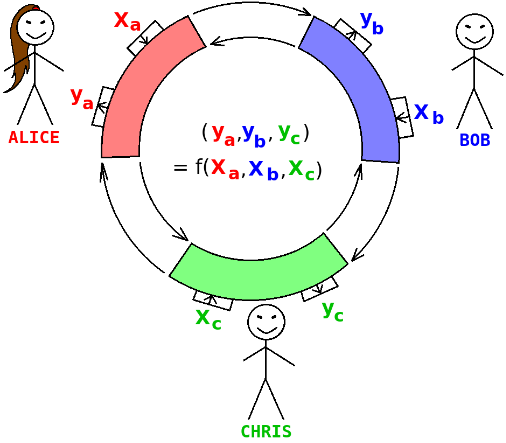
Bit Decomposition Protocols in Secure Multiparty Computation

19.10.18

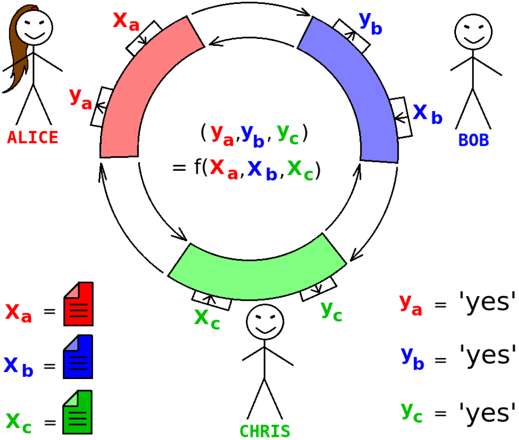
Peeter Laud Alisa Pankova

CYBERNETICA

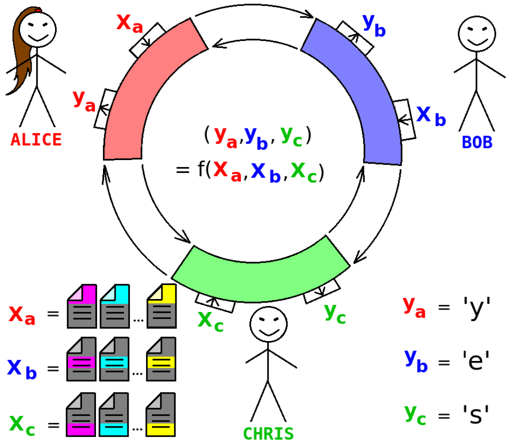
Secure Multiparty Computation



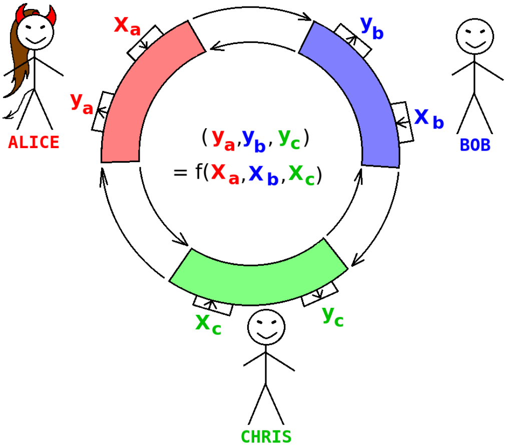
Secure Multiparty Computation



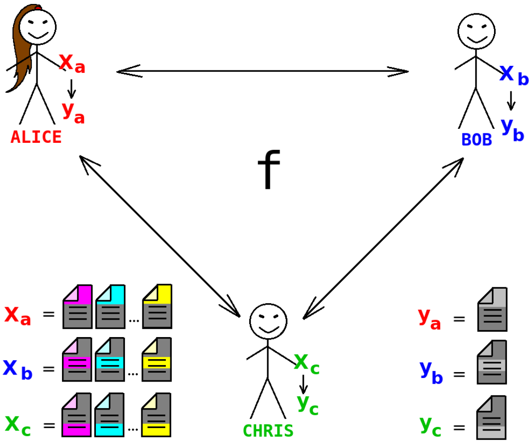
Secure Multiparty Computation



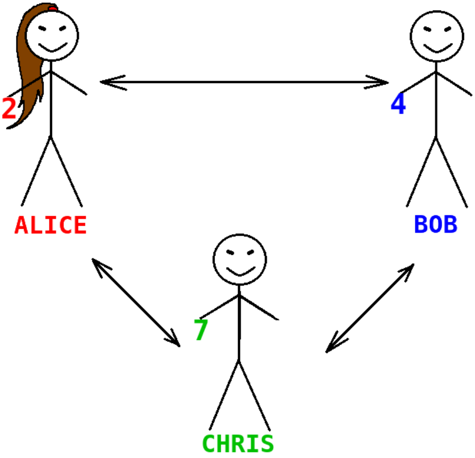
Secure Multiparty Computation



Secure Multiparty Computation

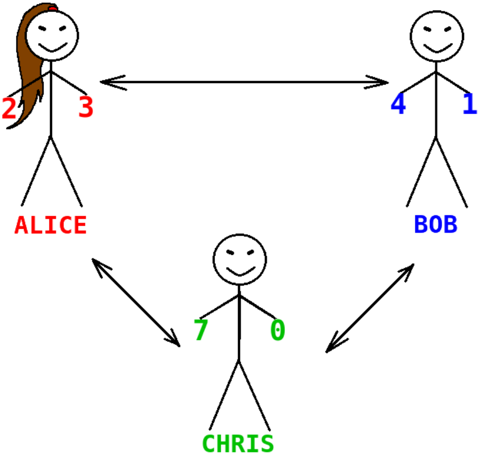


Two main types of sharing



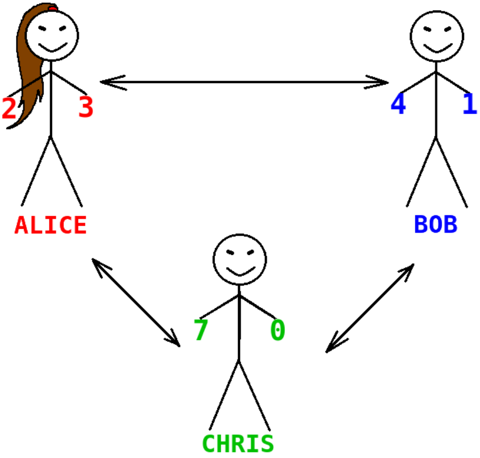
- $x = 2 + 4 + 7 = 13 = 5 \pmod{2^3}$

Two main types of sharing



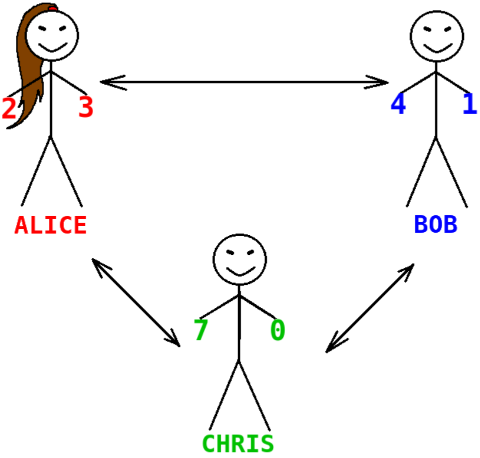
- $x = 2 + 4 + 7 = 13 = 5 \pmod{2^3}$
- $y = 3 + 1 + 0 = 4 \pmod{2^3}$

Two main types of sharing



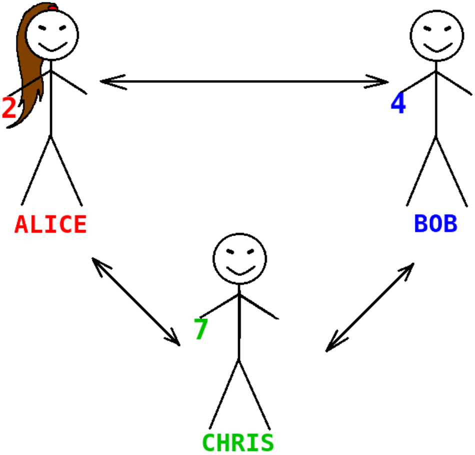
- $x = 2 + 4 + 7 = 13 = 5 \pmod{2^3}$
- $y = 3 + 1 + 0 = 4 \pmod{2^3}$
- $x + y = 2 + 3 + 4 + 1 + 7 + 0 = 17 = 1 \pmod{2^3}$

Two main types of sharing



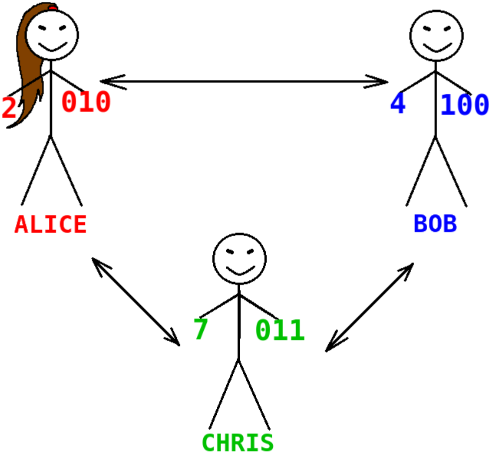
- $x = 2 + 4 + 7 = 13 = 5 \pmod{2^3}$ good for linear operations
- $y = 3 + 1 + 0 = 4 \pmod{2^3}$
- $x + y = 2 + 3 + 4 + 1 + 7 + 0 = 17 = 1 \pmod{2^3}$

Two main types of sharing



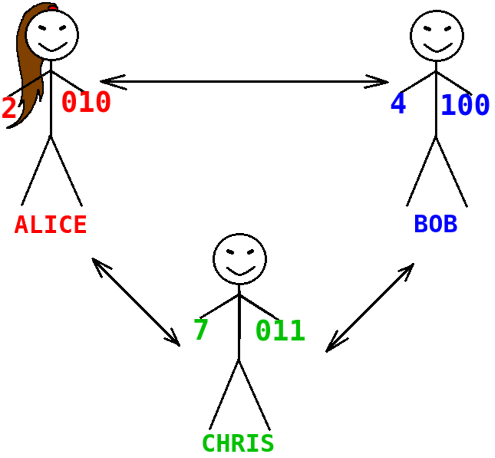
- $x = 2 + 4 + 7 = 13 = 5 \pmod{2^3}$ good for linear operations

Two main types of sharing



- $x = 2 + 4 + 7 = 13 = 5 \pmod{2^3}$ good for linear operations
- $x = 010 \oplus 100 \oplus 011 = 101_2$ good for bitwise operations

Two main types of sharing



- $x = 2 + 4 + 7 = 13 = 5 \pmod{2^3}$ good for linear operations
- $x = 010 \oplus 100 \oplus 011 = 101_2$ good for bitwise operations
- We need to convert between the two sharings $\mathbb{Z}_{2^n}$ and $\mathbb{Z}_2^n$.

Sharing a bit: over $\mathbb{Z}_2$ vs over $\mathbb{Z}_{2^n}$

$$\mathbb{Z}_2 : \square\,\square\,\boxed{1} + \square\,\square\,\boxed{1} = \square\,\square\,\boxed{0}$$

Sharing a bit: over $\mathbb{Z}_2$ vs over $\mathbb{Z}_{2^n}$

$$\mathbb{Z}_2 : \boxed{}\boxed{}\boxed{1} + \boxed{}\boxed{}\boxed{1} = \boxed{}\boxed{}\boxed{0}$$

$$\mathbb{Z}_{2^3} : \boxed{0}\boxed{0}\boxed{1} + \boxed{0}\boxed{0}\boxed{1} = \boxed{0}\boxed{1}\boxed{0}$$

Sharing a bit: over $\mathbb{Z}_2$ vs over $\mathbb{Z}_{2^n}$

$$\mathbb{Z}_2: \quad \boxed{\ }\boxed{\ }\boxed{1} + \boxed{\ }\boxed{\ }\boxed{1} = \boxed{\ }\boxed{\ }\boxed{0}$$

$$\mathbb{Z}_{2^3}: \quad \boxed{0}\boxed{0}\boxed{1} + \boxed{0}\boxed{0}\boxed{1} = \boxed{0}\boxed{1}\boxed{0}$$

$$\boxed{0}\boxed{0}\boxed{0} + \boxed{0}\boxed{0}\boxed{0} = \boxed{0}\boxed{0}\boxed{0}$$

Sharing a bit: over $\mathbb{Z}_2$ vs over $\mathbb{Z}_{2^n}$

$\mathbb{Z}_2$:			1	+			1	=			0
$\mathbb{Z}_{2^3}$:	0	0	1	+	0	0	1	=	0	1	0
	0	0	0	+	0	0	0	=	0	0	0
	1	0	1	+	0	1	1	=	0	0	0

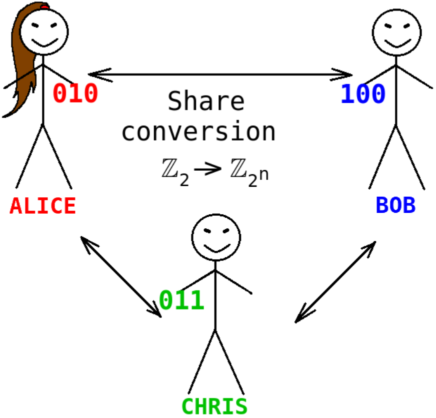
Sharing a bit: over $\mathbb{Z}_2$ vs over $\mathbb{Z}_{2^n}$

$\mathbb{Z}_2$:	1	+	1	=	0
$\mathbb{Z}_{2^3}$:	0 0 1	+	0 0 1	=	0 1 0
	0 0 0	+	0 0 0	=	0 0 0
	1 0 1	+	0 1 1	=	0 0 0

There exists Share conversion SMC protocol: $\mathbb{Z}_2 \to \mathbb{Z}_{2^n}$

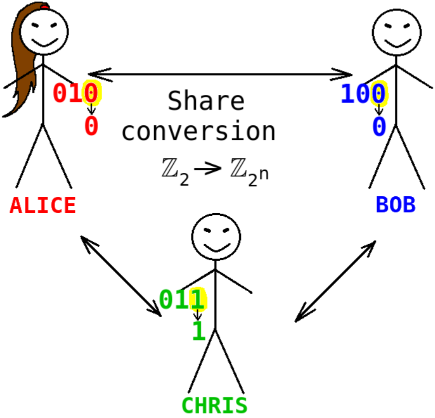
- Let us use it as a black box.

The first method for $\mathbb{Z}_2^n \to \mathbb{Z}_{2^n}$: bitwise conversion



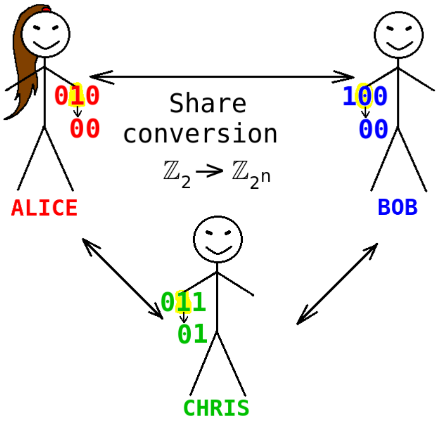
- $x = 010 \oplus 100 \oplus 011 = 101_2 = 5$

The first method for $\mathbb{Z}_2^n \to \mathbb{Z}_{2^n}$: bitwise conversion



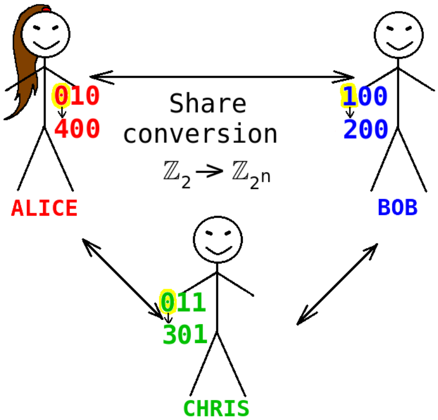
- $x = 010 \oplus 100 \oplus 011 = 101_2 = 5$

The first method for $\mathbb{Z}_2^n \to \mathbb{Z}_{2^n}$: bitwise conversion



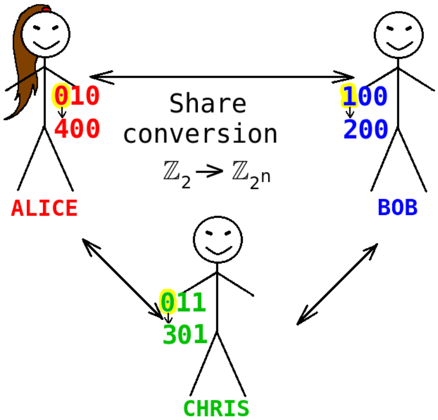
- $x = 010 \oplus 100 \oplus 011 = 101_2 = 5$

The first method for $\mathbb{Z}_2^n \to \mathbb{Z}_{2^n}$: bitwise conversion



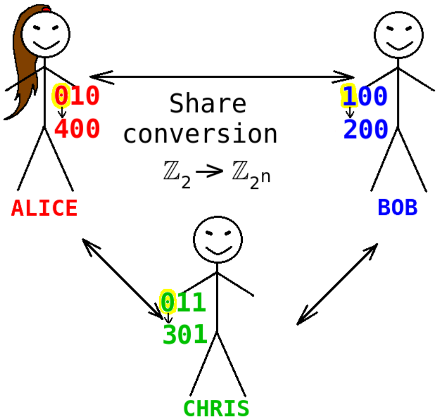
- $x = 010 \oplus 100 \oplus 011 = 101_2 = 5$

The first method for $\mathbb{Z}_2^n \to \mathbb{Z}_{2^n}$: bitwise conversion



- $x = 010 \oplus 100 \oplus 011 = 101_2 = 5$
- $x = (4 \cdot 4 + 2 \cdot 0 + 0) + (4 \cdot 2 + 2 \cdot 0 + 0) + (4 \cdot 3 + 2 \cdot 0 + 1) = 16 + 8 + 12 + 1 = 37 = 5 \pmod{2^3}$

The first method for $\mathbb{Z}_2^n \to \mathbb{Z}_{2^n}$: bitwise conversion



- $x = 010 \oplus 100 \oplus 011 = 101_2 = 5$
- $x = (4 \cdot 4 + 2 \cdot 0 + 0) + (4 \cdot 2 + 2 \cdot 0 + 0) + (4 \cdot 3 + 2 \cdot 0 + 1) = 16 + 8 + 12 + 1 = 37 = 5 \pmod{2^3}$
- Need n share conversions for n-bit numbers.

Sharing in $\mathbb{Z}_{2^n}$ is a sum of sharings over $\mathbb{Z}_2^n$

$$\mathbb{Z}_{2^3}\text{:}\ \boxed{1\,|\,0\,|\,1} + \boxed{0\,|\,1\,|\,1} = \boxed{0\,|\,0\,|\,0}$$

Sharing in $\mathbb{Z}_{2^n}$ is a sum of sharings over $\mathbb{Z}_2^n$

$\mathbb{Z}_{2^3}$: $\boxed{1\,|\,0\,|\,1} + \boxed{0\,|\,1\,|\,1} = \boxed{0\,|\,0\,|\,0}$

$\mathbb{Z}_2^3$: $\boxed{1\,|\,0\,|\,1} \oplus \boxed{0\,|\,0\,|\,0} = \boxed{1\,|\,0\,|\,1}$

$\boxed{0\,|\,0\,|\,0} \oplus \boxed{0\,|\,1\,|\,1} = \boxed{0\,|\,1\,|\,1}$

Sharing in $\mathbb{Z}_{2^n}$ is a sum of sharings over $\mathbb{Z}_2^n$

$$\mathbb{Z}_{2^3}\colon\ \boxed{1\,|\,0\,|\,1} + \boxed{0\,|\,1\,|\,1} = \boxed{0\,|\,0\,|\,0}$$

$$\mathbb{Z}_2^3\colon\ \boxed{1\,|\,0\,|\,1} \oplus \boxed{0\,|\,0\,|\,0} = \boxed{1\,|\,0\,|\,1}$$

$$\boxed{0\,|\,0\,|\,0} \oplus \boxed{0\,|\,1\,|\,1} = \boxed{0\,|\,1\,|\,1}$$

$$\boxed{1\,|\,1\,|\,0} \oplus \boxed{1\,|\,1\,|\,1} = \boxed{1\,|\,0\,|\,1}$$

Sharing in $\mathbb{Z}_{2^n}$ is a sum of sharings over $\mathbb{Z}_2^n$

$$\mathbb{Z}_{2^3}:\ \boxed{1}\boxed{0}\boxed{1} + \boxed{0}\boxed{1}\boxed{1} = \boxed{0}\boxed{0}\boxed{0}$$

$$\begin{array}{lc}
\mathbb{Z}_2^3: & \boxed{1}\boxed{0}\boxed{1} \oplus \boxed{0}\boxed{0}\boxed{0} = \boxed{1}\boxed{0}\boxed{1} \\
 & \boxed{0}\boxed{0}\boxed{0} \oplus \boxed{0}\boxed{1}\boxed{1} = \boxed{0}\boxed{1}\boxed{1} \\
\hline
 & \boxed{1}\boxed{1}\boxed{0} \oplus \boxed{1}\boxed{1}\boxed{1} = \boxed{1}\boxed{0}\boxed{1}
\end{array}$$

This gives us conversion: $\mathbb{Z}_{2^n} \to \mathbb{Z}_2^n$

Sharing in $\mathbb{Z}_{2^n}$ is a sum of sharings over $\mathbb{Z}_2^n$

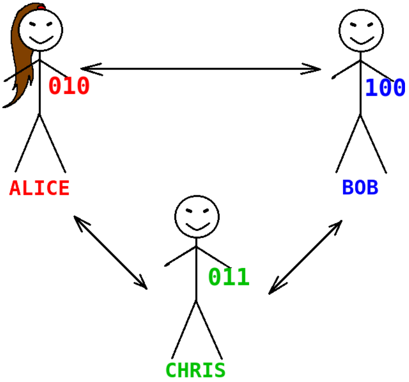
$$\mathbb{Z}_{2^3}\colon \boxed{1\,|\,0\,|\,1} + \boxed{0\,|\,1\,|\,1} = \boxed{0\,|\,0\,|\,0}$$

$$\begin{array}{rcccccc}
\mathbb{Z}_2^3\colon & \boxed{1\,|\,0\,|\,1} & \oplus & \boxed{0\,|\,0\,|\,0} & = & \boxed{1\,|\,0\,|\,1} \\
& \boxed{0\,|\,0\,|\,0} & \oplus & \boxed{0\,|\,1\,|\,1} & = & \boxed{0\,|\,1\,|\,1} \\
\hline
& \boxed{1\,|\,1\,|\,0} & \oplus & \boxed{1\,|\,1\,|\,1} & = & \boxed{1\,|\,0\,|\,1}
\end{array}$$

This gives us conversion: $\mathbb{Z}_{2^n} \to \mathbb{Z}_2^n$

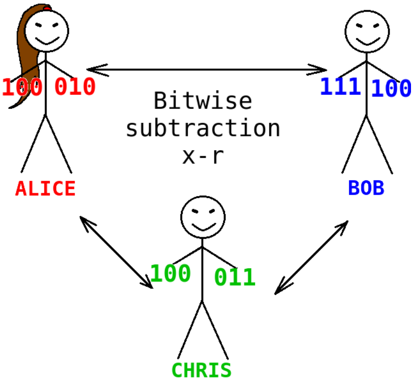
- We can apply the same idea for $\mathbb{Z}_2^n \to \mathbb{Z}_{2^n}$.

The second method for $\mathbb{Z}_2^n \to \mathbb{Z}_{2^n}$: bitwise subtraction



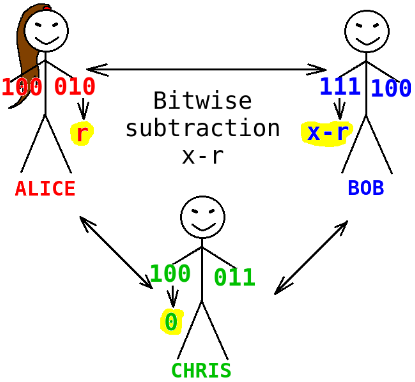
- $x = 010 \oplus 100 \oplus 011 = 101_2 = 5$

The second method for $\mathbb{Z}_2^n \to \mathbb{Z}_{2^n}$: bitwise subtraction



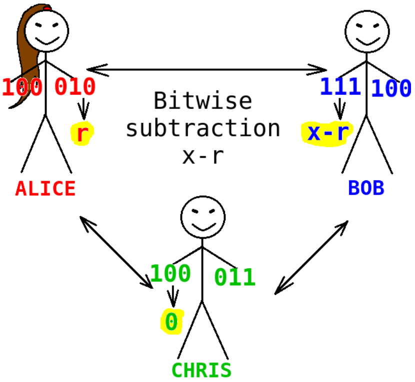
- $x = 010 \oplus 100 \oplus 011 = 101_2 = 5$
- $r = 100 \oplus 111 \oplus 100 = 111_2 = 7$

The second method for $\mathbb{Z}_2^n \to \mathbb{Z}_{2^n}$: bitwise subtraction



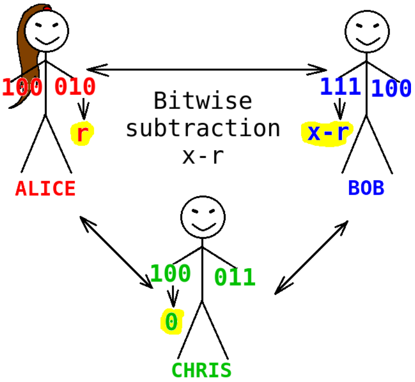
- $x = 010 \oplus 100 \oplus 011 = 101_2 = 5$
- $r = 100 \oplus 111 \oplus 100 = 111_2 = 7$
- Open r to Alice, and x - r to Bob, let Chris take 0.

The second method for $\mathbb{Z}_2^n \to \mathbb{Z}_{2^n}$: bitwise subtraction



- $x = 010 \oplus 100 \oplus 011 = 101_2 = 5$
- $r = 100 \oplus 111 \oplus 100 = 111_2 = 7$
- Open r to Alice, and x - r to Bob, let Chris take 0.
- r + x - r + 0 = x

The second method for $\mathbb{Z}_2^n \to \mathbb{Z}_{2^n}$: bitwise subtraction



- $x = 010 \oplus 100 \oplus 011 = 101_2 = 5$
- $r = 100 \oplus 111 \oplus 100 = 111_2 = 7$
- Open r to Alice, and x - r to Bob, let Chris take 0.
- r + x - r + 0 = x
- Need one bitwise subtraction of an *n*-bit number.

The bitwise subtraction protocol

- Let $[\![x]\!] = (x, x, x)$ denote a secret-shared value x.

Algorithm 1: Secure subtraction of XOR-shared numbers

Data: $n \in \mathbb{N}$, shared bits $[\![x_0]\!], \ldots, [\![x_{n-1}]\!], [\![y_0]\!], \ldots, [\![y_{n-1}]\!] \in \mathbb{Z}_2$

$[\![c_0]\!] = 0$
for $k = 0$ **to** $n - 1$ **do**
  $[\![s_k]\!] = [\![x_k]\!] \oplus [\![y_k]\!] \oplus [\![c_k]\!]$;
  $[\![c_{k+1}]\!] = (([\![c_k]\!] \oplus [\![y_k]\!]) \wedge ([\![c_k]\!] \oplus [\![s_k]\!])) \oplus [\![c_k]\!]$
end
Return $[\![s_0]\!], \ldots, [\![s_{n-1}]\!]$.

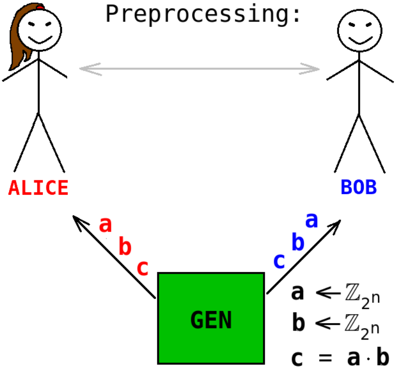
Uses n secure AND-s.

Comparing the two methods

- Bitwise subtraction: generic, uses only secure AND / XOR.
  - The complexity is n ANDs.
  - (+ opening of x and $x - r$).
- Bitwise conversion: elaborated, uses 3-party benefits.
  - The complexity is n conversions $\mathbb{Z}_2 \to \mathbb{Z}_{2^n}$.
  - Was preferred by Sharemind system.

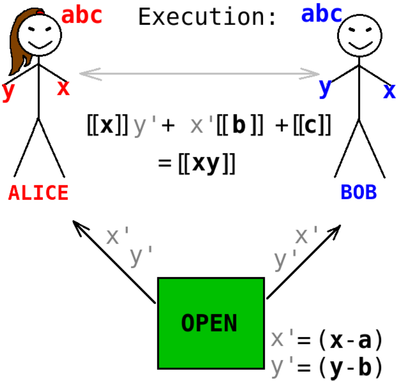
Assisted 2-party computation using correlated randomness

- Beaver triples: $([\![a]\!], [\![b]\!], [\![c]\!])$ where $a, b \xleftarrow{\$} \mathbb{Z}_{2^n}$, $c = a \cdot b$.
  - Complexity of generating one n-bit triple is $O(n)$.



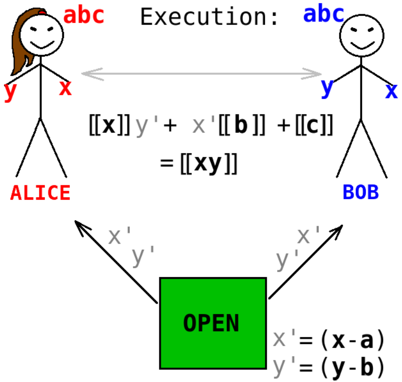
Assisted 2-party computation using correlated randomness

- Beaver triples: $([\![a]\!], [\![b]\!], [\![c]\!])$ where $a, b \xleftarrow{\$} \mathbb{Z}_{2^n}$, $c = a \cdot b$.
  - Complexity of generating one n-bit triple is $O(n)$.

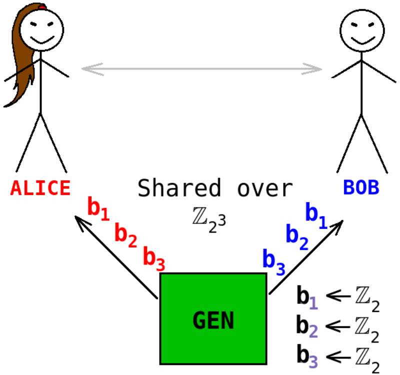


Assisted 2-party computation using correlated randomness

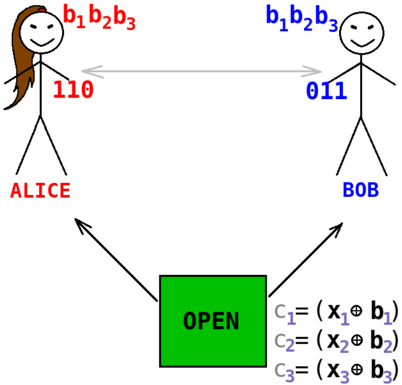
- Beaver triples: $([\![a]\!], [\![b]\!], [\![c]\!])$ where $a, b \xleftarrow{\$} \mathbb{Z}_{2^n}$, $c = a \cdot b$.
  - Complexity of generating one n-bit triple is $O(n)$.
- Trusted bits: $[\![b]\!]$ where $b \xleftarrow{\$} \mathbb{Z}_2$ is shared over $\mathbb{Z}_{2^n}$.
  - Complexity of generating one trusted bit is $O(n)$.



Using trusted bits for $\mathbb{Z}_2^n \to \mathbb{Z}_{2^n}$ (previous work)

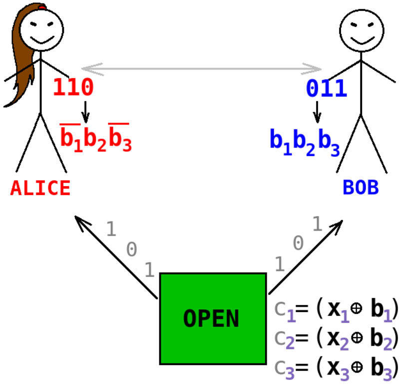


Using trusted bits for $\mathbb{Z}_2^n \to \mathbb{Z}_{2^n}$ (previous work)



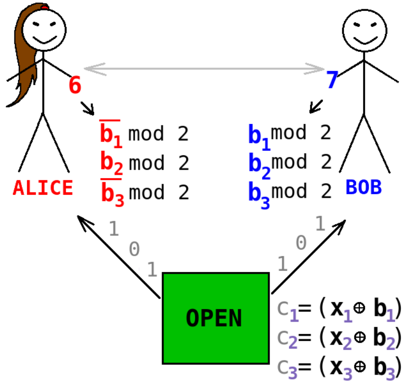
- $x = 110 \oplus 011 = 101_2$
- $x_1 = 1, x_2 = 0, x_3 = 1$

Using trusted bits for $\mathbb{Z}_2^n \to \mathbb{Z}_{2^n}$ (previous work)



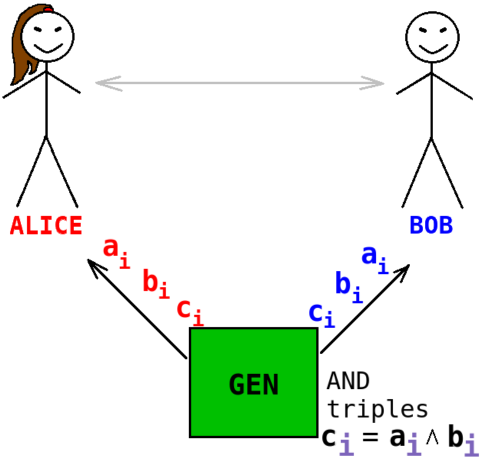
- $x = 110 \oplus 011 = 101_2$
- $x_1 = 1,\ x_2 = 0,\ x_3 = 1$
- $x = \overline{b_3} b_2 \overline{b_1} \oplus b_3 b_2 b_1 = 101_2$
- $x = (4 \cdot \overline{b_3} + 2 \cdot b_2 + \overline{b_1}) + (4 \cdot b_3 + 2 \cdot b_2 + b_1) = 5 \pmod{2^3}$

Using trusted bits for $\mathbb{Z}_{2^n} \to \mathbb{Z}_2^n$ (previous work)

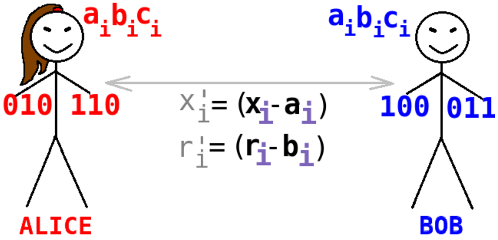


- $x = 6 + 7 = 5 \pmod{2^3} = 101_2$
- $x_1 = 1,\ x_2 = 0,\ x_3 = 1$
- $x = \overline{b_3} b_2 \overline{b_1} \oplus b_3 b_2 b_1 = 101_2$
- $x = (4 \cdot \overline{b_3} + 2 \cdot b_2 + \overline{b_1}) + (4 \cdot b_3 + 2 \cdot b_2 + b_1) = 5 \pmod{2^3}$

Using Beaver triples for $\mathbb{Z}_2^n \to \mathbb{Z}_{2^n}$ (this work)

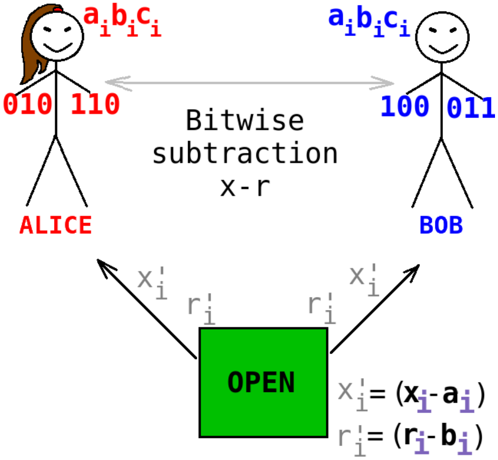


Using Beaver triples for $\mathbb{Z}_2^n \to \mathbb{Z}_{2^n}$ (this work)



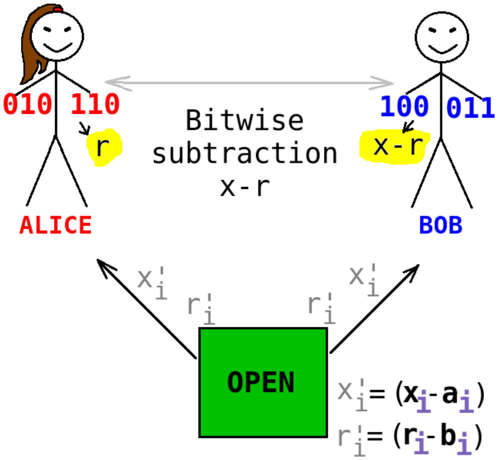
- $x = 110 \oplus 011 = 101_2$
- $r = 010 \oplus 100 = 110_2$

Using Beaver triples for $\mathbb{Z}_2^n \to \mathbb{Z}_{2^n}$ (this work)



- $x = 110 \oplus 011 = 101_2$
- $r = 010 \oplus 100 = 110_2$

Using Beaver triples for $\mathbb{Z}_2^n \to \mathbb{Z}_{2^n}$ (this work)



- $x = 110 \oplus 011 = 101_2$
- $r = 010 \oplus 100 = 110_2$

Using Beaver triples for $\mathbb{Z}_{2^n} \to \mathbb{Z}_2^n$ (this work)

- $x = 6 + 7 = 13 = 5 \pmod{2^3}$

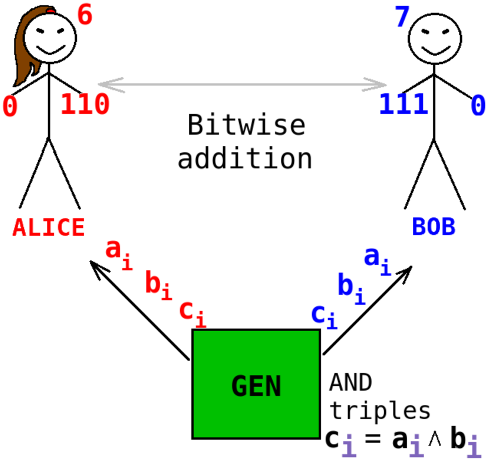
Using Beaver triples for $\mathbb{Z}_{2^n} \to \mathbb{Z}_2^n$ (this work)

- $x = 6 + 7 = 13 = 5 \pmod{2^3}$

Using Beaver triples for $\mathbb{Z}_{2^n} \to \mathbb{Z}_2^n$ (this work)

- $x = 6 + 7 = 13 = 5 \pmod{2^3}$
- $x = (4 \cdot 1 + 2 \cdot 1 + 0) + (4 \cdot 1 + 2 \cdot 1 + 1) = 13 = 5 \pmod{2^3}$

Using Beaver triples for $\mathbb{Z}_{2^n} \to \mathbb{Z}_2^n$ (this work)

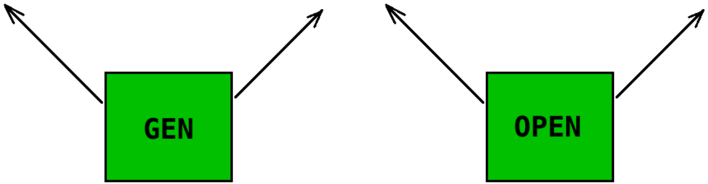


- $x = 6 + 7 = 13 = 5 \pmod{2^3}$
- $x = (4 \cdot 1 + 2 \cdot 1 + 0) + (4 \cdot 1 + 2 \cdot 1 + 1) = 13 = 5 \pmod{2^3}$

Summary for assisted 2-party $\mathbb{Z}_2^n \leftrightarrow \mathbb{Z}_{2^n}$ conversions

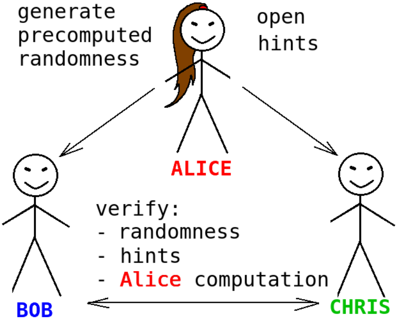
	correlated randomness	communication cost	
		preprocessing	verification
Bitwise conversion	Beaver triples	$2 \cdot n^2$	n
Bitwise addition	trusted bits	$2 \cdot 3n$	$2n$

(without taking into account the inside of the *Magic Box*).



Sharemind 3-party actively secure computation

1. Precompute sufficiently Beaver triples and trusted bits.
2. Run 3-party passively secure protocol.
3. Run assisted 2-party actively secure protocol to verify each party's computation.
   - The prover acts as an assistant of the verifiers.



Improvements for active security

Integer operation	Total bit communication for 32-bit integer protocols	
	Using bitwise conversion (old)	Using bitwise addition (new)
$[\![x]\!] \cdot [\![y]\!]$	192 : 768 : 4034 *1 : 4 : **21***	192 : 768 : 4034 *1 : 4 : **21***
$[\![x]\!] / [\![y]\!]$	31.7k : 275k : 28M *1 : 8 : **884***	31.7k : 358k : 1.7M *1 : 11 : **54***
$[\![x]\!] \ll [\![y]\!]$	1296 : 9374 : 64.6k *1 : 7 : **50***	1296 : 9758 : 50.9k *1 : 7 : **39***
$[\![x]\!] \gg [\![y]\!]$	2384 : 53k : 303.6k *1 : 22 : **127***	2608 : 53.0k : 278.8k *1.1 : 22.2 : **117***
$[\![x]\!] \gg y$	1092 : 9946 : 184.1k *1 : 9 : **169***	1092 : 12.5k : 61.2k *1 : 11 : **56***
$[\![x]\!] = [\![y]\!]$	218 : 872 : 14.3k *1 : 4 : **66***	218 : 1000 : 5252 *1 : 4 : **24***

Result is of the form x : y : z where

- x is the online passively secure phase;
- y is the verification phase;
- z is the preprocessing phase.