Improved High-Order Conversion From Boolean to Arithmetic Masking

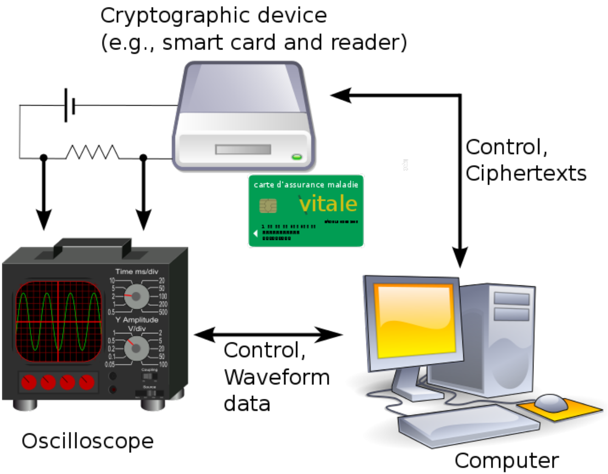
Luk Bettale[1] Jean-Sébastien Coron[2] Rina Zeitoun[1]

[1] IDEMIA, France

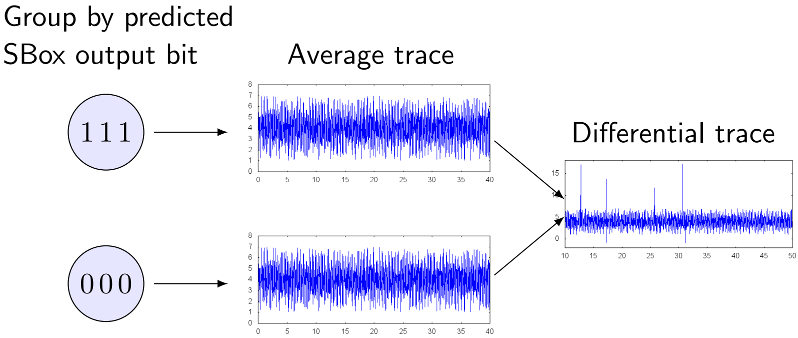
[2] University of Luxembourg

CHES 2018

Side-channel Attacks



Differential Power Analysis [KJJ99]



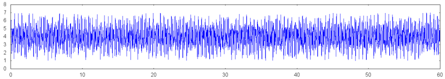
Masking Countermeasure

- 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

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

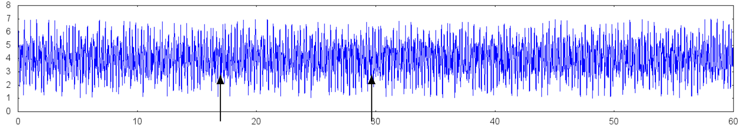
- Some algorithms use arithmetic operations, for example IDEA, RC6, XTEA, SPECK, SHA-1.
- For these algorithms, we can use arithmetic masking:

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

  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:** high-order Boolean to arithmetic conversion algorithm, simpler and more efficient than [Cor17].
  - complexity independent of the register size k
  - still with a proof of security in the ISW probing model

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:** high-order Boolean to arithmetic conversion algorithm, simpler and more efficient than [Cor17].
  - complexity independent of the register size k
  - still with a proof of security in the ISW probing model

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	$\mathcal{O}(1)$	-
	A $\rightarrow$ B	$\mathcal{O}(k)$	-
[CGV14]	B $\rightarrow$ A	-	$\mathcal{O}(n^2 \cdot k)$
	A $\rightarrow$ B		
[CGTV15]	B $\rightarrow$ A	-	$\mathcal{O}(n^2 \cdot \log k)$
	A $\rightarrow$ B	$\mathcal{O}(\log k)$	
[Cor17]	B $\rightarrow$ A	-	$14 \cdot 2^n + \mathcal{O}(n)$
This talk	**B $\rightarrow$ A**	-	$10 \cdot 2^n + \mathcal{O}(n)$

- Complexity independent of the register size k, as in [Cor17]
- Exponential complexity, but one order of magnitude faster than [CGV14] and [CGTV15] for small values of n.

Prior work and this talk

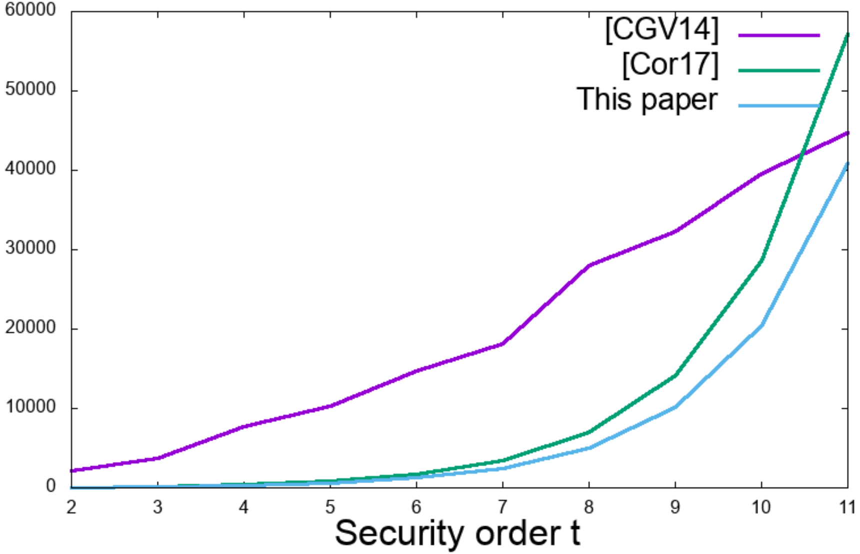
n: number of shares

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	$\mathcal{O}(1)$	-
	A $\rightarrow$ B	$\mathcal{O}(k)$	-
[CGV14]	B $\rightarrow$ A A $\rightarrow$ B	-	$\mathcal{O}(n^2 \cdot k)$
[CGTV15]	B $\rightarrow$ A	-	$\mathcal{O}(n^2 \cdot \log k)$
	A $\rightarrow$ B	$\mathcal{O}(\log k)$	
[Cor17]	B $\rightarrow$ A	-	$14 \cdot 2^n + \mathcal{O}(n)$
This talk	**B $\rightarrow$ A**	-	$10 \cdot 2^n + \mathcal{O}(n)$

- Complexity independent of the register size k, as in [Cor17]
- Exponential complexity, but one order of magnitude faster than [CGV14] and [CGTV15] for small values of n.

Boolean to arithmetic conversion: comparison with prior work ($k = 32$ bits)



Comparison with CHES 2017 algorithm

[Cor17]	B $\rightarrow$ A	-	$14 \cdot 2^n + \mathcal{O}(n)$
This talk	**B $\rightarrow$ A**	-	$10 \cdot 2^n + \mathcal{O}(n)$

- Our new algorithm is roughly 25% faster, and simpler.

[Cor17]

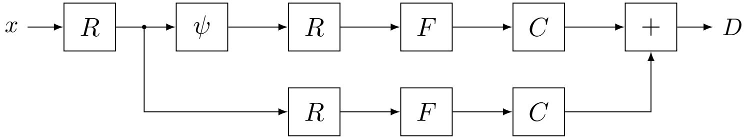
This talk

Comparison with CHES 2017 algorithm

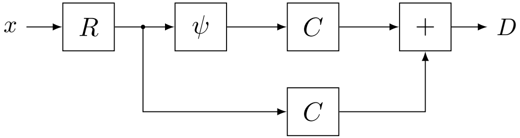
[Cor17]	B $\rightarrow$ A	-	$14 \cdot 2^n + \mathcal{O}(n)$
This talk	**B $\rightarrow$ A**	-	$10 \cdot 2^n + \mathcal{O}(n)$

- Our new algorithm is roughly 25% faster, and simpler.

[Cor17]



This talk



Our contribution

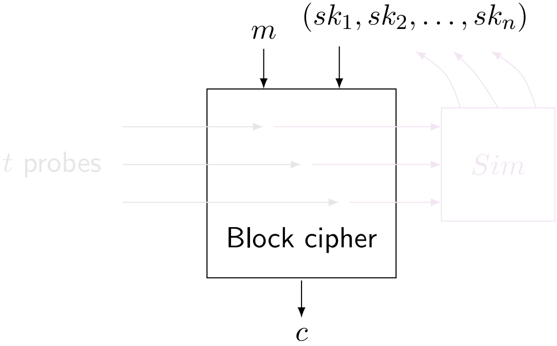
- Our contribution: high-order conversion algorithm from Boolean to arithmetic masking
  - simplified variant of CHES 2017 algorithm
  - still with a proof of security in the ISW probing model.
- Approach initiated by Hutter and Tunstall [HT16] (eprint)
  - but no proof of security against high-order attacks was provided by the authors.
  - 3rd order attack for any number of shares n described in [Cor17]
  - 3rd order attack against updated Hutter-Tunstall algorithm (see the proceedings)

Our contribution

- Our contribution: high-order conversion algorithm from Boolean to arithmetic masking
  - simplified variant of CHES 2017 algorithm
  - still with a proof of security in the ISW probing model.
- Approach initiated by Hutter and Tunstall [HT16] (eprint)
  - but no proof of security against high-order attacks was provided by the authors.
  - 3rd order attack for any number of shares n described in [Cor17]
  - 3rd order attack against updated Hutter-Tunstall algorithm (see the proceedings)

ISW security model

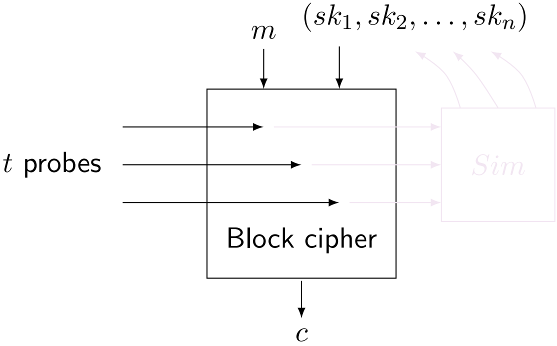
- Simulation framework of [ISW03]:



- 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

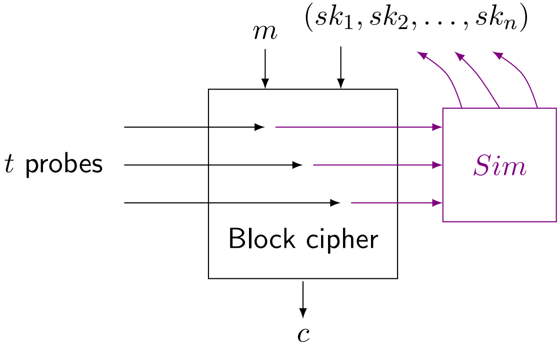
- Simulation framework of [ISW03]:



- 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

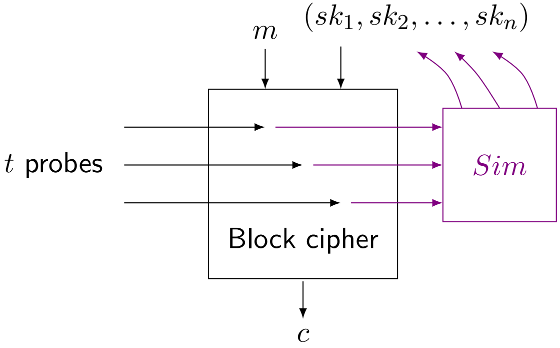
- Simulation framework of [ISW03]:



- 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

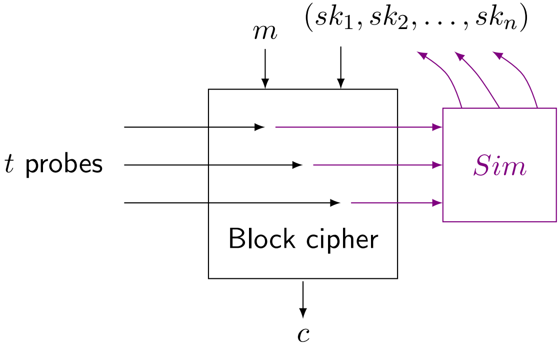
- Simulation framework of [ISW03]:



- 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

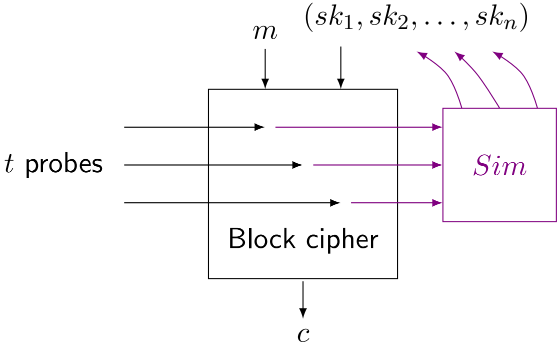
- Simulation framework of [ISW03]:



  - 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:

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

- 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:

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

- 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

- Our recursive algorithm takes $n+1$ input shares (instead of n):

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

- We can apply the algorithm recursively on both terms, from n Boolean shares to $n-1$ arithmetic 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

- Our recursive algorithm takes $n+1$ input shares (instead of n):

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

- We can apply the algorithm recursively on both terms, from n Boolean shares to $n-1$ arithmetic 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

- Our recursive algorithm takes $n+1$ input shares (instead of n):

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

- We can apply the algorithm recursively on both terms, from n Boolean shares to $n-1$ arithmetic 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

- Our recursive algorithm takes $n+1$ input shares (instead of n):

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

- We can apply the algorithm recursively on both terms, from n Boolean shares to $n-1$ arithmetic 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

- Our recursive algorithm takes $n+1$ input shares (instead of n):

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

- We can apply the algorithm recursively on both terms, from n Boolean shares to $n-1$ arithmetic 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

- Our recursive algorithm takes $n+1$ input shares (instead of n):

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

- We can apply the algorithm recursively on both terms, from n Boolean shares to $n-1$ arithmetic 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

- Our recursive algorithm takes $n+1$ input shares (instead of n):

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

- We can apply the algorithm recursively on both terms, from n Boolean shares to $n-1$ arithmetic 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

- Our recursive algorithm takes $n+1$ input shares (instead of n):

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

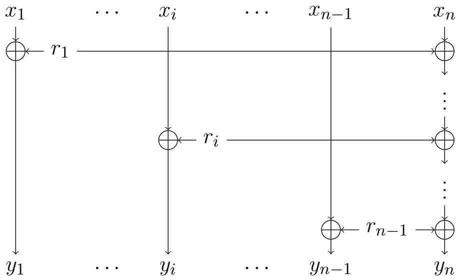
- We can apply the algorithm recursively on both terms, from n Boolean shares to $n-1$ arithmetic shares:

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

- We obtain n arithmetic shares as required.

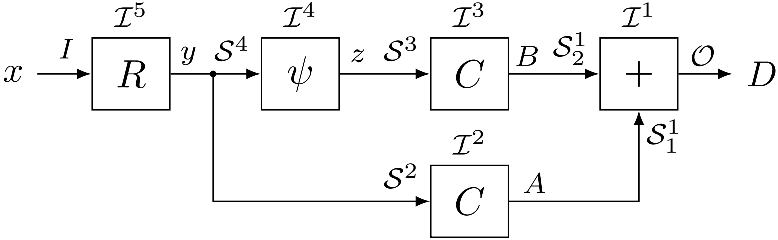
Our new algorithm

- 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
[Cor17]		55	155	367	1 687	7 039	28 519	114 511
Our algorithm		49	123	277	1 225	5 053	20 401	81 829

- For small orders t, [Cor17] and our algorithm are one order of magnitude more efficient than [CGV14].

Formal Verification

- We have formally verified the security of our countermeasure, using the CheckMasks tool [Cor18]
  - Generic verification of masking countermeasures, based on the Common Lisp language
  - Source code: `https://github.com/coron/checkmasks`
- Verification time:

n	#var.	#tuples	Security	Time
2	14	14	✓	ε
3	39	741	✓	0.06 s
4	94	134,044	✓	30 s
5	207	74,303,685	✓	12 h

Conclusion

- We have described a new high-order Boolean to arithmetic conversion algorithm.
  - Simplified variant of [Cor17], roughly 25% more efficient.
  - Provably secure in the ISW probing model
  - Formal verification up to $n = 5$
- 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.
  - Simplified variant of [Cor17], roughly 25% more efficient.
  - Provably secure in the ISW probing model
  - Formal verification up to $n = 5$
- 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.
  - Simplified variant of [Cor17], roughly 25% more efficient.
  - Provably secure in the ISW probing model
  - Formal verification up to $n = 5$
- 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 $\mathcal{O}(2^n)$?