Boolean/Arithmetic Conversions for any Order

Jean-Sébastien Coron

Abstract. At CHES 2001 Goubin presented a nice method for converting from boolean masking to arithmetic masking and conversely. However the method is only secure against first-order attacks. In this paper we provide a generalization to any order.

1 Introduction

Let x be a sensitive variable.

Boolean masking: $x = x' \oplus r$.

Arithmetic masking: $x = A + r \bmod 2^k$, where k is the register size.

2 From Boolean to Arithmetic Masking

It is shown in [1] that the following function is affine with respect to the Boolean operation:

$$\Psi_{x'}(r) = (x' \oplus r) - r \mod 2^k$$

Theorem 1 (Goubin [1]). *The function*

$$\Psi_{x'}(r) = (x' \oplus r) - r \mod 2^k$$

is affine over $GF(2)$.

Proof. The proof in [1] is based on the following equality.

$$\Psi_{x'}(r) = x' \oplus \bigoplus_{i=1}^{k-1} \left(\left(\bigwedge_{j=1}^{i-1} 2^j \bar{x'} \right) \wedge (2^i x') \wedge (2^i r) \right)$$

This equality is proved by induction in [1]; the affine property of $\Psi_{x'}(r)$ follows immediatly. However, it is difficult to understand intuitively why such equality holds. In this section we provide a hopefully more enlightening proof of the affine property of $\Psi_{x'}(r)$. To do. $\square$

The conversion from boolean to arithmetic masking is then straightforward. Given x', r such that $x = x' \oplus r$ we must compute A such that $x = A + r$, which gives:

$$A = (x' \oplus r) - r = \Psi_{x'}(r) = \Psi_{x'}(r \oplus r_2) \oplus (\Psi_{x'}(r_2) \oplus \Psi_{x'}(0))$$

for a random r_2. The technique is clearly secure against first-order attacks.

2.1 Generalization to any Order

In this section we describe a first generalization to any order based on the previous Goubin method; however it does not seem to work.

We want to convert from:

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

to

$$x = A_1 + A_2 + \ldots + A_n$$

without any $(n-1)$-th order leakage.

We proceed iteratively by computing:

$$\begin{aligned}
x &= x_1 \oplus x_2 \oplus \ldots \oplus x_n \\
x &= A_1 + (x_2 \oplus \ldots \oplus x_n) \\
x &= A_1 + A_2 + (x_3 \oplus \ldots \oplus x_n) \\
&\vdots \\
x &= A_1 + \ldots + A_{n-2} + (x_{n-1} \oplus x_n) \\
x &= A_1 + \ldots + A_{n-2} + A_{n-1} + A_n
\end{aligned}$$

In the following we focus on the first conversion; the remaining conversions proceed similarly.

$$\begin{aligned}
x &= x_1 \oplus x_2 \oplus \ldots \oplus x_n \\
x &= A_1 + (x_2 \oplus \ldots \oplus x_n)
\end{aligned}$$

which gives:

$$\begin{aligned}
A_1 &= x_1 \oplus (x_2 \oplus \ldots \oplus x_n) - (x_2 \oplus \ldots \oplus x_n) \\
A_1 &= \Psi_{x_1}(x_2 \oplus \ldots \oplus x_n) \\
A_1 &= \Psi_{x_1}(x_2) \oplus \Psi_{x_1}(x_3) \ldots \oplus \Psi_{x_1}(x_n) \oplus u
\end{aligned}$$

where $u = \Psi_{x_1}(0) = x_1$ if n is odd, and $u = 0$ otherwise. This is thanks to the affine property of $\Psi_{x_1}(r)$.

Now one could compute the $\Psi_{x_1}(x_i)$ for $2 \leq i \leq n$ separately and eventually xor them, but the xor operation could leak information on x (even though the result A_1 does not leak information on x). One could try to randomize the process by xoring the intermediate variables by a sequence of random r_i's, but that does not seem to work.

The complexity of the first step is $\mathcal{O}(n)$; therefore the total complexity is $\mathcal{O}(n^2)$.

3 From Boolean to Arithmetic Masking for any Order

In this section we describe a conversion method from Boolean to Arithmetic masking that seems to work for any order. However it is less efficient, with complexity $\mathcal{O}(n^3)$.

We assume that there is an index j such that the senstitive variable x is shared as follows.

$$x = A_1 + A_2 + \ldots + A_j + (x_{j+1} \oplus \ldots \oplus x_n)$$

Initially we have $j = 0$ (Boolean masking), and eventually $j = n - 1$ (arithmetic masking).

We show how to go from step j to step $j + 1$, without any small order leakage. The technique consists in writing:

$$\begin{aligned}
x &= A_1 + \ldots + A_j + (x_{j+1} \oplus \ldots \oplus x_n) \\
x &= A_1 + \ldots + A_j + (-r) + (r + (x_{j+1} \oplus \ldots \oplus x_n)) \\
x &= A_1 + \ldots + A_j + (-r) + ((r_{j+1} \oplus \ldots \oplus r_n) + (x_{j+1} \oplus \ldots \oplus x_n)) \\
x &= A_1 + \ldots + A_j + A_{j+1} + \left(x'_{j+2} \oplus \ldots \oplus x'_n\right)
\end{aligned}$$

where $r_{j+1} \oplus \ldots \oplus r_n = r$, and $A_{j+1} = (-r)$.

Therefore one must perform the addition:

$$(r_{j+1} \oplus \ldots \oplus r_n) + (x_{j+1} \oplus \ldots \oplus x_n) = x'_{j+2} \oplus \ldots \oplus x'_n$$

without any low-order leakage. This corresponds to a secure evaluation of the circuit for addition, with shares of size $n - j = \mathcal{O}(n)$. For k-bit registers, the addition circuit has size $\mathcal{O}(k)$. Therefore using the ISW technique [2] this can be done in time $\mathcal{O}(k \cdot n^2)$. Therefore the full complexity is $\mathcal{O}(k \cdot n^3)$.

Note that the technique does not use Goubin's conversion method. Goubin's conversion method could be used in the last step, to convert $x_{n-1} \oplus x_n$ into $A_{n-1} + A_n$, but this would not modify the total complexity.

4 From Arithmetic to Boolean Masking

Goubin also described in [1] a technique for converting from arithmetic to boolean masking, secure against first-order leakage. However it is more complex than from boolean to arithmetic masking. Its complexity is $\mathcal{O}(k)$ for registers of k bits. It is based on the following theorem.

Theorem 2 (Goubin [1]). *If we denote $x' = (A + r) \oplus r$, we also have $x' = A \oplus u_{k-1}$, where u_{k-1} is obtained from the following recursion formula:*

$$\begin{cases} u_0 = 0 \\ \forall k \geq 0, u_{k+1} = 2[u_k \wedge (A \oplus r) \oplus (A \wedge r)] \end{cases}$$

5 From Arithmetic to Boolean Masking for any Order

We use the same technique as in Section 3. We write:

$$\begin{aligned}
x &= A_1 + \ldots + A_{j-1} + A_j + (x_{j+1} \oplus \ldots \oplus x_n) \\
x &= A_1 + \ldots + A_{j-1} + (A_j + (x_{j+1} \oplus \ldots \oplus x_n)) \\
x &= A_1 + \ldots + A_{j-1} + ((r_j \oplus \ldots \oplus r_n) + (x_{j+1} \oplus \ldots \oplus x_n)) \\
x &= A_1 + \ldots + A_{j-1} + \left(x'_j \oplus \ldots \oplus x'_n\right)
\end{aligned}$$

where $r_j \oplus \ldots \oplus r_n = A_j$. The complexity for one step is $\mathcal{O}(k \cdot n^2)$ and the full complexity is again $\mathcal{O}(k \cdot n^3)$.

6 From Boolean to Arithmetic Masking for any Order

We show a variant of the technique from Section 2.1 that could work.

We use a procedure PartialConvert which is defined as follows:

$$\mathsf{PartialConvert}(x_1, \ldots, x_d) = (y_1, \ldots, y_{d-1})$$

where

$$(y_1 \oplus \ldots \oplus y_{d-1}) + (x_2 \oplus \ldots \oplus x_d) = x_1 \oplus \ldots \oplus x_d$$

In other words, PartialConvert provides a $(d-1)$-th boolean sharing of A_1, where

$$A_1 + (x_2 \oplus \ldots \oplus x_d) = x_1 \oplus \ldots \oplus x_d$$

Using Goubin's convertion method it is easy to obtain such procedure PartialConvert:

$$\begin{aligned}
A_1 &= x_1 \oplus (x_2 \oplus \ldots \oplus x_d) - (x_2 \oplus \ldots \oplus x_d) \\
A_1 &= \Psi_{x_1}(x_2 \oplus \ldots \oplus x_d) \\
A_1 &= \Psi_{x_1}(x_2) \oplus \Psi_{x_1}(x_3) \ldots \oplus (\Psi_{x_1}(x_d) \oplus u) \\
A_1 &= y_1 \oplus \ldots \oplus y_{d-1}
\end{aligned}$$

where $u = \Psi_{x_1}(0) = x_1$ if n is odd, and $u = 0$ otherwise. This is thanks to the affine property of $\Psi_{x_1}(r)$. We take $y_i = \psi_{x_1}(x_{i+1})$ for $2 \le i \le d$. It is probably a good idea to further re-randomize the y_i's. The complexity of PartialConvert is $\mathcal{O}(d)$.

Using PartialConvert with input a d-th order Boolean sharing, we obtain as output a arithmetic sum of 2 Boolean sharing of order $d-1$. We can therefore apply the PartialConvert procedure recursively on both parts.

The total complexity to convert a n-th order Boolean masking into arithmetic masking is then $\tilde{\mathcal{O}}(2^n)$. This is worse than the $\mathcal{O}(n^3)$ complexity from Section 3 but this might be more advantageous for small n, since we don't have to perform bit manipulations. More precisely taking into account the register size, the complexity is $\tilde{\mathcal{O}}(2^n)$ instead of $\mathcal{O}(k \cdot n^3)$ in Section 3. So this method is more advantageous for small n and large register size k; this may correspond to what is used in practice.

Finally we note that we could use a similar technique for the convertion from arithmetic to Boolean masking. The complexity would be $\tilde{\mathcal{O}}(2^n) + \mathcal{O}(k \cdot n^2)$ instead of $\mathcal{O}(k \cdot n^3)$ in Section 5

7 Yet Another Variant

We note that in both Sections 3 and 5 we must compute:

$$r + (x_{j+1} \oplus \cdots \oplus x_n) = x'_j \oplus \cdots \oplus x'_n$$

Now instead of using a circuit of size $\mathcal{O}(k \cdot n^2)$ for performing the arithmetic addition, we can actually use Goubin's second method for conversion from arithmetic too boolean masking.

Namely denoting $x' = x'_j \oplus \cdots \oplus x'_n$, we have:

$$x' \oplus r = x'_j \oplus \cdots \oplus (x'_n \oplus r) = (A + r) \oplus r$$

where

$$A = x_{j+1} \oplus \cdots \oplus x_n$$

Therefore we use the compute the recurrence from Theorem 2, with $x' = (A + r) \oplus r = A \oplus u_{k-1}$, where:

$$\begin{cases} u_0 = 0 \\ \forall k \geq 0, u_{k+1} = 2[u_k \wedge (A \oplus r) \oplus (A \wedge r)] \end{cases}$$

where we compute u_k and A with n boolean shares. Therefore the result $x' = A \oplus u_{k-1}$ has also n boolean shares (if there are too many boolean shares, we can simply reduce the number of shares).

The complexity of this step is still $\mathcal{O}(k \cdot n^2)$ operations, and the total complexity is still $\mathcal{O}(k \cdot n^3)$. However, this latter method might be easier to implement.

References

1. L. Goubin, *A Sound Method for Switching between Boolean and Arithmetic Masking*. Proceedings of CHES 2001. 2, 1, 2, 4, 2
2. Y. Ishai, A. Sahai and D. Wagner, Private Circuits: Securing Hardware against Probing Attacks, Proceedings of Crypto 2003. 3