2024 Design, Automation & Test in Europe Conference (DATE 2024)

Securing ISW Masking Scheme Against Glitches

Sofiane Takarabt[‡], Mohammad Ebrahimabadi[*] Javad Bahrami[*], Sylvain Guilley[‡], and Naghmeh Karimi[*]

[‡]Secure-IC S.A.S., France
[*]University of Maryland Baltimore County, United States

***Abstract*—Ishai-Sahai-Wagner (ISW) masking scheme has been proposed in literature to protect cryptographic circuitries against side-channel analysis attacks. Although provably secure from a theoretical standpoint, its hardware implementation may not be secure as such security proof holds true if the gates are only evaluated after all of their inputs are available, yet such requirement is not met in hardware as the gates are evaluated as soon as any single input of them is changed. This paper provides a repair for ISW to address its security concern and prevent the key recovery. Our method is based on inserting artificial delays and/or "refreshing" on some sensitive paths to ensure that the underlying combinational gates are evaluated in the order expected by the ISW rationale. We verify the security of our proposed structure by leakage detection. Our solution is called E-ISW standing for Enhanced-ISW.**

I. Introduction

Masking schemes have been proposed in literature to protect the cryptographic devices against Side-Channel Analysis (SCA) attacks [1, Chap. 9]. Among them, the Ishai-Sahai-Wagner (ISW) is the first provable *universal* masking scheme that allows to perform arbitrary (i.e., XOR and AND) computations over bits (that is in finite field $(\mathbb{F}_2, \oplus, \cdot)$); thus it is highly appealing for crypto applications as any function can be decomposed as a sum of products. Yet in general, masking is not 100% bullet-proof, as some transient effects (known as *glitches*) can annihilate the benefit of masking. In this respect, ISW has been also shown to be vulnerable to glitches [2]. To fill the gap this paper proposes a simple repair mechanism that makes the glitches unexploitable to recover the key. Our method (so-called E-ISW referring to Enhanced ISW) operates by controlling the arrival time of each gate's inputs. This paper discusses the shortcoming of ISW and leverages delay elements (buffers) to keep the ISW function intact while preventing the premises of glitches exploitation; thus making it secure.

Masking Countermeasure: Masking is a register transfer level protection against SCA and is realized by randomly splitting every sensitive variable $A \in \{0,1\}$ into several shares $(A_1, \ldots, A_n)$, such that A can be rebuilt by XORing all shares (invariant is $A = \bigoplus_{i=1}^{n} A_i$). Such masking (a.k.a. "Boolean masking") is based on the fact that an attacker who gets to know, e.g., by probing, up to $(n-1)$ shares, cannot rebuild the sensitive variable, and must probe all n shares to recover a.

Motivation: The way the shared data are combined in the execution of computations is not obvious, since many flawed masking schemes have been proposed. In contrast, ISW comes with a provable realization of multiplication, which explains why it is a seminal masking scheme. It remains secure even if the intermediate variables required to perform computations are probed, at any order n.

Indeed in ISW (Boolean) addition is a matter of computing XORs between the shares. Correction of this addition stems from the associativity of XOR operation. (Boolean) multiplication is more complex; typically, two shared bits $(A_1, \ldots, A_n)$ and $(B_1, \ldots, B_n)$, yield, after multiplication, $(C_1, \ldots, C_n)$, such that $\bigoplus_{i=1}^{n} C_i = (\bigoplus_{i=1}^{n} A_i)(\bigoplus_{i=1}^{n} B_i) = \bigoplus_{i,j=1}^{n} A_i B_j$, which needs, for its implementation to be secure, to involve some additional (tautological) randomness, termed *refresh*. This makes the specificity of ISW. Figure 1a shows the netlist of an AND gate in the original ISW. Here R is a set of *refresh* signals; it doesn't have a functional role but guarantees the security at desired order.

Although ISW was considered a constructive method to achieve side-channel resistance at any order initially, thorough analyses showed that ISW contained a flaw [2] in its hypotheses as it assumes that *combinational gates* are *synchronizing*. This is not obviously the case in practice, as combinational gates evaluate as soon as any single input changes values. In particular, the output of combinational gates may toggle multiple times within one clock period owing to this fact. Accordingly, ISW as a *hardware* protection against SCA has been disregarded lately and ISW's merits have shifted to *software* implementations. The seminal work in this respect is [3][1]. Indeed, the software relies on basic instruction sets, made up of simple Boolean operations executed sequentially. In this respect, ISW in software is secure as it meets the requirement of each "gate" evaluating only once.

II. Enhancing the Security of ISW

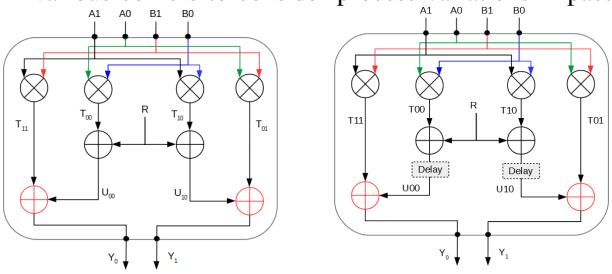
Considering the security shortcoming of the hardware implementation of ISW, a number of state-of-the-art alternatives were proposed in literature including Threshold Implementation (TI [5]) and Domain Oriented Masking (DOM [6]). TI Creates masked circuits such that one (or more) share is missing from each path being evaluated. This ensures, by design, that glitches cannot reveal information on unmasked values. DOM can be seen as a version of TI where some gadgets are pipelined. Still, (high-order) TI has encountered some security flaws [7], which shows that the security of an alternative to ISW is not obvious. In addition TI needs 3 shares at least to fulfill the requirement of incompletness. DOM has 2 shares but adds a pipeline stage; thus adds more latency. Thus we propose to add delay elements (buffers) in the ISW paths to enforce the order of gate evaluations in hardware as expected and make this low-cost scheme secure to be able to use it in cases where the area overhead of including DOM or TI is prohibitive. Thus we propose re-enforcing the conditions in which gates are synchronizing by benefiting from local *engineering change order* (ECO). Table I compares the area and delay of the AND gate implementation in different

[1]This work contains a minor flaw owing to reuse of some masked values without refresh [4].

979-8-3503-4859-0/DATE24/© 2024 EDAA

unprotected and masking schemes as well as our proposed E-ISW. We target the AND gate here as it is the most complex component to be made secure and also because it is the basic unit for measuring area. E-ISW is similar in size compared to its contenders (TI & DOM), but TI complexity grows faster with netlist size as it is not amenable to any decomposition strategy. Note that lightweight block ciphers are designed to have short critical paths. while they are typically used in low-end Internet of Things (IoT) applications, which work at low clock frequencies. For those two reasons, the delay in the netlist is a metric of secondary importance in our study.

Applying our method to original ISW AND gate (Fig. 1a) results in Fig. 1b, where two delay elements (each including several buffers) are inserted. In this case, no refresh is necessary. Obviously, our approach must be backed with simulations in various corners to consider process variations impacts.



(a) Original.

(b) Secured using delay element (i.e. buffers).

Figure 1: ISW and E-ISW AND gates. The symbols "+" and "×" represent XOR and AND, respectively.

III. Our ISW Repair

ISW is made up of a combination of masked AND and XOR gates. The masked AND gate shown in Fig. 1 is secure if the inserted delay elements ensure that the signal T_{11} (resp T_{01}) arrives before U_{00} (resp. U_{10}), and no transitions are combined in the final XOR gates (shown in red). Without the delay elements shown in this figure, this masked AND gate is vulnerable. Accordingly to make it secure we add delay elements to ensure ISW gates are evaluated in the order that was expected. Note that in the original ISW (Fig. 1a) the inputs affecting U_{00} and T_{11} are different; thus no guarantee that U_{00} comes after T_{11}. For example, in the case where the signals A_1 and A_0 get stable before B_1 and B_0, glitches arrive (randomly) from B_1 and B_0, thus the final XOR (generating Y_0 or Y_1) may see some glitches at the same time from both entries. This transition results in $A_0 \oplus A_1$ which is the unmasked value; thereby leaking the secret data.

IV. Results Validations and Discussions

A. Experimental Results

We implemented the AND gate in different unprotected and masking styles including GLUT, DOM, TI, and ISW as well as our E-ISW protected version in transistor level using a 45 nm NANGATE library. We sample the power every 1 ps after feeding the related final value till each circuit gets stable. Both initial and final values are generated randomly, and *input is intentionally glitched* to test in challenging conditions. We generated input traces exhaustively for each case.

In general, to ensure that a design is leakage-free against first-order attacks, we need to perform leakage detection analysis. However, if such analysis depicts that there is a leakage, the next step would be offensive attacks to ensure there is no false alarm for such leakage reporting. Here we show the first step, i.e., leakage analysis (*as per* ISO/IEC 20897 method).

Table I: Comparing proposed E-ISW with the original ISW and other masking schemes [8]. LUT is unprotected. Total Equivalent gates refers to the number of gates normalized by the number of equivalent 2-input NAND gates.

	LUT	GLUT	ISW	E-ISW	TI	DOM
Total Equ. Gates	1.5	9.5	16.0	34.0	28.5	44.0
# Random Bits	0	3	3	3	3	3
Max Delay (ns)	0.03	0.16	0.18	0.41	0.19	0.15
Latency (#Cycles)	0	0	0	0	0	1

We compare the unprotected LUT as well as a number of masking schemes including TI, DOM, GLUT, and ISW along with our E-ISW in terms of Normalized Inter-Class Variance (NICV) using HSpice power traces. Recall that NICV is:

$$\text{NICV}(X,T) = \mathbb{V}[\mathbb{E}[T|\ X]]\ /\ \mathbb{V}[T]\ \ \in\ \ [0,1]\ \text{, where}$$

X is the sensitive variable and T is the side-channel trace.

Figure 2 shows the NICV results of the different implementations of the masked AND gate. As we can see, the leakage on ISW is very clear and can be easily exploited, while E-ISW does not show any leakage. It shows an NICV pattern similar to those of TI and DOM, which are known to be secure. This demonstrates the effectiveness of our approach.

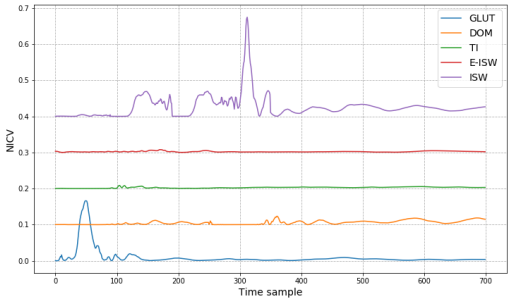


Figure 2: NICV result on different masking schemes of AND gate. For clarity concerns we have separated the graphs with a vertical offset of 0.1 each.

V. Conclusion

ISW is a masking scheme, provably secure under the assumption that the gates it utilizes as building blocks are *synchronizing*. Not satisfying this constraint leads to glitches, and in turn the key recovery. In this paper, we only focus on the AND gates resided in the ISW implementation and insert delay elements (buffers) to prevent glitches by controlling the order in which the gates should be evaluated.

References

[1] S. Mangard, E. Oswald, and T. Popp, *Power analysis attacks - revealing the secrets of smart cards*. Springer, 2007.

[2] B. R. Debapriya et al., "From theory to practice of private circuit: A cautionary note," in *ICCD*, 2015, pp. 296–303.

[3] M. Rivain and E. Prouff, "Provably Secure Higher-Order Masking of AES," in *CHES*, 2010, pp. 413–427.

[4] J. Coron et al., "Higher-Order Side Channel Security and Mask Refreshing," in *Fast Software Encryption*, 2013, pp. 410–424.

[5] S. Nikova et al., "Threshold Implementations Against Side-Channel Attacks and Glitches," in *ICICS*, vol. 4307, 2006, pp. 529–545.

[6] H. Groß et al., "Domain-Oriented Masking: Compact Masked Hardware Implementations with Arbitrary Protection Order," in *Workshop on Theory of Implementation Security*. ACM, 2016, p. 3.

[7] T. Moos et al., "Glitch-Resistant Masking Revisited or Why Proofs in the Robust Probing Model are Needed," *CHES*, pp. 256–292, 2019.

[8] J. Bahrami et al., "Leakage Power Analysis in Different S-Box Masking Protection Schemes," in *DATE*, 2022, pp. 1263–1268.