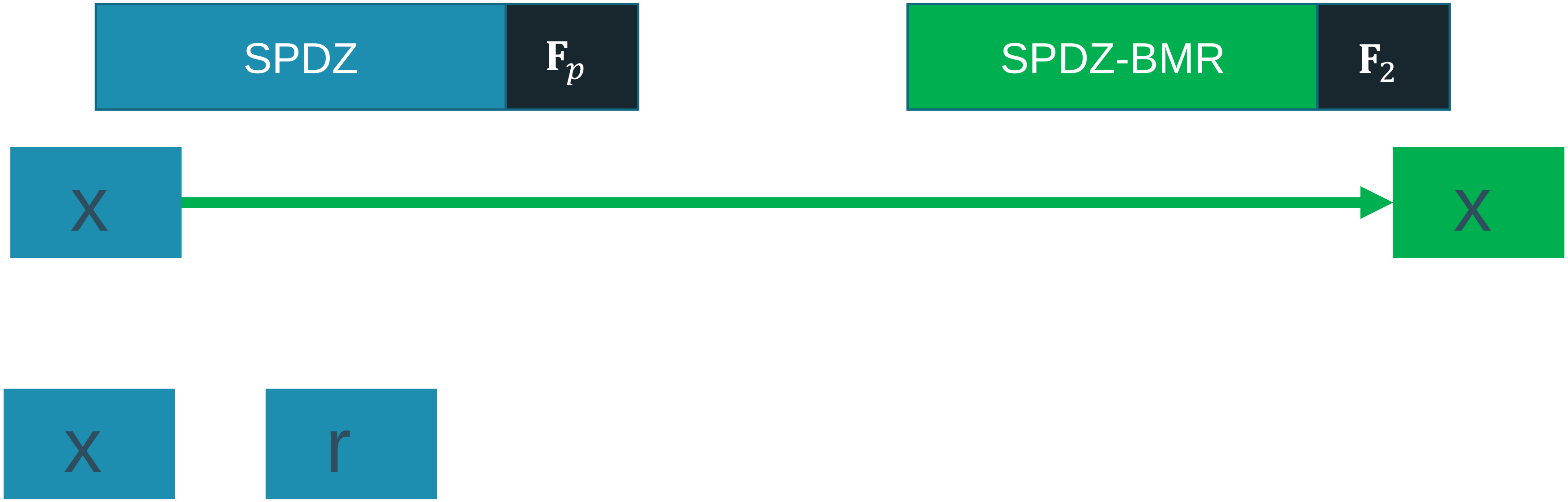
XOR - free

Mod p arithmetic - some AND gates

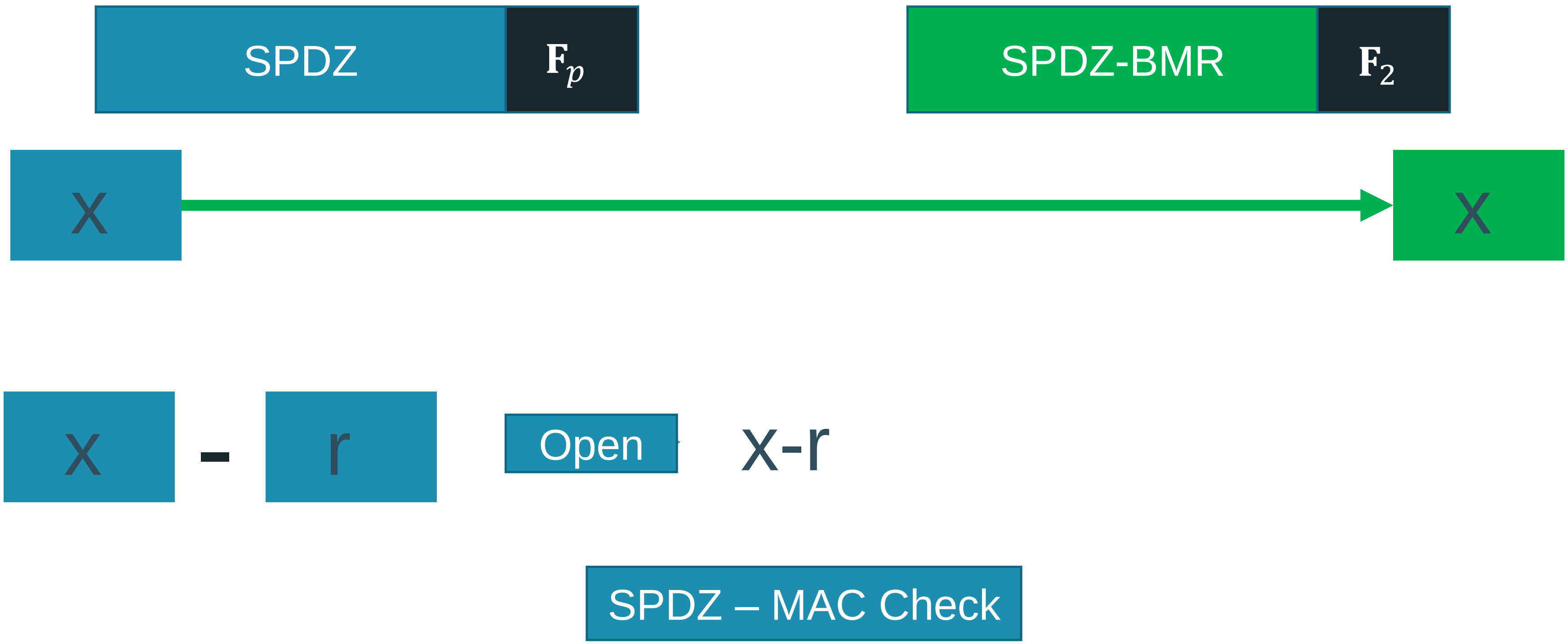
Main idea:



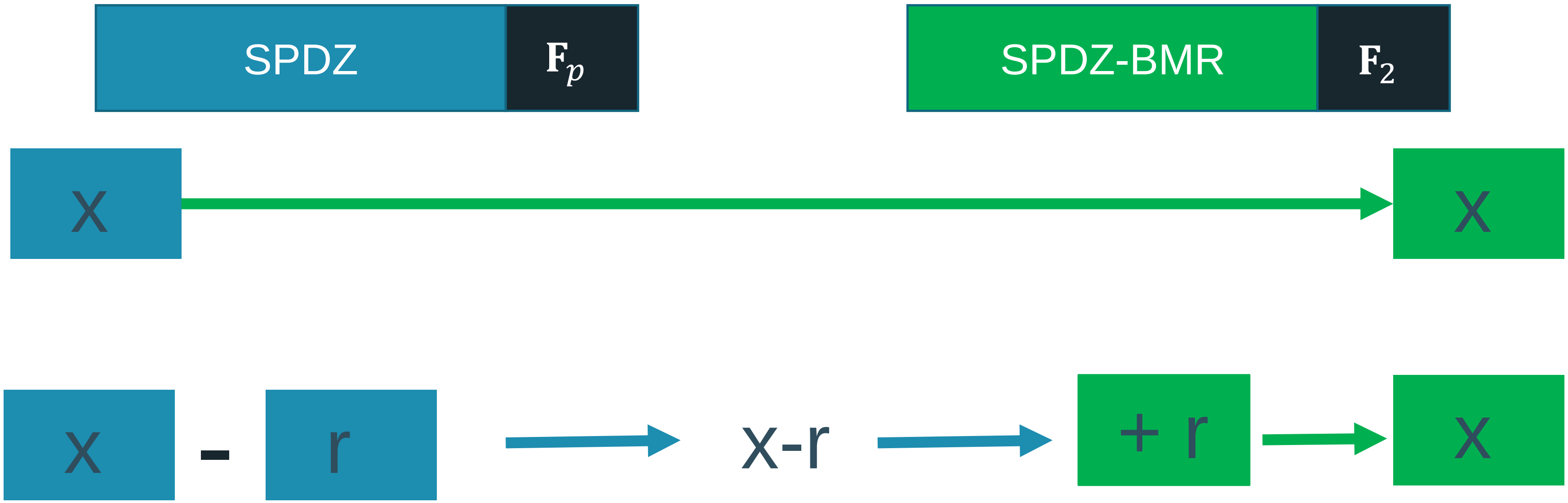
Main idea:



Main idea:



Main idea:



Introducing daBits

Introducing daBits

SPDZ | $\mathbf{F}_p$

SPDZ-BMR | $\mathbf{F}_2$

b_A

b_B

b_C

Introducing daBits

SPDZ $\mathbf{F}_p$

SPDZ-BMR $\mathbf{F}_2$

SPDZ Input

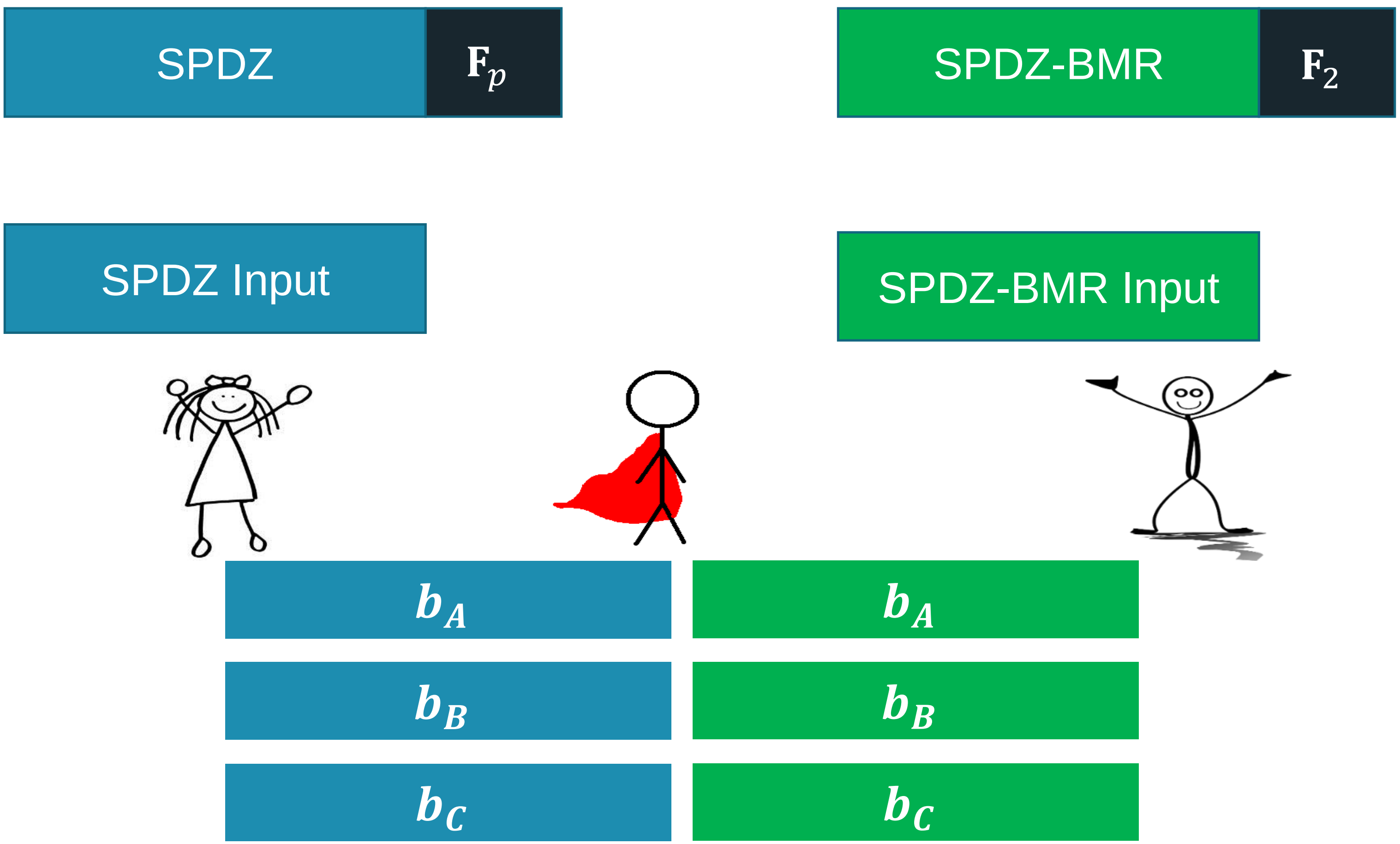
SPDZ-BMR Input

b_A

b_B

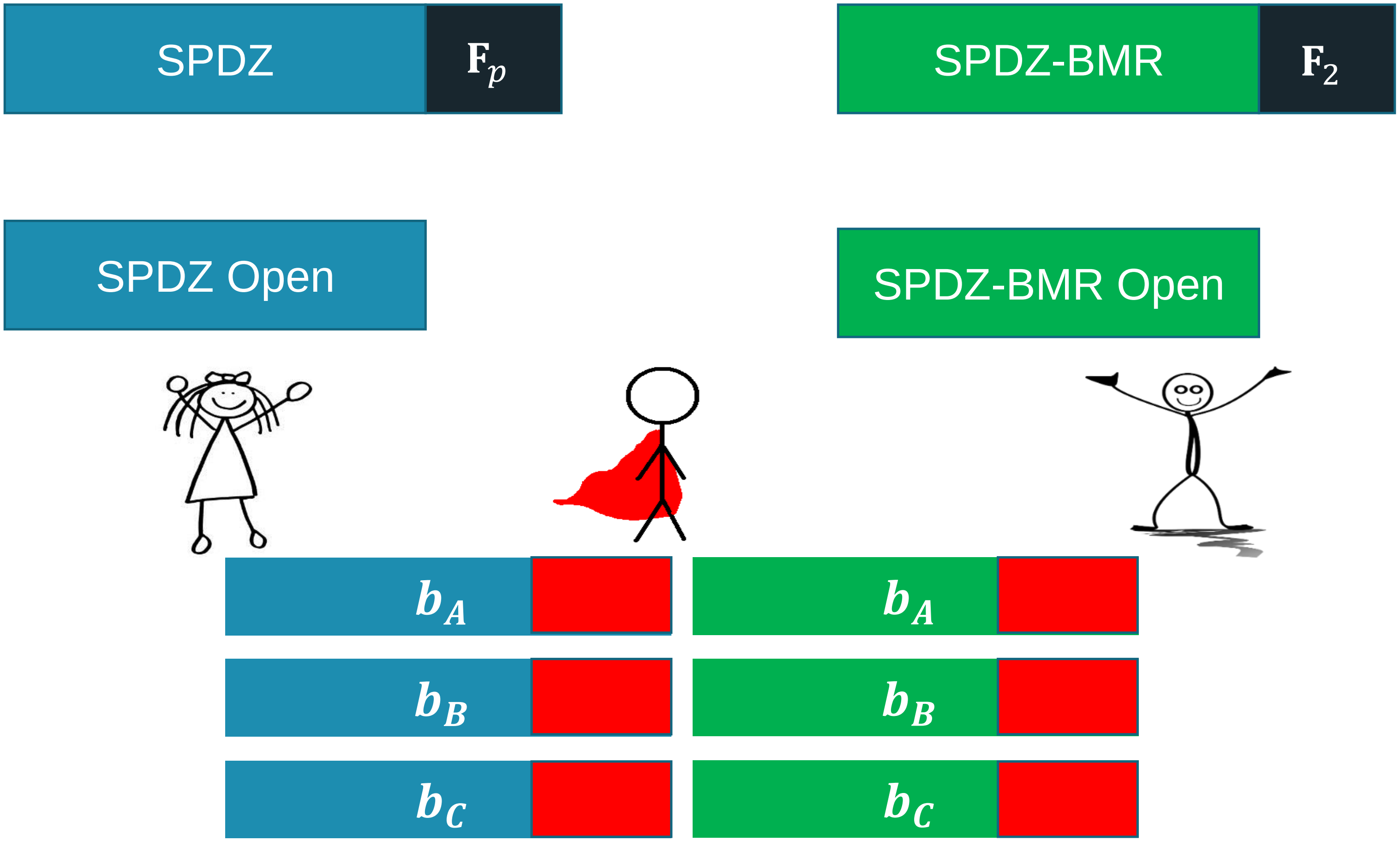
b_C

Introducing daBits



Dragoș Rotaru
imec-Cosic, Dept. Electrical Engineering
KU LEUVEN

Introducing daBits



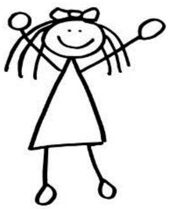
Introducing daBits

SPDZ $\mathbf{F}_p$

SPDZ-BMR $\mathbf{F}_2$

SPDZ XOR

SPDZ-BMR XOR



$b_A \oplus b_B \oplus b_C$

$b_A \oplus b_B \oplus b_C$

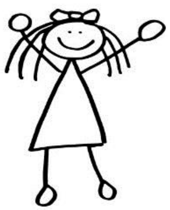
Introducing daBits

SPDZ $\mathbf{F}_p$

SPDZ-BMR $\mathbf{F}_2$

SPDZ Open

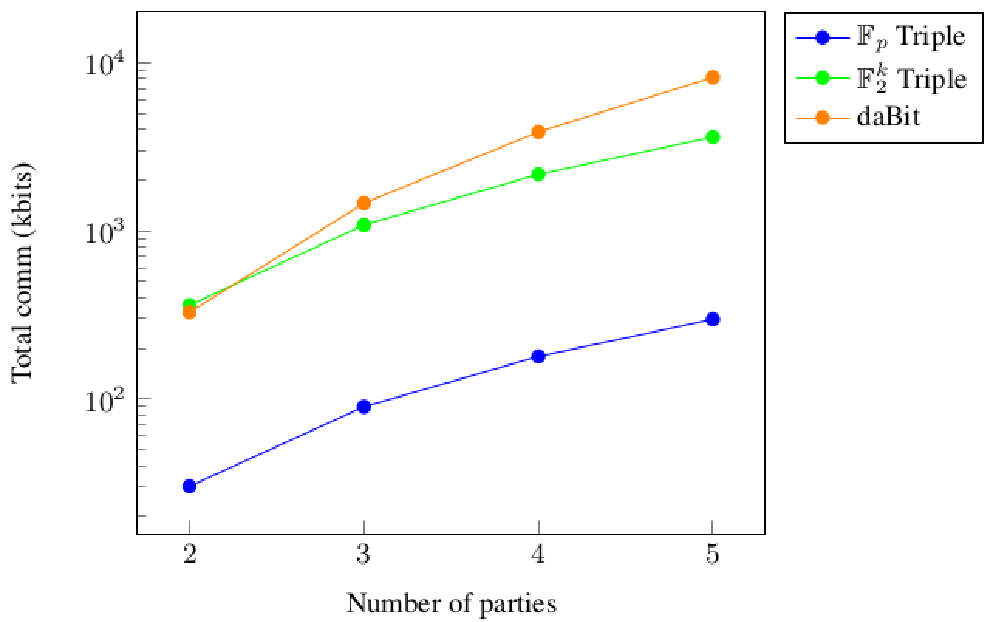
SPDZ-BMR Open



$b_A \oplus b_B \oplus b_C$

$b_A \oplus b_B \oplus b_C$

daBit cost



SPDZ-BMR

SPDZ

Total communication costs for all parties per preprocessed element.

Preprocessing cost per conversion

sec	$\log p$	k	Comm. (kb)			Total (kb)	Time (ms)			Total(ms)
			$\mathcal{F}^{p}_{\mathsf{MPC}}$	$\mathcal{F}^{2^k}_{\mathsf{MPC}}$	daBitgen		$\mathcal{F}^{p}_{\mathsf{MPC}}$	$\mathcal{F}^{2^k}_{\mathsf{MPC}}$	daBitgen	
40	128	128	76.60	2.30	6.94	85.84	0.159	$<$ 10ns	0.004	0.163

Table 2. 1Gb/s LAN experiments for two-party daBit generation per party. For all cases, the daBit batch has length 8192.

Example code in MP-SPDZ

```
bit_len = 7
x = sint(42) # mod p share
xb = sbits.switch_to_gc(bit_len, x) # mod 2 shares
bits = xb.bit_decompose(bit_len)

for i in range(len(bits)):
    print_str('%s', bits[i].reveal())
# prints 0101010
```

Online cost per conversion

Conversion	SPDZ-BMR	
	ANDs	Online (ms)
sint $\mapsto$ sbits	379	0.106
sbits $\mapsto$ sint	0	0.005

8X overhead than using ABY

Online cost per conversion

Conversion	SPDZ-BMR	
	ANDs	Online (ms)
sint $\mapsto$ sbits	379	0.106
sbits $\mapsto$ sint	0	0.005

8X overhead than using ABY

What's next?

- SCALE-MAMBA has WRK'17.
- It also has all preprocessing phases connected – ideal candidate for daBits in a more realistic system.
- Moral: Stitch your work together so it would be easier to build more efficient protocols on top of them.

Dragoș Rotaru
imec-Cosic, Dept. Electrical Engineering
KU LEUVEN

Thank you!

Thank you!

- Questions?
- **https://ia.cr/2019/207**