TPMPC 2019

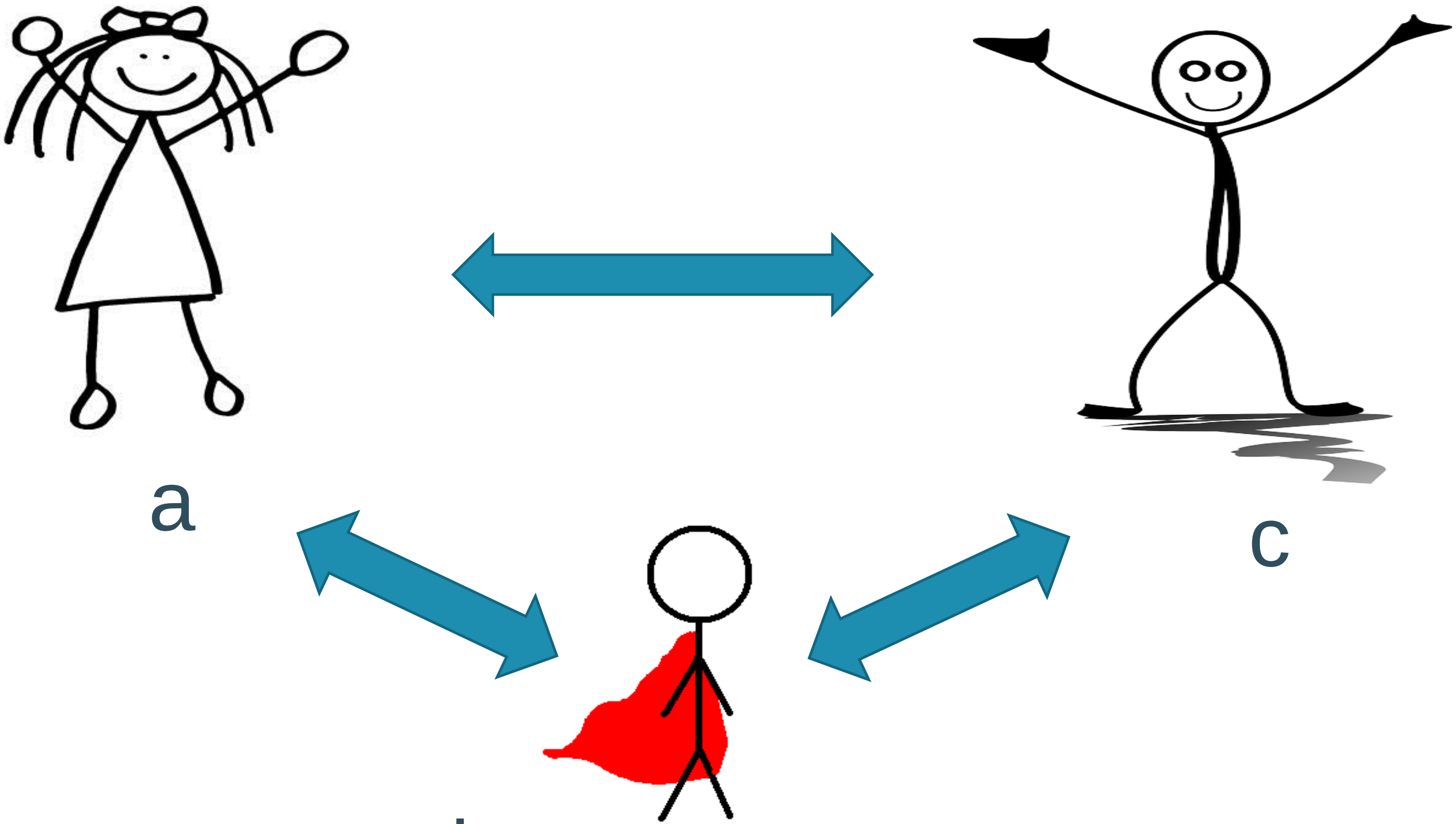
MArBled Circuits: Mixing Arithmetic and Boolean Circuits with Active Security*

Dragoș Rotaru and **Tim Wood**

University of Bristol, KU Leuven

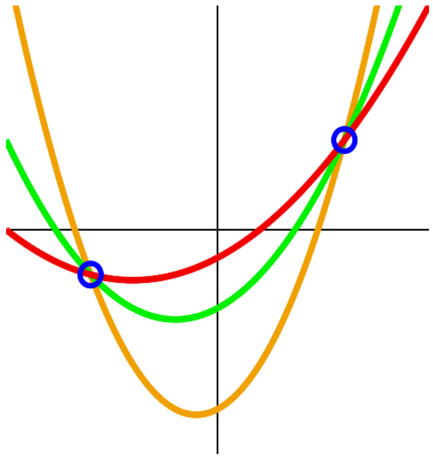
* ***https://ia.cr/2019/207***

What is multiparty computation?



Goal: Compute F(a, b, c)

How can we achieve MPC?



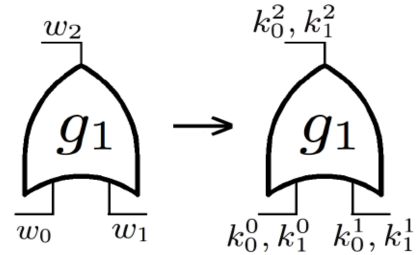


Figure 1: Garbling a single gate

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	$H(k_0^1 \| k_1^1 \| g_1) \oplus k_2^1$

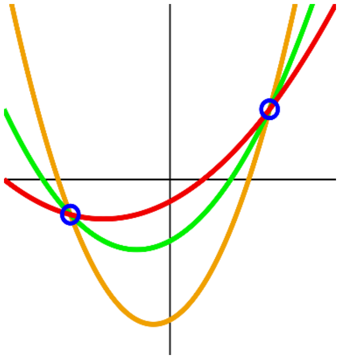
(b) Garbled Values

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?



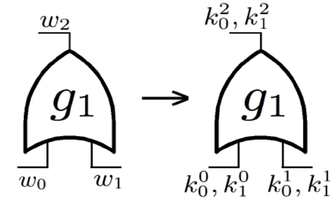


Figure 1: Garbling a single gate

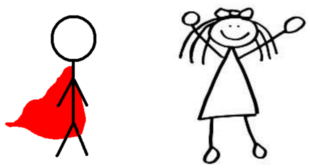
w_0	w_1	w_2
0	0	0
0	1	1
1	0	1
1	1	1

(a) Original Values

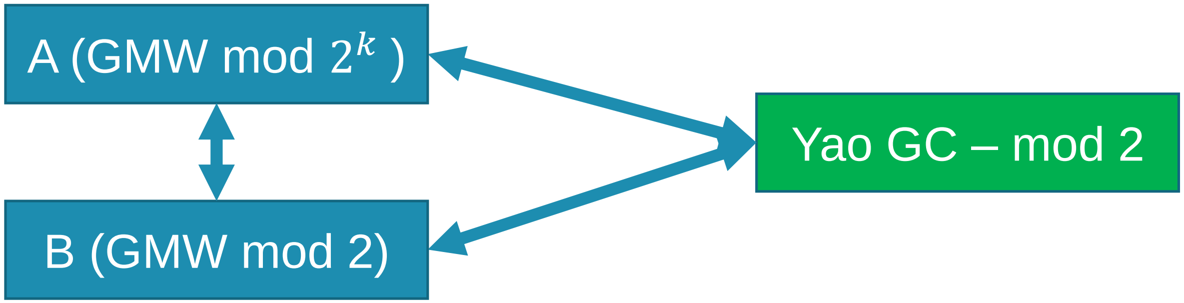
w_0	w_1	w_2	garbled value
k_0^0	k_1^0	k_2^0	$H(k_0^0 \| \| k_1^0 \| \| g_1) \oplus k_2^0$
k_0^0	k_1^1	k_2^1	$H(k_0^0 \| \| k_1^1 \| \| g_1) \oplus k_2^1$
k_0^1	k_1^0	k_2^1	$H(k_0^1 \| \| k_1^0 \| \| g_1) \oplus k_2^1$
k_0^1	k_1^1	k_2^1	$H(k_0^1 \| \| k_1^1 \| \| g_1) \oplus k_2^1$

(b) Garbled Values

Figure 2: Computation table for g_1^{OR}

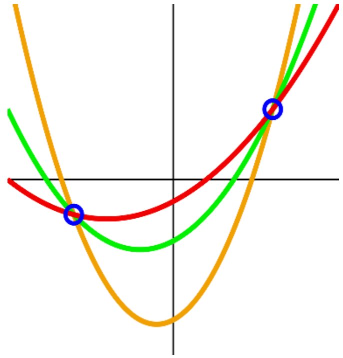


ABY [DSZ'15]



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

Can we switch between?



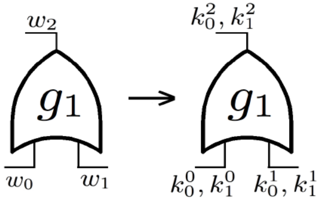


Figure 1: Garbling a single gate

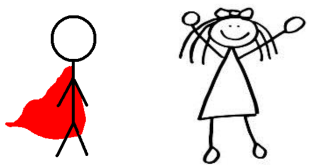
w_0	w_1	w_2
0	0	0
0	1	1
1	0	1
1	1	1

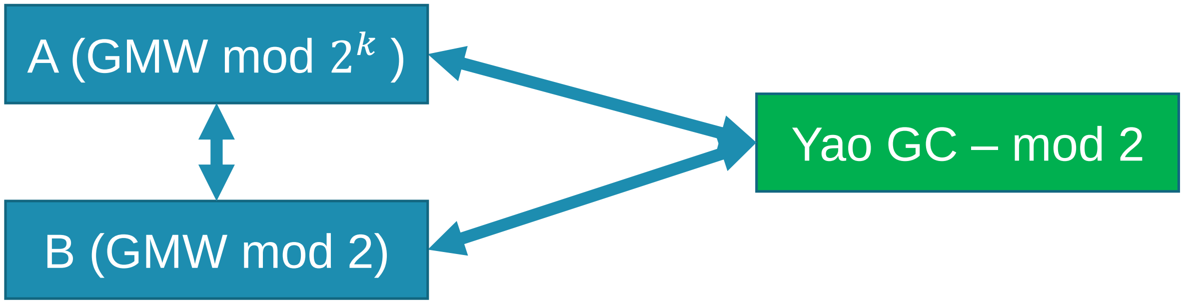
(a) Original Values

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	$H(k_0^1 \| k_1^1 \| g_1) \oplus k_2^1$

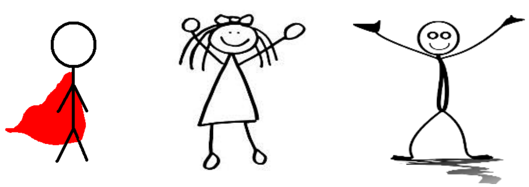
(b) Garbled Values

Figure 2: Computation table for g_1^{OR}



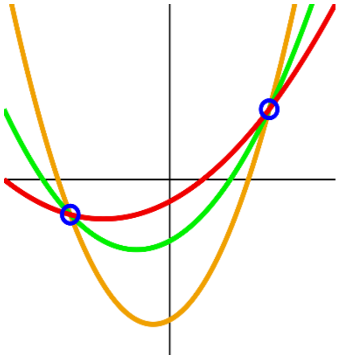


ABY [DSZ'15]



ABY3 [MR'18]

Can we switch between?



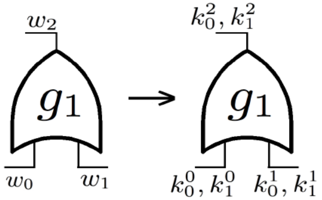


Figure 1: Garbling a single gate

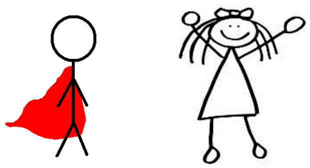
w_0	w_1	w_2
0	0	0
0	1	1
1	0	1
1	1	1

(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}

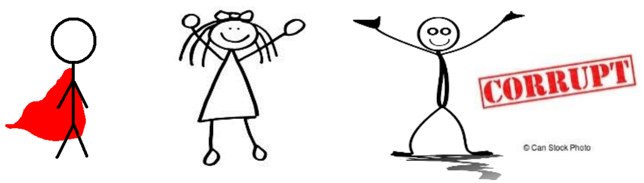


A (GMW mod 2^k)

B (GMW mod 2)

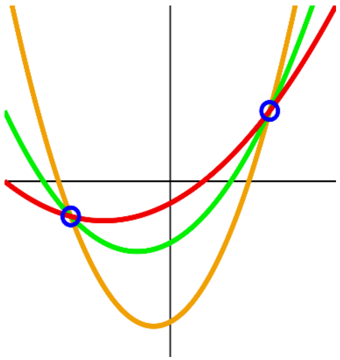
Yao GC – mod 2

ABY [DSZ'15]



ABY3 [MR'18]

What about dishonest majority?



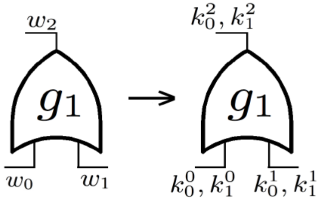


Figure 1: Garbling a single gate

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

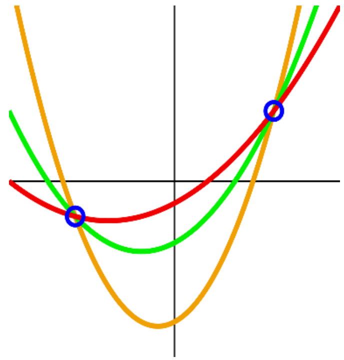
(a) Original Values

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?



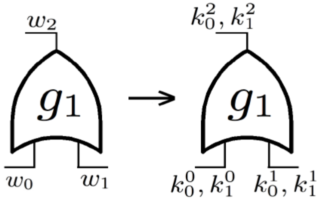


Figure 1: Garbling a single gate

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

(a) Original Values

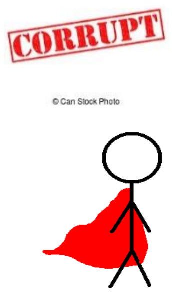
w_0	w_1	w_2	garbled value
k_0^0	k_1^0	k_2^0	$H(k_0^0 \| k_1^0 \| g_1) \oplus k_2^0$
k_0^0	k_1^1	k_2^1	$H(k_0^0 \| k_1^1 \| g_1) \oplus k_2^1$
k_0^1	k_1^0	k_2^1	$H(k_0^1 \| k_1^0 \| g_1) \oplus k_2^1$
k_0^1	k_1^1	k_2^1	$H(k_0^1 \| k_1^1 \| g_1) \oplus k_2^1$

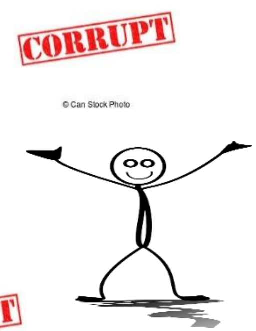
(b) Garbled Values

Figure 2: Computation table for g_1^{OR}

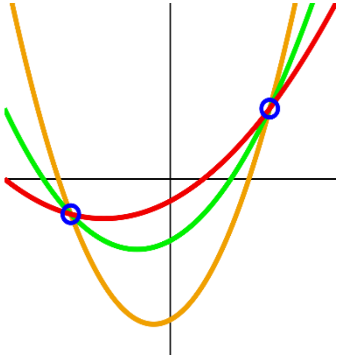
SPDZ

WRK'17





What about dishonest majority?



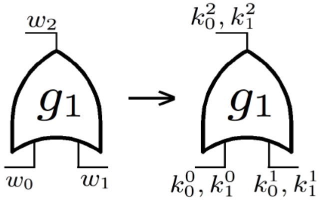


Figure 1: Garbling a single gate

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}

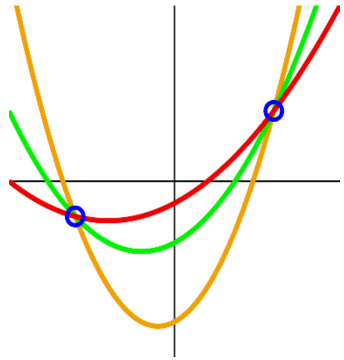
Naive

>110K ANDs

SPDZ

WRK'17

What about dishonest majority?



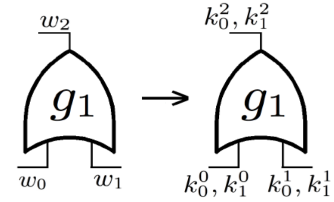


Figure 1: Garbling a single gate

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}

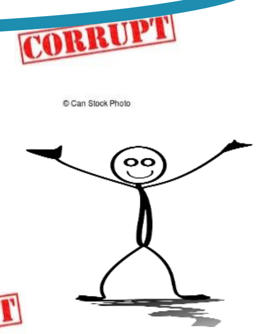
Naive

SPDZ

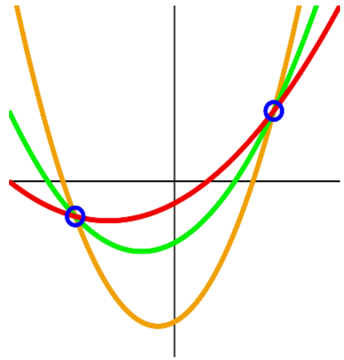
>110K ANDs

WRK'17

>110K ANDs



What about dishonest majority?



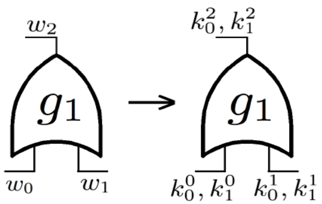


Figure 1: Garbling a single gate

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}

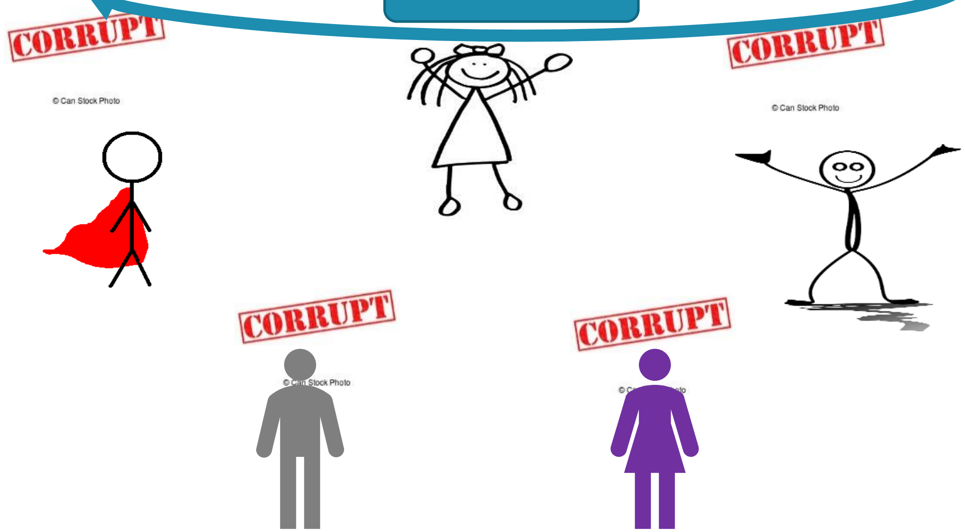
~~Naive~~

<1K ANDs

SPDZ

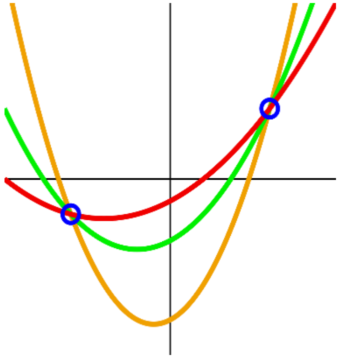
WRK’17

0 ANDs



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

How general is this?



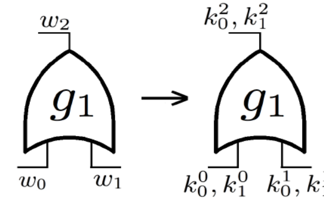


Figure 1: Garbling a single gate

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

(a) Original Values

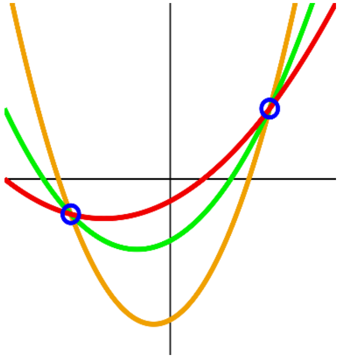
w_0	w_1	w_2	garbled value
k_0^0	k_1^0	k_2^0	$H(k_0^0 \| k_1^0 \| g_1) \oplus k_2^0$
k_0^0	k_1^1	k_2^1	$H(k_0^0 \| k_1^1 \| g_1) \oplus k_2^1$
k_0^1	k_1^0	k_2^1	$H(k_0^1 \| k_1^0 \| g_1) \oplus k_2^1$
k_0^1	k_1^1	k_2^1	$H(k_0^1 \| k_1^1 \| g_1) \oplus k_2^1$

(b) Garbled Values

Figure 2: Computation table for g_1^{OR}

Dragoş Rotaru
imec-Cosic, Dept. Electrical Engineering

How general is this?



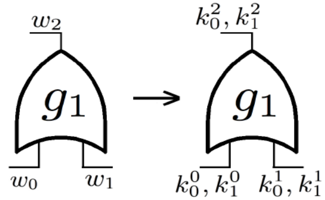


Figure 1: Garbling a single gate

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}

SPDZ $\mathbf{F}_p$

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

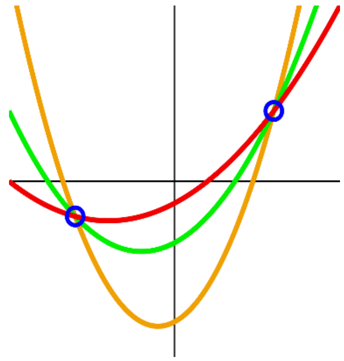
SPDZ-BMR

WRK'17

HSS'17

Very fast using DEFKSV'19 tricks

How general is this?



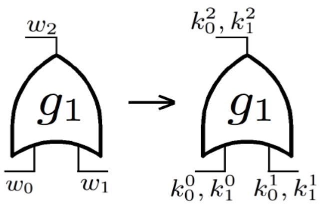


Figure 1: Garbling a single gate

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

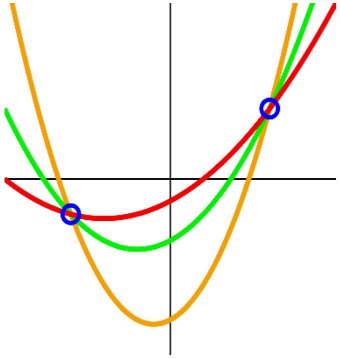
WRK’17

HSS’17

Dragoș Rotaru

imec-Cosic, Dept. Electrical Engineering

How general is this?



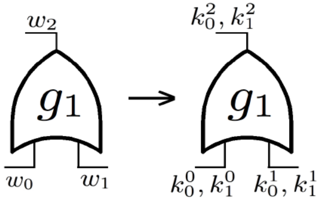


Figure 1: Garbling a single gate

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

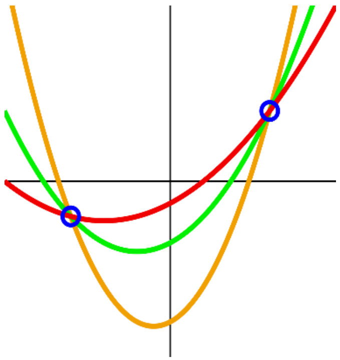
Any honest majority protocol

Dragoş Rotaru

imec-Cosic, Dept. Electrical Engineering

KU LEUVEN

Our focus



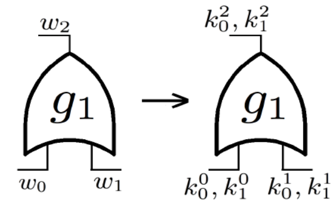


Figure 1: Garbling a single gate

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

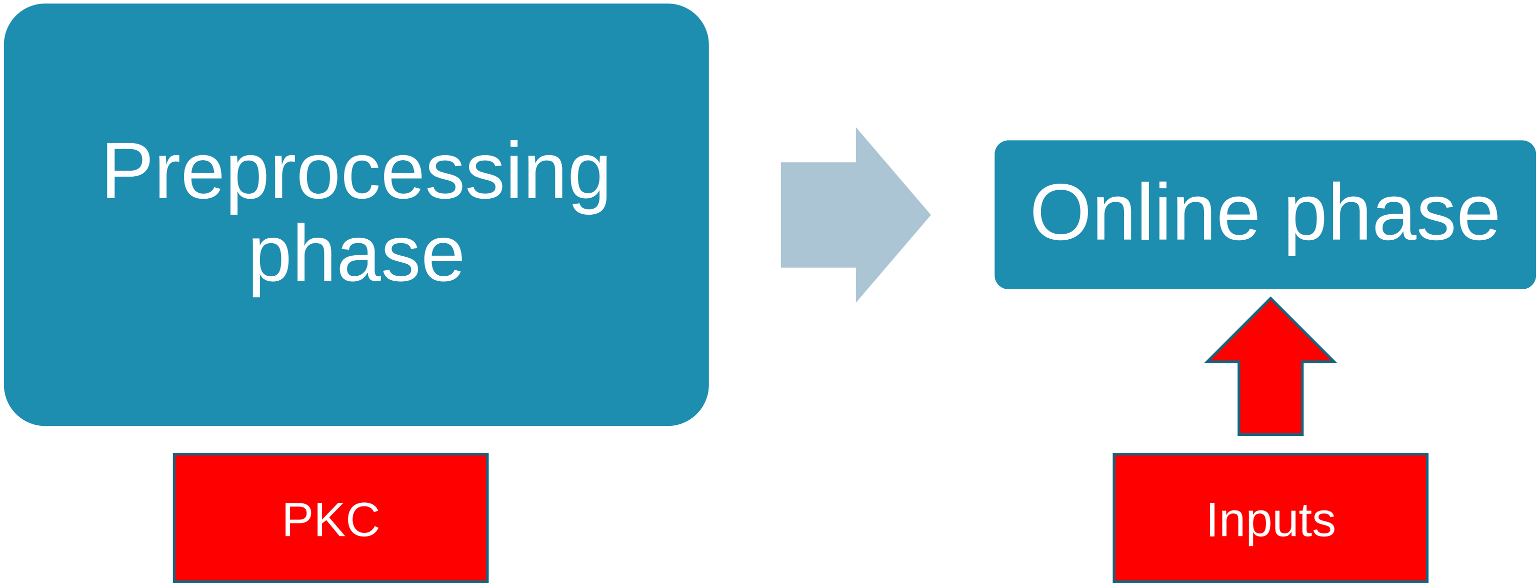
HSS’17

Dragoș Rotaru

imec-Cosic, Dept. Electrical Engineering

KU LEUVEN

Malicious MPC protocols

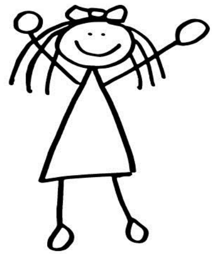


SPDZ, TinyOT, BDOZa, MASCOT, WRK’17, HSS’17, …

Let’s talk about

SPDZ	$\mathbf{F}_p$

SPDZ $\mathbf{F}_p$ online phase

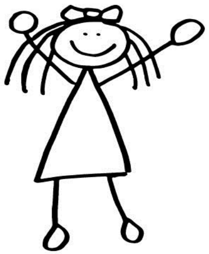




$$\alpha_1 + \alpha_2 + \alpha_3 = \boxed{\alpha}$$

$$x_1 + x_2 + x_3 = \boxed{x}$$

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

SPDZ $\mathbf{F}_p$ online phase

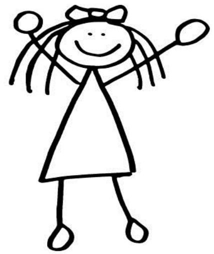
  

$$\alpha_1 + \alpha_2 + \alpha_3 = \boxed{\alpha}$$

$$x_1 + y_1 + x_2 + y_2 + x_3 + y_3 = \boxed{x + y}$$

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

SPDZ $\mathbf{F}_p$ online phase



Input

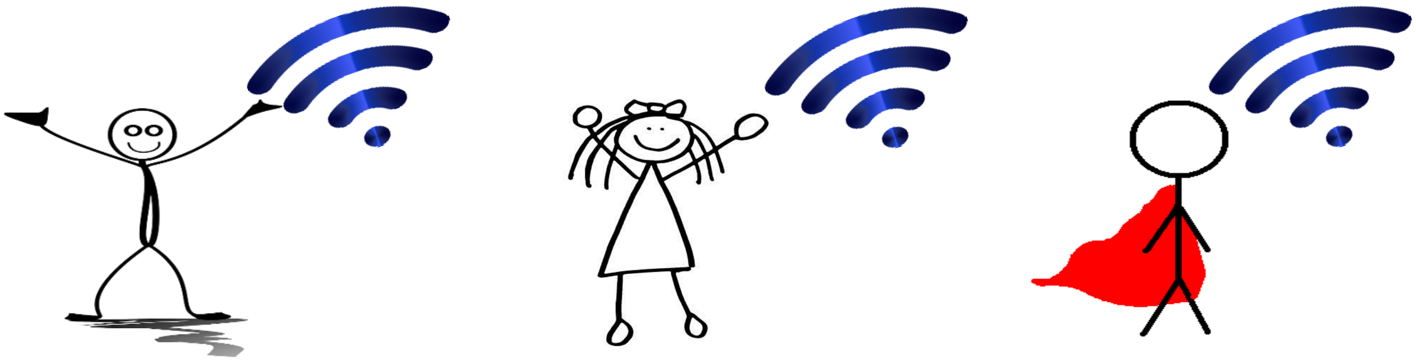
Retrieve a random mask

$\boxed{X_A} \leftarrow X_A$

SPDZ $\mathbf{F}_p$ online phase

Input $\boxed{X_A} \leftarrow X_A$

SPDZ $\mathbf{F}_p$ online phase



Input $\quad [X_A] \leftarrow X_A$

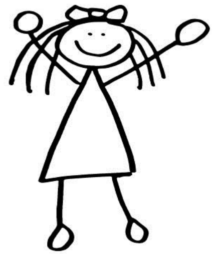
Open $\quad X \leftarrow [X]$

SPDZ $\mathbf{F}_p$ online phase

Input $\quad X_A \leftarrow X_A$

Open $\quad$ MAC Check $\quad x \leftarrow x$

SPDZ $\mathbf{F}_p$ online phase

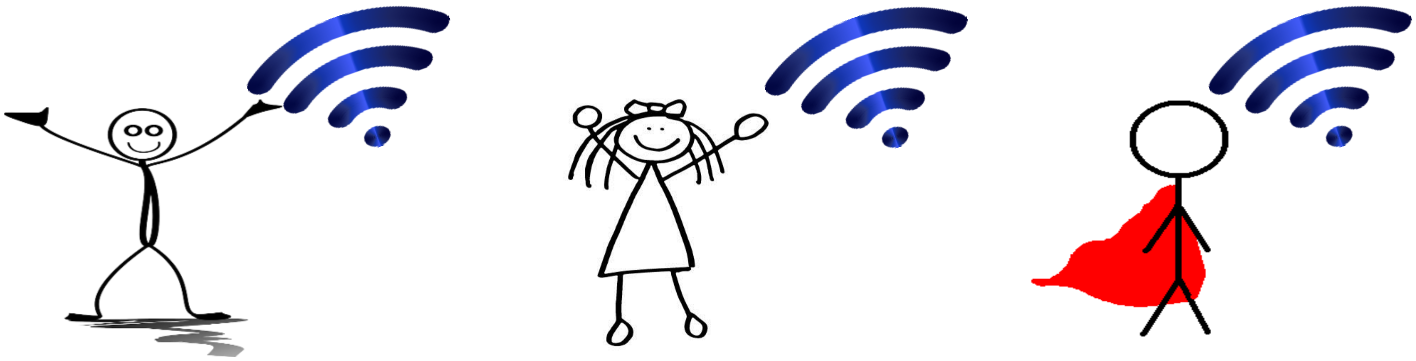


Input $\boxed{x_A} \leftarrow x_A$

Open $\boxed{x} \leftarrow \boxed{x}$

XOR Retrieve a Beaver triple $\boxed{z} \leftarrow \boxed{x} \oplus \boxed{y}$

SPDZ $\mathbf{F}_p$ online phase



Input $\quad X_A \leftarrow X_A$

Open $\quad$ MAC Check $\quad x \leftarrow x$

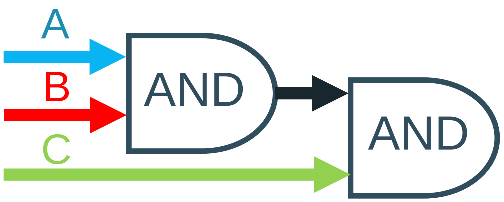
XOR $\quad z \leftarrow x \oplus y$

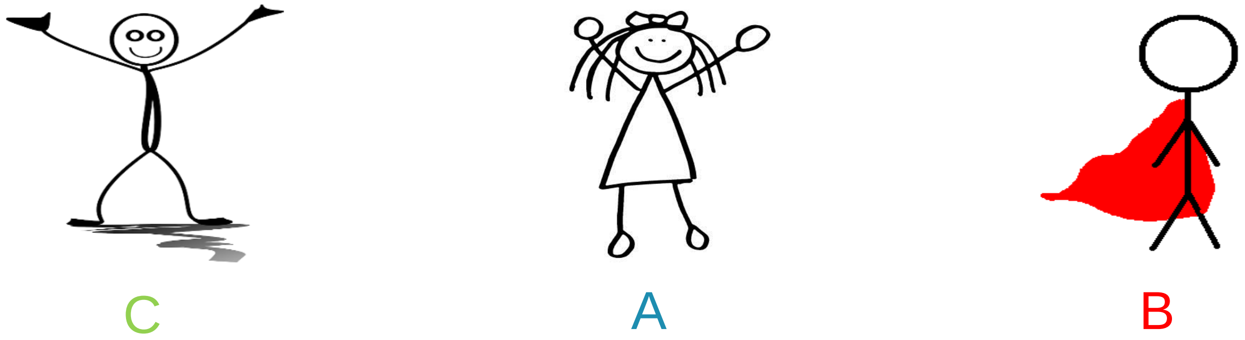
Let's talk about

WRK'17	$\mathbf{F}_2$

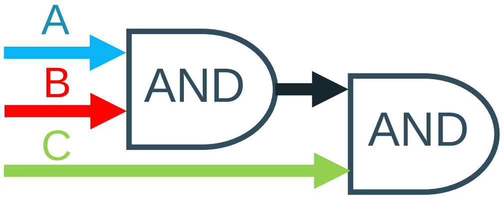
WRK'17 $\mathbb{F}_2$

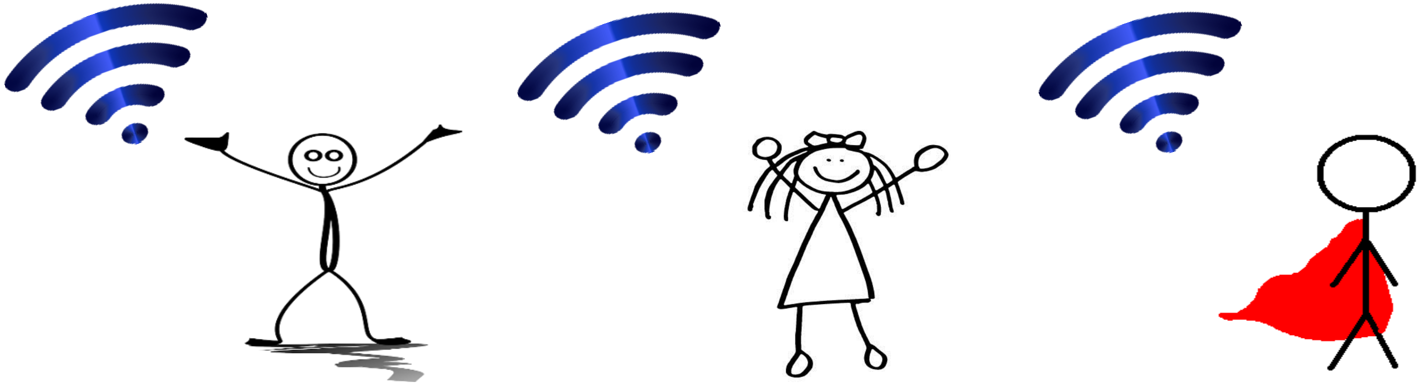
online phase





WRK'17 $\mathbb{F}_2$ online phase

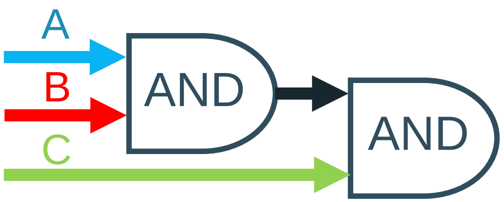


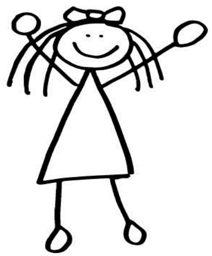


MAC Check

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

WRK'17 $\mathbb{F}_2$ online phase





MAC Check

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

Inputs - cheap

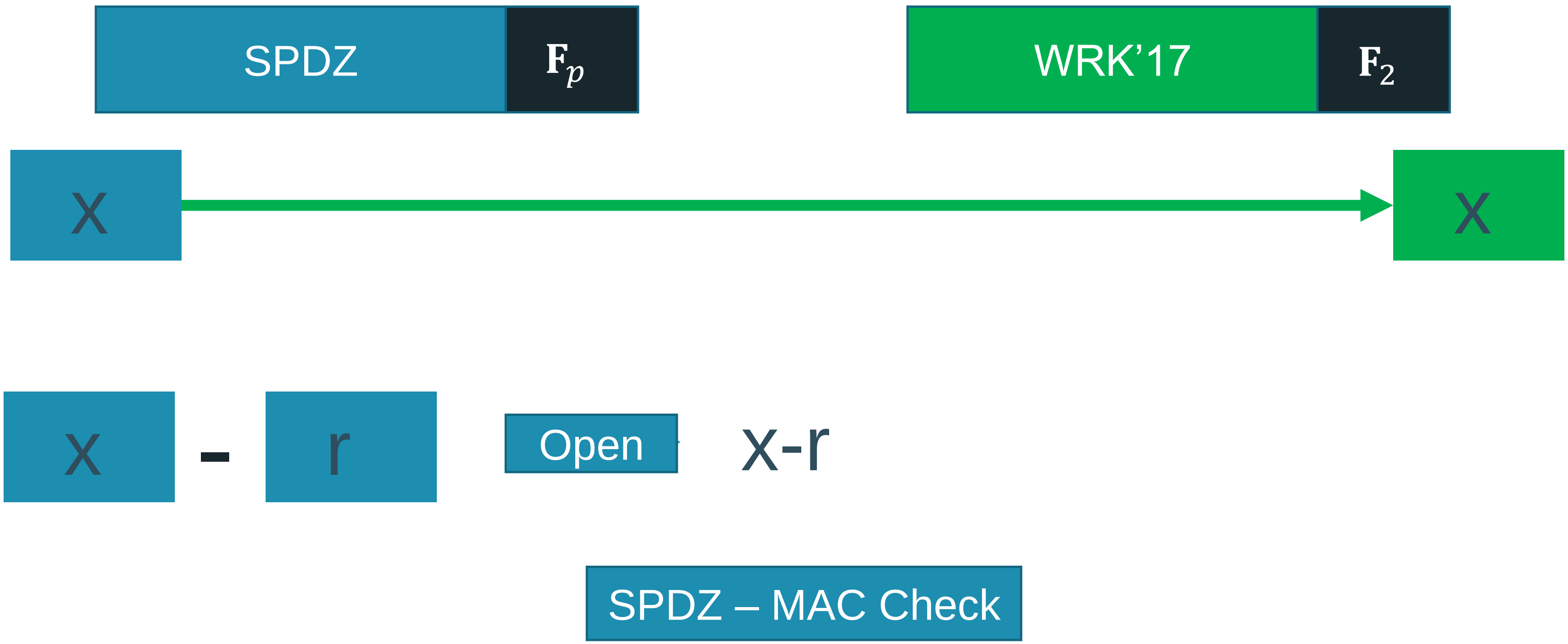
XOR - free

Mod p arithmetic - some AND gates

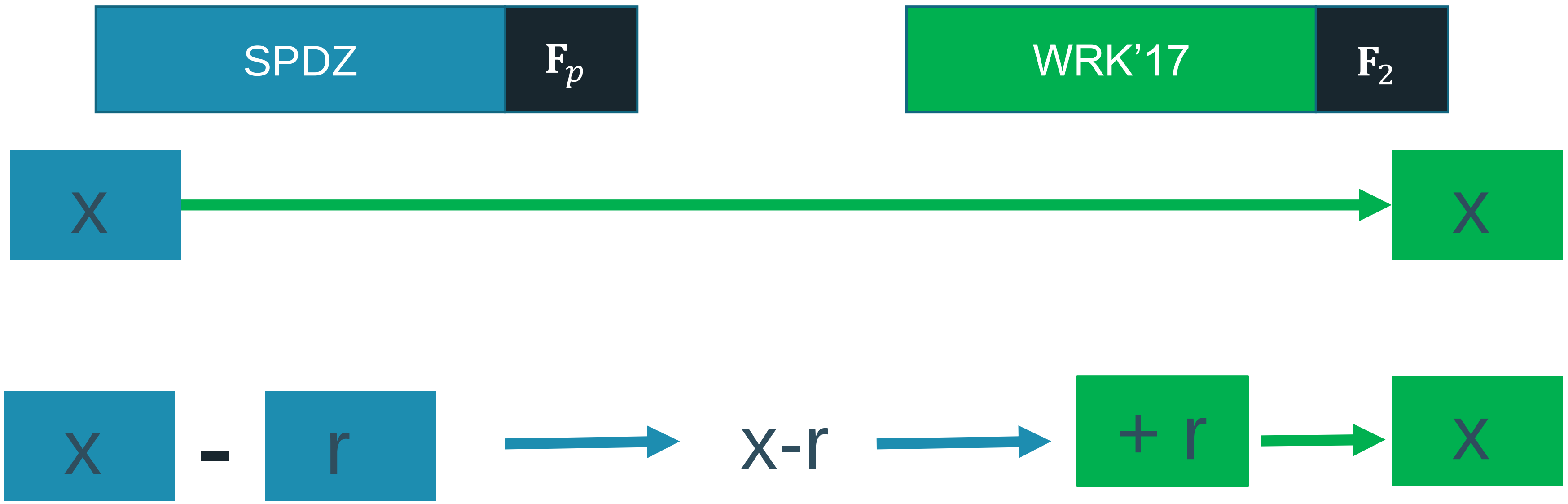
Main idea:

Main idea:

Main idea:



Main idea:

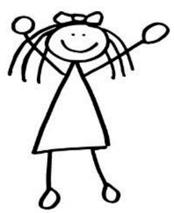


Introducing daBits

Introducing daBits

SPDZ	$\mathbf{F}_p$

WRK	$\mathbf{F}_2$



b_A

b_B

b_C

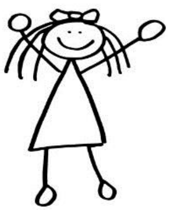
Introducing daBits

SPDZ $\mathbf{F}_p$

WRK $\mathbf{F}_2$

SPDZ Input

WRK Input



b_A b_B b_C

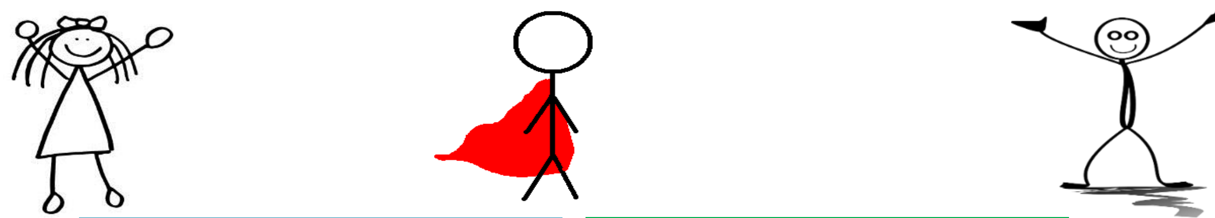
Introducing daBits

SPDZ $\mathbf{F}_p$

WRK $\mathbf{F}_2$

SPDZ Input

WRK Input



SPDZ	WRK
b_A	b_A
b_B	b_B
b_C	b_C

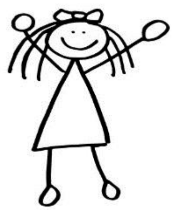
Introducing daBits

SPDZ $\mathbf{F}_p$

WRK $\mathbf{F}_2$

SPDZ Open

WRK Open



b_A

b_B

b_C

b_A

b_B

b_C

Dragoș Rotaru

imec-Cosic, Dept. Electrical Engineering

KU LEUVEN

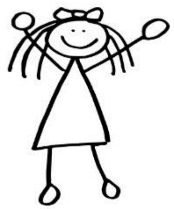
Introducing daBits

SPDZ $\mathbf{F}_p$

WRK $\mathbf{F}_2$

SPDZ XOR

WRK XOR



$b_A \oplus b_B \oplus b_C$

$b_A \oplus b_B \oplus b_C$

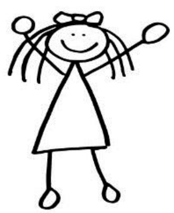
Introducing daBits

SPDZ $\mathbf{F}_p$

WRK $\mathbf{F}_2$

SPDZ Open

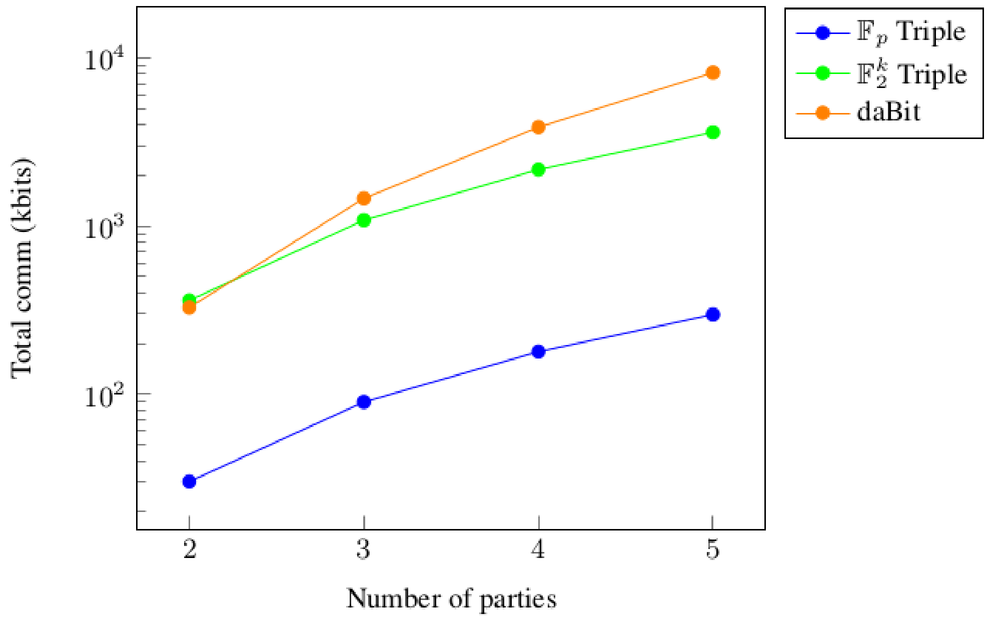
WRK Open



$b_A \oplus b_B \oplus b_C$

$b_A \oplus b_B \oplus b_C$

daBit cost



MASCOT

SPDZ

Total communication costs for all parties per preprocessed element.

Total cost per conversion

Framework	Sub-Protocol	Communication (MBytes)			Time (ms)		
		Prep.	Online	Total	Prep.	Online	Total
MP-SPDZ	LowGear	1.11		≈25	17	0.1	314
	SPDZ-BMR	23.87			297		
SCALE-MAMBA	Top Gear	1.58	0.35	2.11			93
	WRK	0.16					
—	WRK	54.46		≈55			

Conversion costs for MP-SPDZ and SCALE-MAMBA on a LAN Network.

SVM Example in MP-SPDZ

Protocol	Sub-Protocol	Online cost			Preprocessing cost		
		Comm. rounds	Time (ms)	Total (ms)	$\mathbb{F}_p$ triples	$\mathbb{F}_p$ bits	AND gates
SPDZ		54	2661	2661	19015	9797	-
SPDZ-BMR		0	2786	2786	-	-	14088217
Marbled-SPDZ	SPDZ	1	133	271.73	13056	0	-
	daBit convert	2	137		63546	0	27030
	SPDZ-BMR	0	1.73		-	-	8383

Two-party linear SVM: single-threaded (non-amortized) online phase costs and preprocessing costs with sec $= 64$.

SVM Example in MP-SPDZ

Circuit type	Sub-Protocol	Preprocessing protocol (comm.)			Total
		LowGear	WRK (indep.)	WRK (dep.)	
SPDZ		49.4 MB	-	-	49.4 MB
GC		-	4917 MB	1768 MB	6685 MB
Marbled	SPDZ	24.48 MB	-	-	108.87 MB
	daBit convert	71.13 MB	6.83 MB	2.45 MB	
	GC	-	2.92 MB	1.05 MB	

Two-party linear SVM communication cost for preprocessing in MBytes and statistical security $\mathsf{sec} = 40$.

Dragoș Rotaru
imec-Cosic, Dept. Electrical Engineering

daBit 2.0

daBit 2.0

- Inspired from **DEFKSV'19**

$\mathbf{F}_2 \rightarrow \mathbf{Z}_{2^k}$

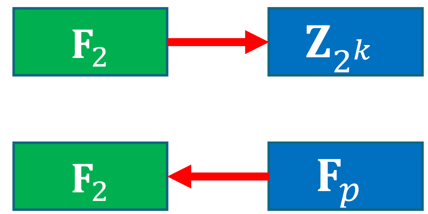
Dragoş Rotaru

imec-Cosic, Dept. Electrical Engineering

KU LEUVEN

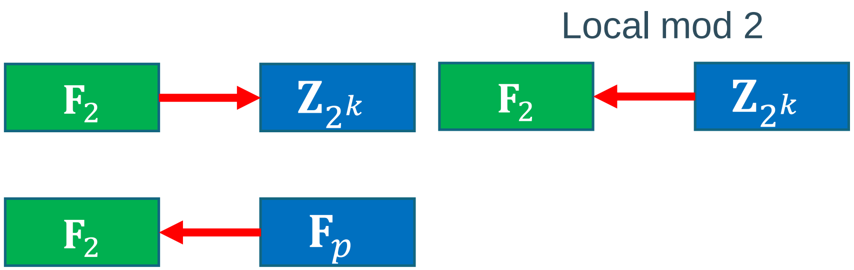
daBit 2.0

- Inspired from **DEFKSV’19**



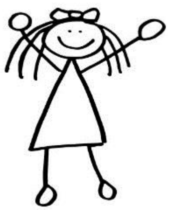
daBit 2.0

- Inspired from **DEFKSV'19**



daBit 2.0

$\mathbb{F}_2 \leftarrow \mathbb{F}_p$



SPDZ[p].Random()

$b_1 \quad \ldots \quad b_n$

$r_1 \quad \ldots \quad r_s$

$b_1^A \bmod 2 \ \ldots \ b_n^A \bmod 2$

$r_1^A \bmod 2 \ \ldots \ r_s^A \bmod 2$

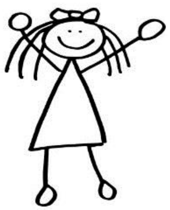
TinyOT.Input()

$b_1^B \bmod 2 \ \ldots \ b_n^B \bmod 2$

$r_1^B \bmod 2 \ \ldots \ r_s^B \bmod 2$

daBit 2.0

$\mathbf{F}_2 \leftarrow \mathbf{F}_p$



SPDZ[p].Random()

b_1 … b_n

r_1 … r_s

---

b_1^A mod 2 … b_n^A mod 2 TinyOT.Input() b_1^B mod 2 … b_n^B mod 2

r_1^A mod 2 … r_s^A mod 2 r_1^B mod 2 … r_s^B mod 2

$b_1 \leftarrow b_1^A$ mod 2 xor b_1^B mod 2 xor 1

Take s linear combinations r_i b_1 … b_n and r_i b_1 … b_n

Dragoș Rotaru

imec-Cosic, Dept. Electrical Engineering

KU LEUVEN

Conclusions and future work

- Can we generate daBits faster?
- More interesting examples where this conversions are good will come soon…

Thank you!

Thank you!

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

dabit Cost

- 0.82 ms per dabit
- 119 kbit