High-Order Conversion From Boolean to Arithmetic Masking

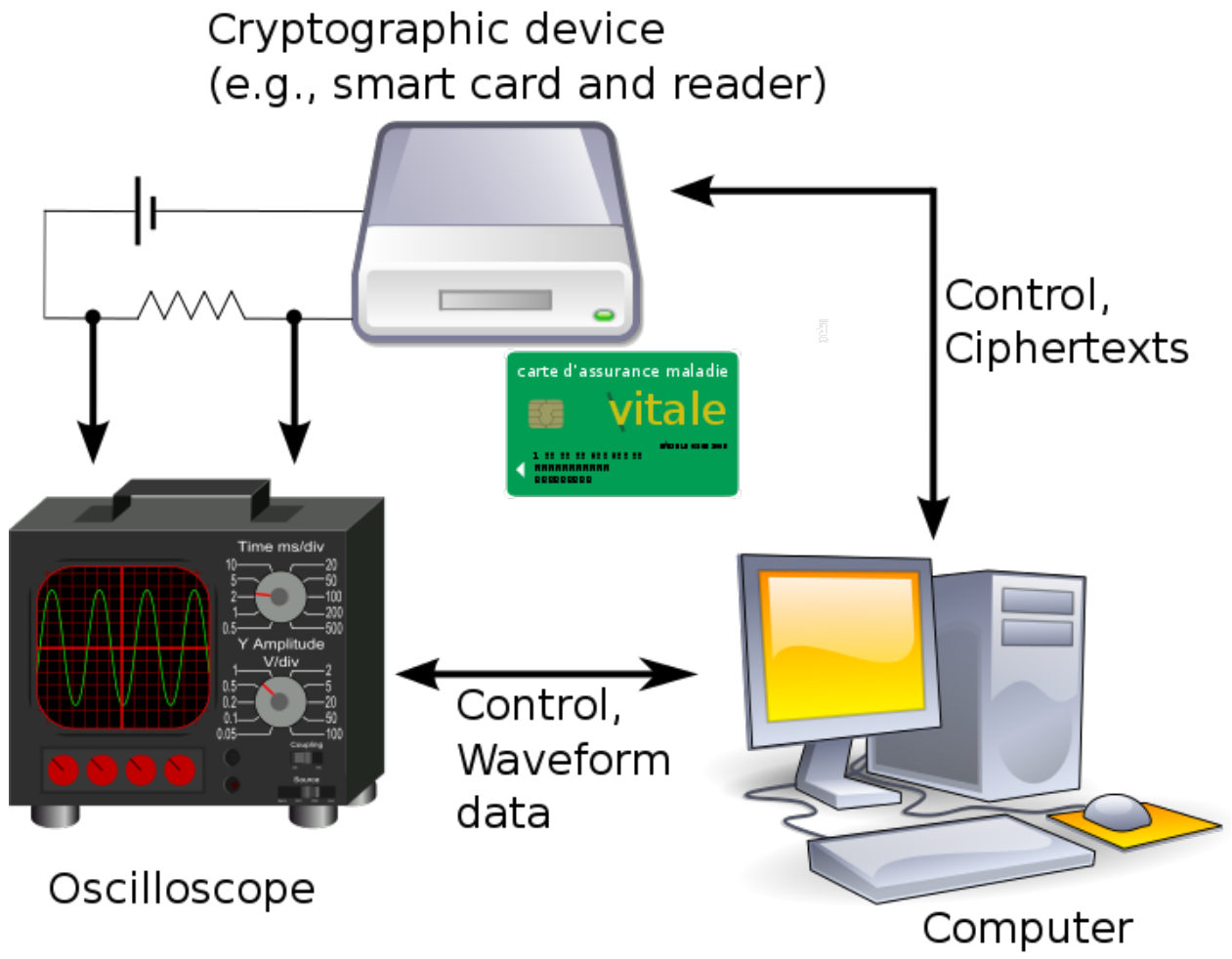
Jean-Sébastien Coron

University of Luxembourg

CHES 2017

Speaker: Srinivas Vivek

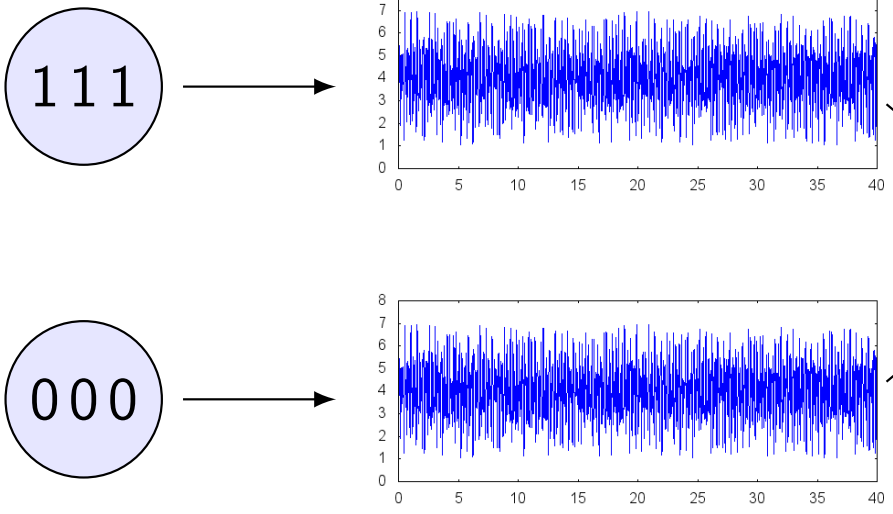
Side-channel Attacks



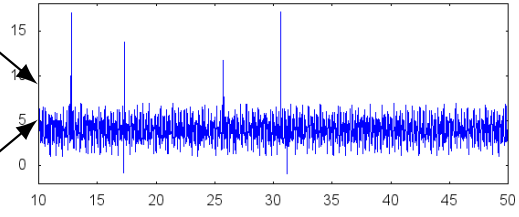
Differential Power Analysis [KJJ99]

Group by predicted SBox output bit

Average trace



Differential trace

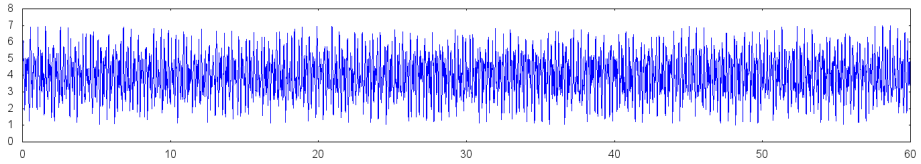


Masking Countermeasure

- Let x be some variable in a block-cipher.
- Masking countermeasure: generate a random r, and manipulate the masked value x'

$$x' = x \oplus r$$

  instead of x.
- r is random $\Rightarrow$ x' is random
  $\Rightarrow$ power consumption of x' is random



  $\Rightarrow$ no information about x is leaked

Masking Countermeasure

- How do we compute with $x' = x \oplus r$ instead of x ?
- Linear operation $f(x)$ (*e.g.* `MixColumns` in AES): easy

$$f(x') = f(x) \oplus f(r)$$

  - We compute $f(x')$ and $f(r)$ separately.
  - $f(x)$ is now masked with $f(r)$ instead of r.
- Non-linear operations (SBOX): randomized table [CJRR99]

Arithmetic Masking

- Some algorithms use arithmetic operations, for example IDEA, RC6, XTEA, SPECK, SHA-1.
- For these algorithms, we can use arithmetic masking:

$$x = A + r \bmod 2^k$$

  where we manipulate A and r separately.
- Problem: how do we convert between Boolean and arithmetic masking ?
  - Goubin's algorithm (CHES 01): first-order secure conversion between Boolean and arithmetic masking.

Arithmetic Masking

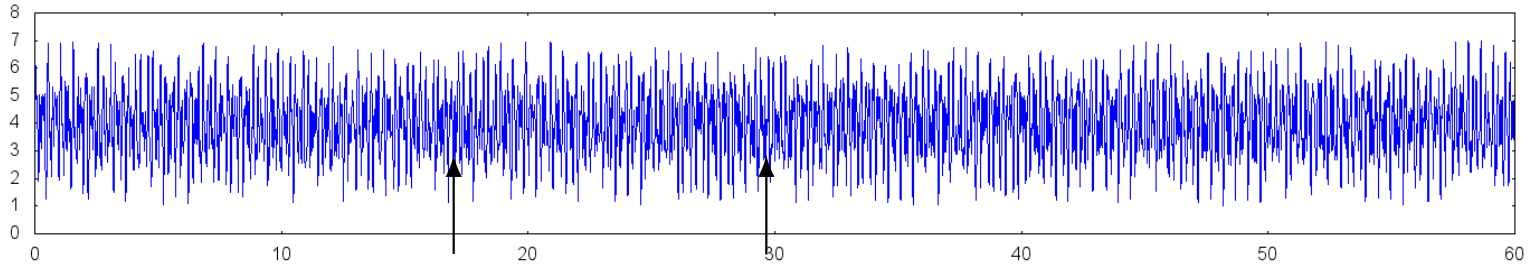
- Some algorithms use arithmetic operations, for example IDEA, RC6, XTEA, SPECK, SHA-1.
- For these algorithms, we can use arithmetic masking:

$$x = A + r \bmod 2^k$$

  where we manipulate A and r separately.
- Problem: how do we convert between Boolean and arithmetic masking ?
  - Goubin's algorithm (CHES 01): first-order secure conversion between Boolean and arithmetic masking.

Second-order Attack

- Second-order attack:



$E(x')$ $E(r)$

$f(E(x'), E(r))$ correlated with $x = x' \oplus r$

- Requires more curves but can be practical

Higher-order masking

- Solution: n shares instead of 2:

$$x = x_1 \oplus x_2 \oplus \cdots \oplus x_n$$

- Any subset of $n-1$ shares is uniformly and independently distributed
  - If we probe at most $n-1$ shares x_i, we learn nothing about x
  - $\Rightarrow$ secure against a DPA attack of order $n-1$.

Higher-order masking

- Solution: n shares instead of 2:

$$x = x_1 \oplus x_2 \oplus \cdots \oplus x_n$$

- Any subset of $n-1$ shares is uniformly and independently distributed
  - If we probe at most $n-1$ shares x_i, we learn nothing about x
  - $\Rightarrow$ secure against a DPA attack of order $n-1$.

Higher-order masking

- High-order Boolean masking:

$$x = x_1 \oplus x_2 \oplus \cdots \oplus x_n$$

- High-order arithmetic masking:

$$x = A_1 + A_2 + \ldots + A_n \bmod 2^k$$

- Problem: how do we convert between Boolean and arithmetic masking ?
- **This talk:** efficient and provably secure high-order conversion from Boolean to arithmetic masking.

Prior work and this talk

n: number of shares

k: arithmetic modulo 2^k ($k = 32$ for HMAC-SHA-1).

	Direction	First-order complexity	High-order complexity
Goubin's algorithm [Gou01]	B $\rightarrow$ A A $\rightarrow$ B	$\mathcal{O}(1)$ $\mathcal{O}(k)$	- -
[CGV14]	B $\rightarrow$ A A $\rightarrow$ B	-	$\mathcal{O}(n^2 \cdot k)$
[CGTV15]	B $\rightarrow$ A A $\rightarrow$ B	- $\mathcal{O}(\log k)$	$\mathcal{O}(n^2 \cdot \log k)$
This talk	**B $\rightarrow$ A**	-	$\mathcal{O}(2^n)$

- **This talk**: complexity of Boolean to arithmetic conversion independent of k, as in Goubin's first-order algorithm.
- $\mathcal{O}(2^n)$: exponential complexity, but one order of magnitude faster than prior work for small values of n !

Prior work and this talk

n: number of shares

k: arithmetic modulo 2^k ($k = 32$ for HMAC-SHA-1).

	Direction	First-order complexity	High-order complexity
Goubin's algorithm [Gou01]	B $\rightarrow$ A A $\rightarrow$ B	$\mathcal{O}(1)$ $\mathcal{O}(k)$	- -
[CGV14]	B $\rightarrow$ A A $\rightarrow$ B	-	$\mathcal{O}(n^2 \cdot k)$
[CGTV15]	B $\rightarrow$ A A $\rightarrow$ B	- $\mathcal{O}(\log k)$	$\mathcal{O}(n^2 \cdot \log k)$
This talk	**B $\rightarrow$ A**	-	$\mathcal{O}(2^n)$

- **This talk**: complexity of Boolean to arithmetic conversion independent of k, as in Goubin's first-order algorithm.
- $\mathcal{O}(2^n)$: exponential complexity, but one order of magnitude faster than prior work for small values of n !

Prior work and this talk

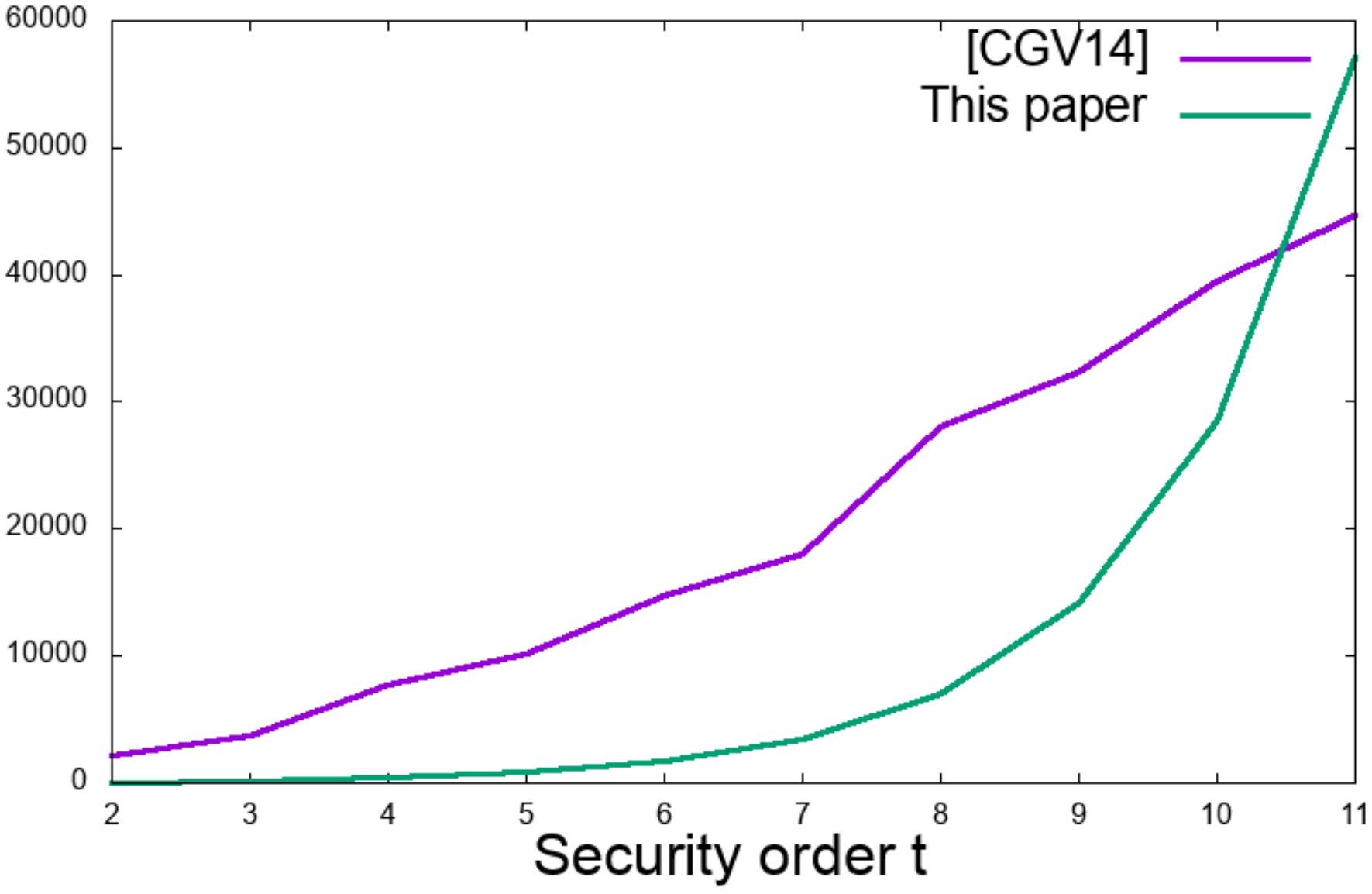
n: number of shares

k: arithmetic modulo 2^k ($k = 32$ for HMAC-SHA-1).

	Direction	First-order complexity	High-order complexity
Goubin's algorithm [Gou01]	B $\rightarrow$ A A $\rightarrow$ B	$\mathcal{O}(1)$ $\mathcal{O}(k)$	- -
[CGV14]	B $\rightarrow$ A A $\rightarrow$ B	-	$\mathcal{O}(n^2 \cdot k)$
[CGTV15]	B $\rightarrow$ A A $\rightarrow$ B	- $\mathcal{O}(\log k)$	$\mathcal{O}(n^2 \cdot \log k)$
This talk	**B $\rightarrow$ A**	-	$\mathcal{O}(2^n)$

- **This talk**: complexity of Boolean to arithmetic conversion independent of k, as in Goubin's first-order algorithm.
- $\mathcal{O}(2^n)$: exponential complexity, but one order of magnitude faster than prior work for small values of n !

Comparison with prior work ($k = 32$ bits)



Our contribution

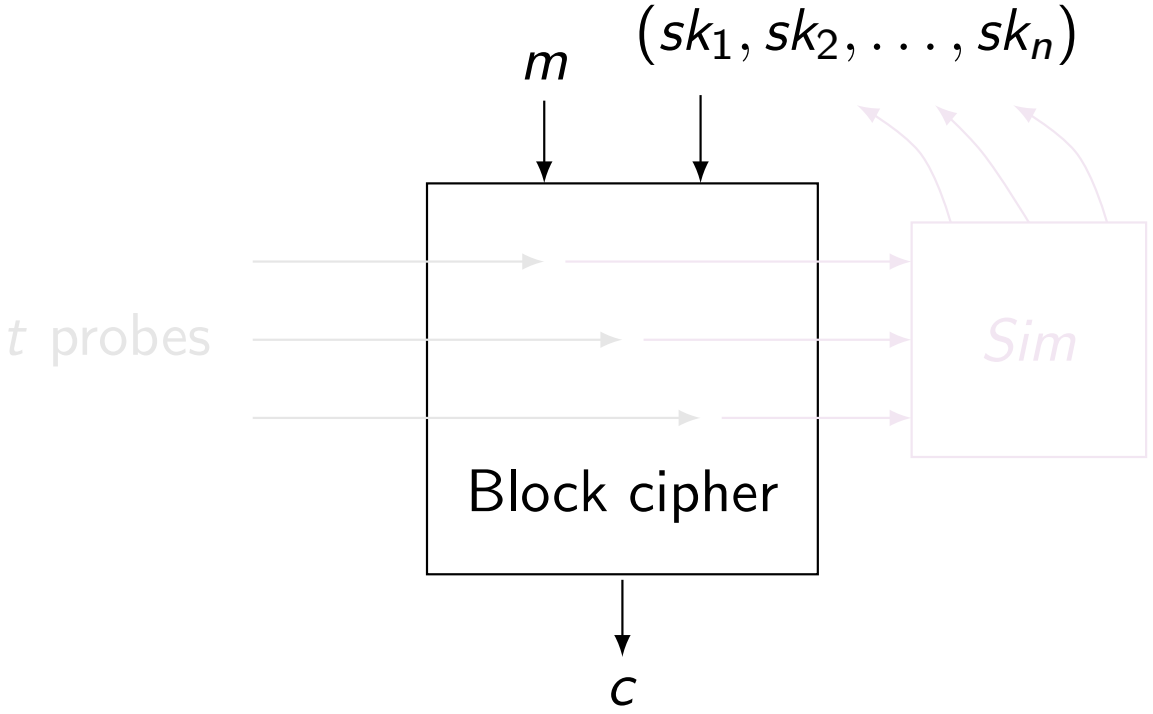
- Our contribution: high-order conversion algorithm from Boolean to arithmetic masking
- with a proof of security in the ISW probing model.
- Our algorithm is a variant of an algorithm published on ePrint by Hutter and Tunstall [HT16]
  - but no proof of security against high-order attacks was provided by the authors.
  - We describe an attack in the proceedings: 3rd order attack that works for any number of shares n.

Our contribution

- Our contribution: high-order conversion algorithm from Boolean to arithmetic masking
- with a proof of security in the ISW probing model.
- Our algorithm is a variant of an algorithm published on ePrint by Hutter and Tunstall [HT16]
  - but no proof of security against high-order attacks was provided by the authors.
  - We describe an attack in the proceedings: 3rd order attack that works for any number of shares n.

ISW security model

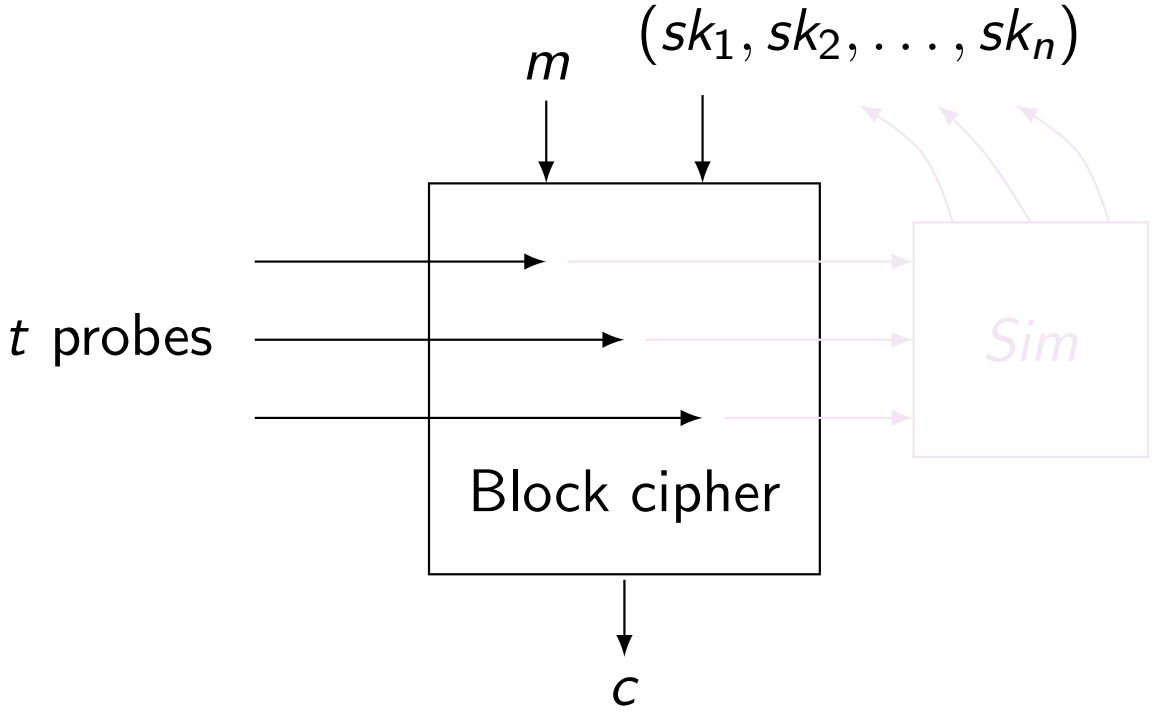
- Simulation framework of [ISW03]:



- Show that any t probes can be perfectly simulated from at most $n-1$ of the sk_i's.
- Those $n-1$ shares sk_i are initially uniformly and independently distributed.
- $\Rightarrow$ the adversary learns nothing from the t probes, since he could perfectly simulate those t probes by himself.

ISW security model

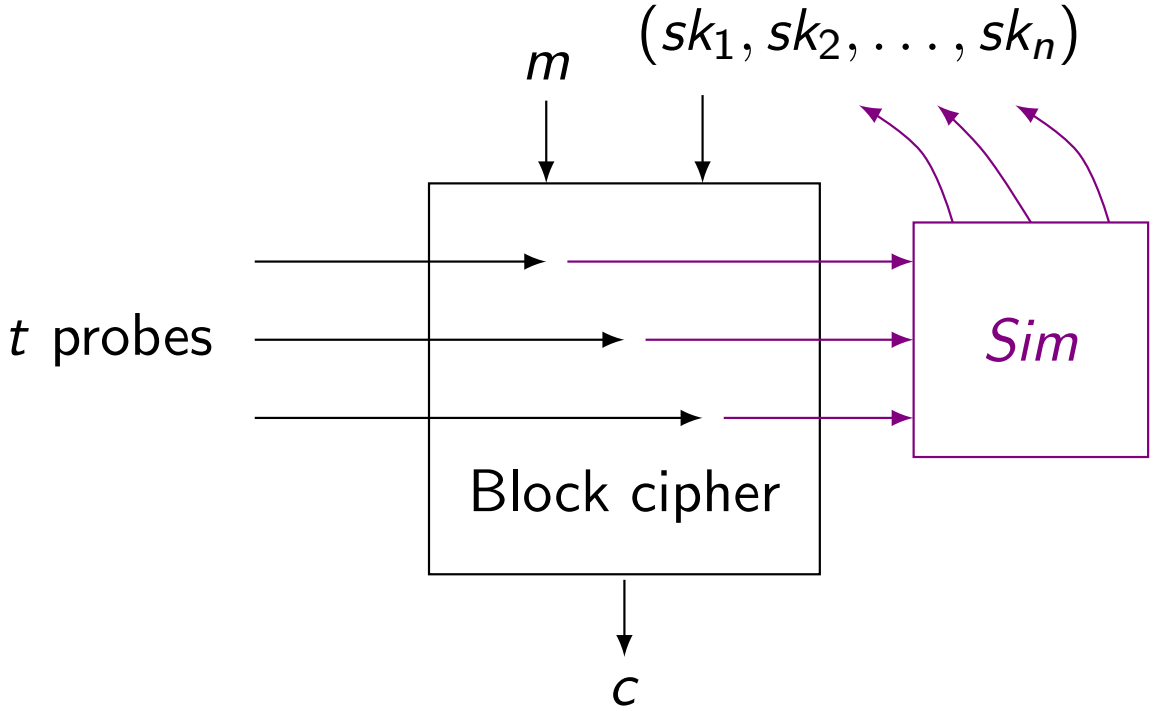
- Simulation framework of [ISW03]:



- Show that any t probes can be perfectly simulated from at most $n-1$ of the sk_i's.
- Those $n-1$ shares sk_i are initially uniformly and independently distributed.
- $\Rightarrow$ the adversary learns nothing from the t probes, since he could perfectly simulate those t probes by himself.

ISW security model

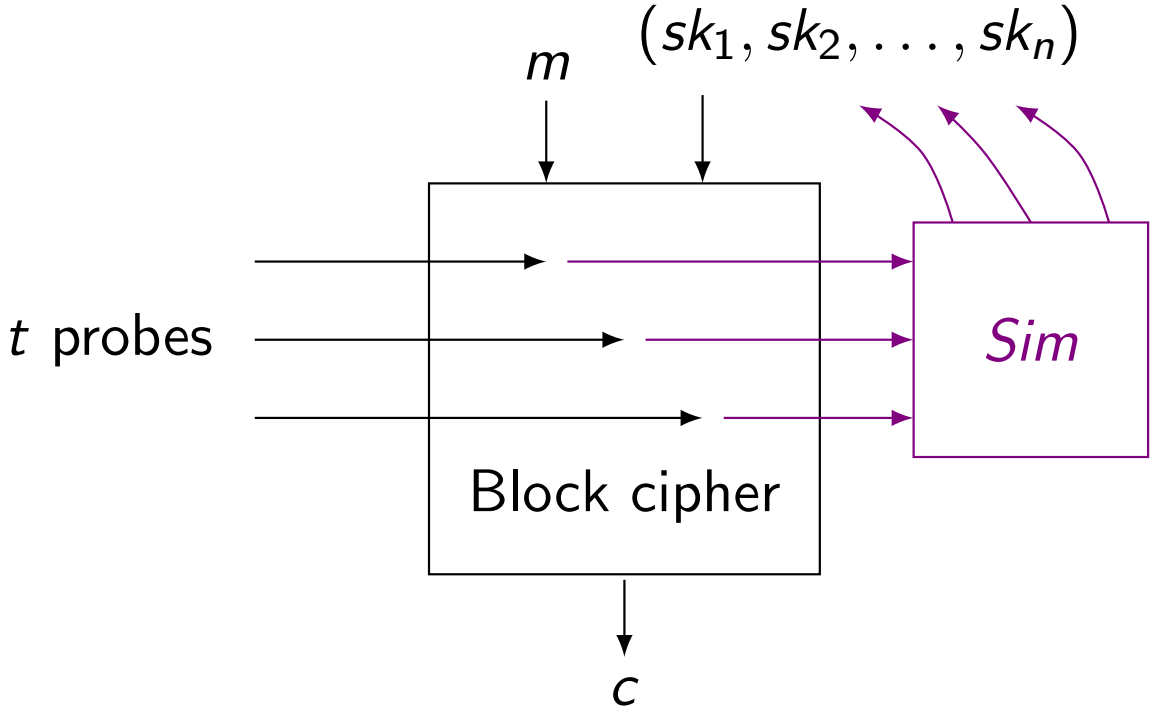
- Simulation framework of [ISW03]:



- Show that any t probes can be perfectly simulated from at most $n-1$ of the sk_i's.
- Those $n-1$ shares sk_i are initially uniformly and independently distributed.
- $\Rightarrow$ the adversary learns nothing from the t probes, since he could perfectly simulate those t probes by himself.

ISW security model

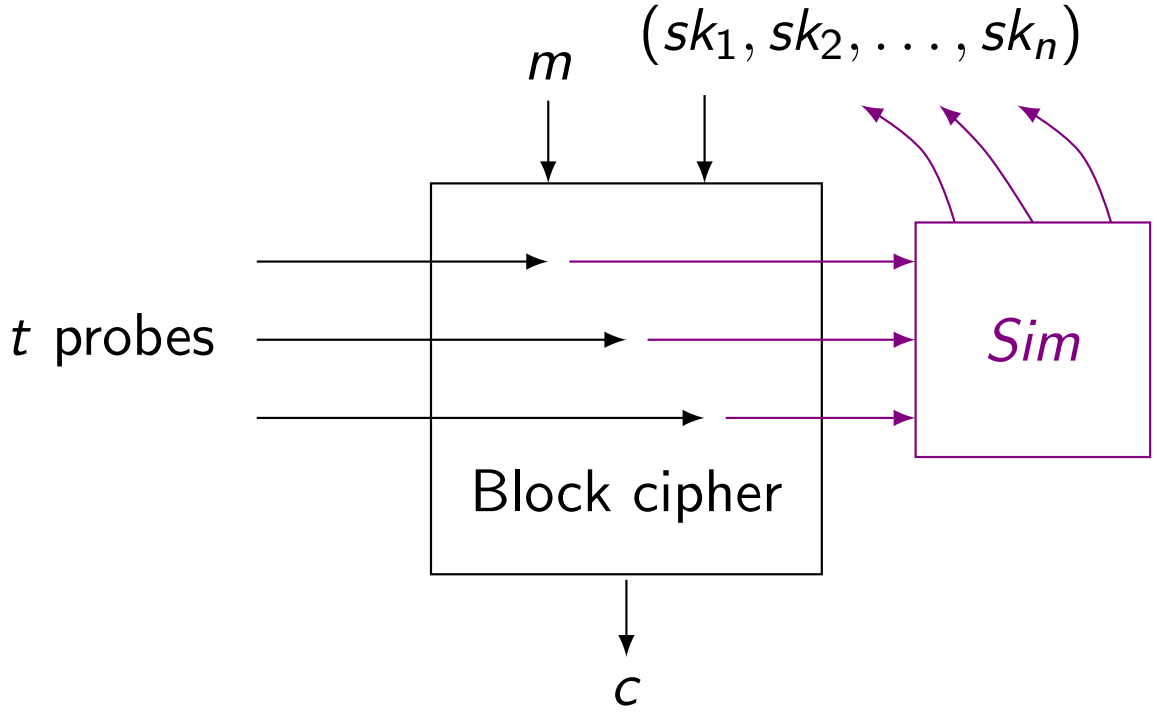
- Simulation framework of [ISW03]:



  - Show that any t probes can be perfectly simulated from at most $n-1$ of the sk_i's.
  - Those $n-1$ shares sk_i are initially uniformly and independently distributed.
  - $\Rightarrow$ the adversary learns nothing from the t probes, since he could perfectly simulate those t probes by himself.

ISW security model

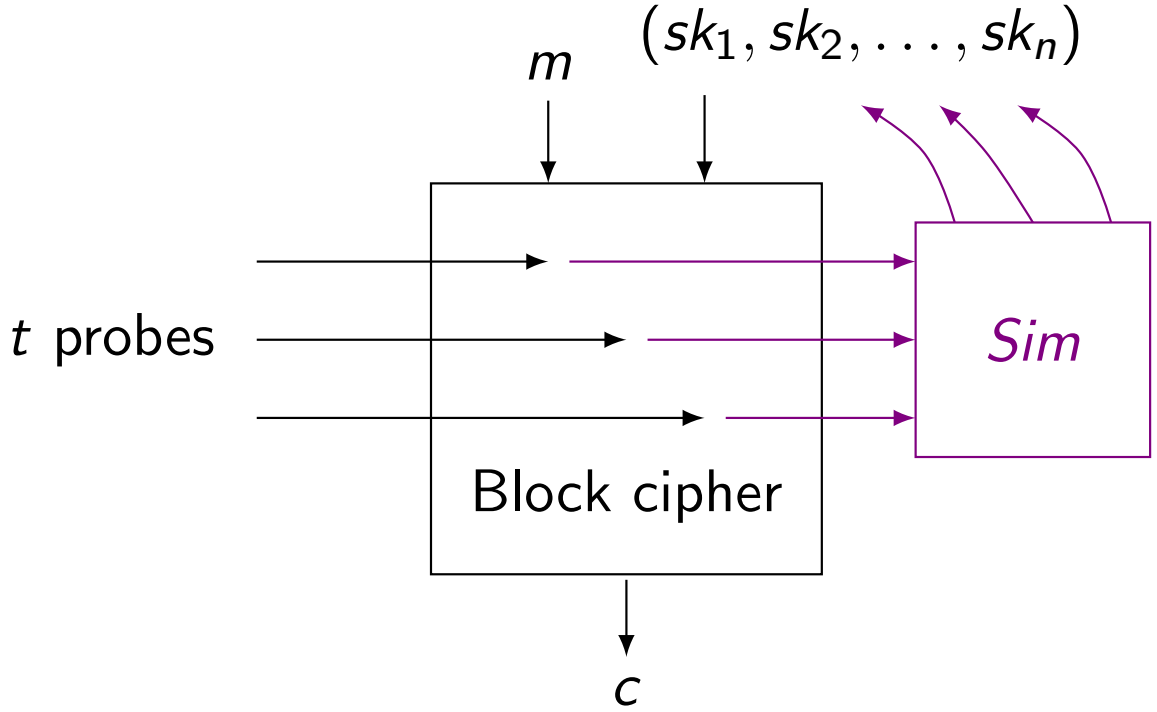
- Simulation framework of [ISW03]:



  - Show that any t probes can be perfectly simulated from at most $n-1$ of the sk_i's.
  - Those $n-1$ shares sk_i are initially uniformly and independently distributed.
  - $\Rightarrow$ the adversary learns nothing from the t probes, since he could perfectly simulate those t probes by himself.

ISW security model

- Simulation framework of [ISW03]:



  - Show that any t probes can be perfectly simulated from at most $n-1$ of the sk_i's.
  - Those $n-1$ shares sk_i are initially uniformly and independently distributed.
  - $\Rightarrow$ the adversary learns nothing from the t probes, since he could perfectly simulate those t probes by himself.

Security proofs for side-channel countermeasures

- Never publish a high-order masking scheme without a proof of security !
  - So many things can go wrong.
  - Many countermeasures without proofs have been broken in the past.
  - We have a poor intuition of high-order security.

Goubin's original conversion algorithm

- Goubin's theorem: the function

$$\Psi(x, r) = (x \oplus r) - r \pmod{2^k}$$

  is affine with respect to r over $\mathbb{F}_2$.
  - This is surprising but true !

- Goubin's Boolean to arithmetic conversion algorithm:

$$\begin{aligned} x &= x_1 \oplus x_2 \\ &= (x_1 \oplus x_2 - x_2) + x_2 \\ &= \Psi(x_1, x_2) + x_2 \\ &= \left[\left(x_1 \oplus \Psi(x_1, r \oplus x_2) \right) \oplus \Psi(x_1, r) \right] + x_2 \\ &= A + x_2 \pmod{2^k} \end{aligned}$$

- One can compute A without leaking information about x, thanks to the random r.

Goubin's original conversion algorithm

- Goubin's theorem: the function

$$\Psi(x, r) = (x \oplus r) - r \pmod{2^k}$$

  is affine with respect to r over $\mathbb{F}_2$.
  - This is surprising but true !

- Goubin's Boolean to arithmetic conversion algorithm:

$$\begin{aligned} x &= x_1 \oplus x_2 \\ &= (x_1 \oplus x_2 - x_2) + x_2 \\ &= \Psi(x_1, x_2) + x_2 \\ &= \left[(x_1 \oplus \Psi(x_1, r \oplus x_2)) \oplus \Psi(x_1, r)\right] + x_2 \\ &= A + x_2 \pmod{2^k} \end{aligned}$$

- One can compute A without leaking information about x, thanks to the random r.

Goubin's original conversion algorithm

- Goubin's theorem: the function

$$\Psi(x, r) = (x \oplus r) - r \pmod{2^k}$$

  is affine with respect to r over $\mathbb{F}_2$.
  - This is surprising but true !

- Goubin's Boolean to arithmetic conversion algorithm:

$$\begin{aligned} x &= x_1 \oplus x_2 \\ &= (x_1 \oplus x_2 - x_2) + x_2 \\ &= \Psi(x_1, x_2) + x_2 \\ &= \left[\left(x_1 \oplus \Psi(x_1, r \oplus x_2)\right) \oplus \Psi(x_1, r)\right] + x_2 \\ &= A + x_2 \pmod{2^k} \end{aligned}$$

- One can compute A without leaking information about x, thanks to the random r.

Goubin's original conversion algorithm

- Goubin's theorem: the function

$$\Psi(x, r) = (x \oplus r) - r \pmod{2^k}$$

  is affine with respect to r over $\mathbb{F}_2$.
  - This is surprising but true !

- Goubin's Boolean to arithmetic conversion algorithm:

$$\begin{aligned} x &= x_1 \oplus x_2 \\ &= (x_1 \oplus x_2 - x_2) + x_2 \\ &= \Psi(x_1, x_2) + x_2 \\ &= \left[\left(x_1 \oplus \Psi(x_1, r \oplus x_2)\right) \oplus \Psi(x_1, r)\right] + x_2 \\ &= A + x_2 \pmod{2^k} \end{aligned}$$

- One can compute A without leaking information about x, thanks to the random r.

Goubin's original conversion algorithm

- Goubin's theorem: the function

$$\Psi(x, r) = (x \oplus r) - r \pmod{2^k}$$

  is affine with respect to r over $\mathbb{F}_2$.
  - This is surprising but true !

- Goubin's Boolean to arithmetic conversion algorithm:

$$\begin{aligned} x &= x_1 \oplus x_2 \\ &= (x_1 \oplus x_2 - x_2) + x_2 \\ &= \Psi(x_1, x_2) + x_2 \\ &= \left[\left(x_1 \oplus \Psi(x_1, r \oplus x_2)\right) \oplus \Psi(x_1, r)\right] + x_2 \\ &= A + x_2 \pmod{2^k} \end{aligned}$$

- One can compute A without leaking information about x, thanks to the random r.

Goubin's original conversion algorithm

- Goubin's theorem: the function

$$\Psi(x, r) = (x \oplus r) - r \pmod{2^k}$$

  is affine with respect to r over $\mathbb{F}_2$.
  - This is surprising but true !

- Goubin's Boolean to arithmetic conversion algorithm:

$$\begin{aligned} x &= x_1 \oplus x_2 \\ &= (x_1 \oplus x_2 - x_2) + x_2 \\ &= \Psi(x_1, x_2) + x_2 \\ &= \left[\left(x_1 \oplus \Psi(x_1, r \oplus x_2)\right) \oplus \Psi(x_1, r)\right] + x_2 \\ &= A + x_2 \pmod{2^k} \end{aligned}$$

- One can compute A without leaking information about x, thanks to the random r.

Goubin's original conversion algorithm

- Goubin's theorem: the function

$$\Psi(x, r) = (x \oplus r) - r \pmod{2^k}$$

  is affine with respect to r over $\mathbb{F}_2$.
  - This is surprising but true !

- Goubin's Boolean to arithmetic conversion algorithm:

$$\begin{aligned} x &= x_1 \oplus x_2 \\ &= (x_1 \oplus x_2 - x_2) + x_2 \\ &= \Psi(x_1, x_2) + x_2 \\ &= \left[\left(x_1 \oplus \Psi(x_1, r \oplus x_2)\right) \oplus \Psi(x_1, r)\right] + x_2 \\ &= A + x_2 \pmod{2^k} \end{aligned}$$

- One can compute A without leaking information about x, thanks to the random r.

Our new algorithm: generalization of Goubin

- We start from

$$
\begin{aligned}
x &= x_1 \oplus \cdots \oplus x_n \\
&= (x_1 \oplus x_2 \oplus \cdots \oplus x_n - x_2 \oplus \cdots \oplus x_n) + x_2 \oplus \cdots \oplus x_n \\
&= \Psi(x_1, x_2 \oplus \cdots \oplus x_n) + x_2 \oplus \cdots \oplus x_n \\
&= (n \wedge 1) \cdot x_1 \oplus \Psi(x_1, x_2) \oplus \cdots \oplus \Psi(x_1, x_n) + x_2 \oplus \cdots \oplus x_n \\
&= z_1 \oplus \cdots \oplus z_{n-1} + x_2 \oplus \cdots \oplus x_n
\end{aligned}
$$

- We can apply the algorithm recursively on both terms with $n-1$ shares:

$$
\begin{aligned}
x &= A_1 + \cdots + A_{n-1} + B_1 + \cdots + B_{n-1} \\
&= (A_1 + B_1) + \cdots + (A_{n-2} + B_{n-2}) + A_{n-1} + B_{n-1} \\
&= D_1 + \cdots + D_{n-2} + D_{n-1} + D_n
\end{aligned}
$$

- We obtain n arithmetic shares as required.

Our new algorithm: generalization of Goubin

- We start from

$$
\begin{aligned}
x &= x_1 \oplus \cdots \oplus x_n \\
&= (x_1 \oplus x_2 \oplus \cdots \oplus x_n - x_2 \oplus \cdots \oplus x_n) + x_2 \oplus \cdots \oplus x_n \\
&= \Psi(x_1, x_2 \oplus \cdots \oplus x_n) + x_2 \oplus \cdots \oplus x_n \\
&= (n \wedge 1) \cdot x_1 \oplus \Psi(x_1, x_2) \oplus \cdots \oplus \Psi(x_1, x_n) + x_2 \oplus \cdots \oplus x_n \\
&= z_1 \oplus \cdots \oplus z_{n-1} + x_2 \oplus \cdots \oplus x_n
\end{aligned}
$$

- We can apply the algorithm recursively on both terms with $n-1$ shares:

$$
\begin{aligned}
x &= A_1 + \cdots + A_{n-1} + B_1 + \cdots + B_{n-1} \\
&= (A_1 + B_1) + \cdots + (A_{n-2} + B_{n-2}) + A_{n-1} + B_{n-1} \\
&= D_1 + \cdots + D_{n-2} + D_{n-1} + D_n
\end{aligned}
$$

- We obtain n arithmetic shares as required.

Our new algorithm: generalization of Goubin

- We start from

$$\begin{aligned} x &= x_1 \oplus \cdots \oplus x_n \\ &= (x_1 \oplus x_2 \oplus \cdots \oplus x_n - x_2 \oplus \cdots \oplus x_n) + x_2 \oplus \cdots \oplus x_n \\ &= \Psi(x_1, x_2 \oplus \cdots \oplus x_n) + x_2 \oplus \cdots \oplus x_n \\ &= (n \wedge 1) \cdot x_1 \oplus \Psi(x_1, x_2) \oplus \cdots \oplus \Psi(x_1, x_n) + x_2 \oplus \cdots \oplus x_n \\ &= z_1 \oplus \cdots \oplus z_{n-1} + x_2 \oplus \cdots \oplus x_n \end{aligned}$$

- We can apply the algorithm recursively on both terms with $n-1$ shares:

$$\begin{aligned} x &= A_1 + \cdots + A_{n-1} + B_1 + \cdots + B_{n-1} \\ &= (A_1 + B_1) + \cdots + (A_{n-2} + B_{n-2}) + A_{n-1} + B_{n-1} \\ &= D_1 + \cdots + D_{n-2} + D_{n-1} + D_n \end{aligned}$$

- We obtain n arithmetic shares as required.

Our new algorithm: generalization of Goubin

- We start from

$$\begin{aligned} x &= x_1 \oplus \cdots \oplus x_n \\ &= (x_1 \oplus x_2 \oplus \cdots \oplus x_n - x_2 \oplus \cdots \oplus x_n) + x_2 \oplus \cdots \oplus x_n \\ &= \Psi(x_1, x_2 \oplus \cdots \oplus x_n) + x_2 \oplus \cdots \oplus x_n \\ &= (n \wedge 1) \cdot x_1 \oplus \Psi(x_1, x_2) \oplus \cdots \oplus \Psi(x_1, x_n) + x_2 \oplus \cdots \oplus x_n \\ &= z_1 \oplus \cdots \oplus z_{n-1} + x_2 \oplus \cdots \oplus x_n \end{aligned}$$

- We can apply the algorithm recursively on both terms with $n-1$ shares:

$$\begin{aligned} x &= A_1 + \cdots + A_{n-1} + B_1 + \cdots + B_{n-1} \\ &= (A_1 + B_1) + \cdots + (A_{n-2} + B_{n-2}) + A_{n-1} + B_{n-1} \\ &= D_1 + \cdots + D_{n-2} + D_{n-1} + D_n \end{aligned}$$

- We obtain n arithmetic shares as required.

Our new algorithm: generalization of Goubin

- We start from

$$
\begin{aligned}
x &= x_1 \oplus \cdots \oplus x_n \\
&= (x_1 \oplus x_2 \oplus \cdots \oplus x_n - x_2 \oplus \cdots \oplus x_n) + x_2 \oplus \cdots \oplus x_n \\
&= \Psi(x_1, x_2 \oplus \cdots \oplus x_n) + x_2 \oplus \cdots \oplus x_n \\
&= (n \wedge 1) \cdot x_1 \oplus \Psi(x_1, x_2) \oplus \cdots \oplus \Psi(x_1, x_n) + x_2 \oplus \cdots \oplus x_n \\
&= z_1 \oplus \cdots \oplus z_{n-1} + x_2 \oplus \cdots \oplus x_n
\end{aligned}
$$

- We can apply the algorithm recursively on both terms with $n-1$ shares:

$$
\begin{aligned}
x &= A_1 + \cdots + A_{n-1} + B_1 + \cdots + B_{n-1} \\
&= (A_1 + B_1) + \cdots + (A_{n-2} + B_{n-2}) + A_{n-1} + B_{n-1} \\
&= D_1 + \cdots + D_{n-2} + D_{n-1} + D_n
\end{aligned}
$$

- We obtain n arithmetic shares as required.

Our new algorithm: generalization of Goubin

- We start from

$$\begin{aligned} x &= x_1 \oplus \cdots \oplus x_n \\ &= (x_1 \oplus x_2 \oplus \cdots \oplus x_n - x_2 \oplus \cdots \oplus x_n) + x_2 \oplus \cdots \oplus x_n \\ &= \Psi(x_1, x_2 \oplus \cdots \oplus x_n) + x_2 \oplus \cdots \oplus x_n \\ &= (n \wedge 1) \cdot x_1 \oplus \Psi(x_1, x_2) \oplus \cdots \oplus \Psi(x_1, x_n) + x_2 \oplus \cdots \oplus x_n \\ &= z_1 \oplus \cdots \oplus z_{n-1} + x_2 \oplus \cdots \oplus x_n \end{aligned}$$

- We can apply the algorithm recursively on both terms with $n-1$ shares:

$$\begin{aligned} x &= A_1 + \cdots + A_{n-1} + B_1 + \cdots + B_{n-1} \\ &= (A_1 + B_1) + \cdots + (A_{n-2} + B_{n-2}) + A_{n-1} + B_{n-1} \\ &= D_1 + \cdots + D_{n-2} + D_{n-1} + D_n \end{aligned}$$

- We obtain n arithmetic shares as required.

Our new algorithm: generalization of Goubin

- We start from

$$\begin{aligned} x &= x_1 \oplus \cdots \oplus x_n \\ &= (x_1 \oplus x_2 \oplus \cdots \oplus x_n - x_2 \oplus \cdots \oplus x_n) + x_2 \oplus \cdots \oplus x_n \\ &= \Psi(x_1, x_2 \oplus \cdots \oplus x_n) + x_2 \oplus \cdots \oplus x_n \\ &= (n \wedge 1) \cdot x_1 \oplus \Psi(x_1, x_2) \oplus \cdots \oplus \Psi(x_1, x_n) + x_2 \oplus \cdots \oplus x_n \\ &= z_1 \oplus \cdots \oplus z_{n-1} + x_2 \oplus \cdots \oplus x_n \end{aligned}$$

- We can apply the algorithm recursively on both terms with $n-1$ shares:

$$\begin{aligned} x &= A_1 + \cdots + A_{n-1} + B_1 + \cdots + B_{n-1} \\ &= (A_1 + B_1) + \cdots + (A_{n-2} + B_{n-2}) + A_{n-1} + B_{n-1} \\ &= D_1 + \cdots + D_{n-2} + D_{n-1} + D_n \end{aligned}$$

- We obtain n arithmetic shares as required.

Our new algorithm: generalization of Goubin

- We start from

$$\begin{aligned} x &= x_1 \oplus \cdots \oplus x_n \\ &= (x_1 \oplus x_2 \oplus \cdots \oplus x_n - x_2 \oplus \cdots \oplus x_n) + x_2 \oplus \cdots \oplus x_n \\ &= \Psi(x_1, x_2 \oplus \cdots \oplus x_n) + x_2 \oplus \cdots \oplus x_n \\ &= (n \wedge 1) \cdot x_1 \oplus \Psi(x_1, x_2) \oplus \cdots \oplus \Psi(x_1, x_n) + x_2 \oplus \cdots \oplus x_n \\ &= z_1 \oplus \cdots \oplus z_{n-1} + x_2 \oplus \cdots \oplus x_n \end{aligned}$$

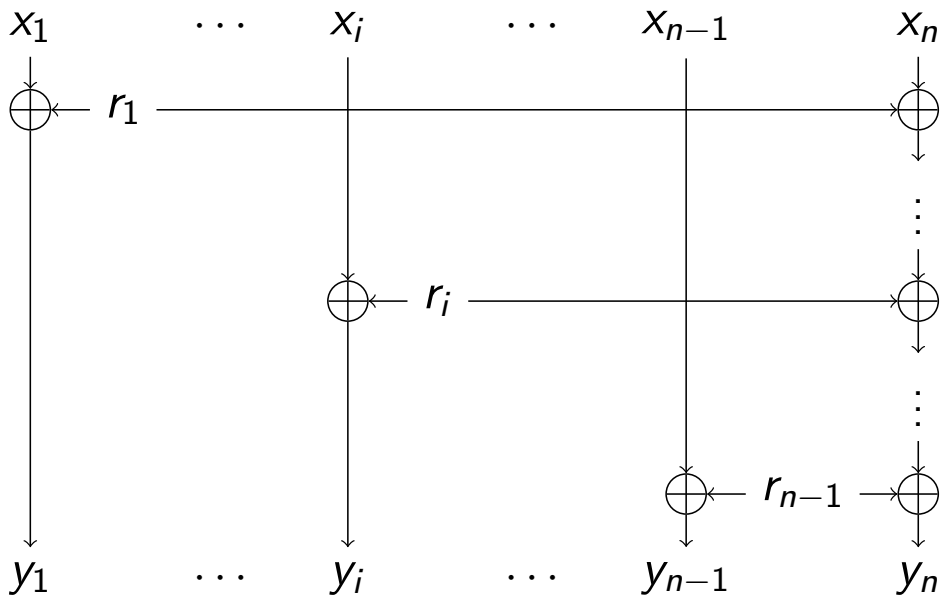
- We can apply the algorithm recursively on both terms with $n-1$ shares:

$$\begin{aligned} x &= A_1 + \cdots + A_{n-1} + B_1 + \cdots + B_{n-1} \\ &= (A_1 + B_1) + \cdots + (A_{n-2} + B_{n-2}) + A_{n-1} + B_{n-1} \\ &= D_1 + \cdots + D_{n-2} + D_{n-1} + D_n \end{aligned}$$

- We obtain n arithmetic shares as required.

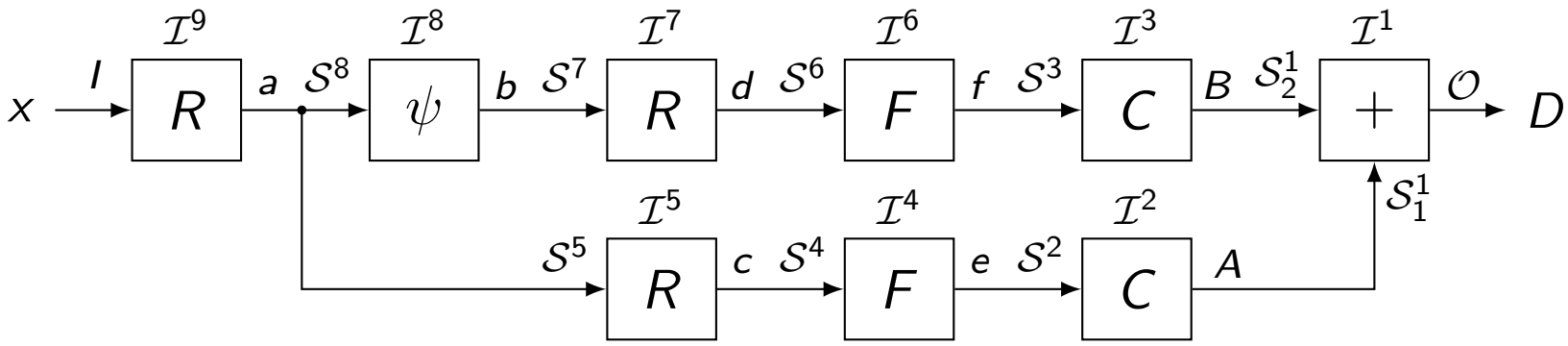
Our new algorithm

- We must add some intermediate mask refreshing, otherwise the algorithm would be insecure:



Proof of Security in the ISW probing model

- We use the t-NI and t-SNI security definitions introduced by Barthe *et al.* in [BBD+16]
  - This enables to have a modular proof
  - We first analyse each gadget separately
  - We then compose the gadgets



- See the proof in the ePrint version of the paper.

Operation count

- Operation count for Boolean to arithmetic conversion algorithms, with $n = t + 1$ shares.

B $\rightarrow$ A conversion	Security order t							
	1	2	3	4	6	8	10	12
Goubin [Gou01]	7							
Hutter-Tunstall [HT16]		31						
CGV, 32 bits [CGV14]		2 098	3 664	7 752	14 698	28 044	39 518	56 344
Our algorithm		55	155	367	1 687	7 039	28 519	114 511

- For small orders t, our algorithm is one order of magnitude more efficient than [CGV14].

Conclusion

- We have described a new high-order Boolean to arithmetic conversion algorithm.
  - Extension of Goubin's first-order algorithm
  - Provably secure in the ISW probing model
- Complexity: $\mathcal{O}(2^n)$ for n shares, independent of the register size k.
  - Instead of $\mathcal{O}(n^2 \cdot k)$ in [CGV14]
  - but one order of magnitude faster for small n
- Open problem: can we do better than $\mathcal{O}(2^n)$?

Conclusion

- We have described a new high-order Boolean to arithmetic conversion algorithm.
  - Extension of Goubin's first-order algorithm
  - Provably secure in the ISW probing model
- Complexity: $\mathcal{O}(2^n)$ for n shares, independent of the register size k.
  - Instead of $\mathcal{O}(n^2 \cdot k)$ in [CGV14]
  - but one order of magnitude faster for small n
- Open problem: can we do better than $\mathcal{O}(2^n)$?

Conclusion

- We have described a new high-order Boolean to arithmetic conversion algorithm.
  - Extension of Goubin's first-order algorithm
  - Provably secure in the ISW probing model
- Complexity: $\mathcal{O}(2^n)$ for n shares, independent of the register size k.
  - Instead of $\mathcal{O}(n^2 \cdot k)$ in [CGV14]
  - but one order of magnitude faster for small n
- Open problem: can we do better than $\mathcal{O}(2^n)$?