# Low-Cost First-Order Secure Boolean Masking in Glitchy Hardware

Dilip Kumar S V*, Josep Balasch†, Benedikt Gierlichs*, Ingrid Verbauwhede*
*imec-COSIC, †e-Media Research Lab STADIUS, KU Leuven, Leuven, Belgium
{dshanmug, josep.balasch, benedikt.gierlichs, ingrid.verbauwhede}@esat.kuleuven.be

***Abstract*—We describe how to securely implement the logical AND of two bits in hardware in the presence of glitches without the need for fresh randomness. As a case study, we design, implement and evaluate a DES core using our AND gate. Our goal is an overall practically relevant tradeoff between area, latency, randomness cost and security. We focus on first-order secure Boolean masking and we do not aim for provable security. The resulting DES engine shows no evidence of first-order leakage in a non-specific leakage assessment with 50M traces.**

***Index Terms*—Side-channel analysis, Masking, Glitches**

## I. Introduction

Over the last few decades, a lot of attention has been dedicated to researching and developing fast and efficient cryptographic implementations that are secure against power analysis attacks. Masking is a well-known technique that can be used to protect both hardware and software implementations. Its core idea is to split the data being processed by an implementation into random shares, effectively eliminating its correlation with the device's power consumption. Modern masking techniques, such as Threshold Implementation (TI) [1] and Domain-oriented Masking (DOM) [2], have been designed to address the problem caused by glitches. In contrast to classical Boolean masking, they eliminate the propagation of glitches through register layers and maintain the uniformity of the intermediate values by injecting fresh randomness. As a result, they achieve *provable* security against first-order attacks. But provable security comes with higher costs. Protected implementations using modern masking schemes require more resources in terms of area, latency and randomness than classical Boolean masking. In hardware, masking is commonly applied at the gate level. As logic gates are used as a fundamental building block in gate-level masking, any cost reduction in building a masked logic gate benefits the overall cost of a masked circuit significantly. To this end, in this work, we develop a low-cost Boolean masked AND gate suitable for hardware implementations which requires no fresh randomness. Additionally, we discuss guidelines for building low-cost circuits using our proposed low-cost AND gadget. And as a case study, we design and implement a masked DES encryption engine that provides *practical* security in the presence of glitches. And finally, we evaluate the performance of our design both in terms of cost (area, latency, randomness) and first-order side-channel leakage on an FPGA platform.

This research has been funded in part by KU Leuven Startfinanciering STG/20/047, by CyberSecurity Research Flanders VR20192203, by VLAIO ICON CS IDIoTS HBC.2020.2789, by the Research Council KU Leuven C16/15/058, by the European Commission through Belfort ERC Advanced Grant 101020005 695305, through HORIZON-CL3-2021-CS-01-02 ORSHIN 101070008, and through H2020 Twinning SAFEST 952252.

## II. Low-Cost Masked AND2 Gadget

We start our work from the masked AND2 gadget proposed by Biryukov et al. [3] for software implementations. To compute $z = x \cdot y$, where $x = x_0 \oplus x_1$, $y = y_0 \oplus y_1$ and $z = z_0 \oplus z_1$:

$$\begin{aligned} z_0 &= (x_0 \cdot y_0) \oplus (x_0 + \overline{y_1}) \\ z_1 &= (x_1 \cdot y_0) \oplus (x_1 + \overline{y_1}) \end{aligned} \tag{1}$$

From now on, we will refer to this gadget as `secAND2`. A remarkable property of this gadget is that it does not require fresh randomness to be secure. Instead, the uniformity of the output is achieved by reusing the randomness of the inputs. This characteristic becomes critical when implementing circuits that combine several terms, for instance, through addition. It can lead to a decrease in security if the added terms are not independent. For this reason, the outputs of some of `secAND2` gadgets are selectively refreshed. Adding security measures only when necessary in turn decreases the cost of fresh randomness required to build circuits.

A further well-known problem arises when implementing such a gadget in hardware as glitches can happen. Glitches on the output of a logic gate are created by different arrival times of its input signals. It is impossible to predict the order in which the inputs arrive in a large circuit. To implement `secAND2` securely, we take control over the order of and the delay between the input signals, and send the inputs in a safe sequence such that there are no glitches and thus no leakage of any information about the sensitive inputs or intermediate values. For `secAND2`, as long as either $y_0$ or $y_1$ arrive last, we observe no leakage. These results can be explained from the `secAND2` equations in (1). It is easy to see that $z_0$ depends on $x_0, y_0$ and $y_1$ whereas $z_1$ depends on $x_1, y_0$ and $y_1$. Therefore, by delaying the arrival of $y_0$ or $y_1$ we can achieve a *temporary* non-completeness property (refer to the non-completeness property of Threshold Implementations) for both output bits during the evaluation. For our purpose of creating a secure low-cost masked AND gate, we use a flip flop (FF) to delay one of the shares $y_0/y_1$, see Figure 1. Products of more than just two variables, can be computed by cascading multiple `secAND2` gadgets. With careful placement of inputs and interconnections between the cascaded `secAND2` gadgets, we can obtain a construction with no additional (i.e. external) FFs. Thereby ensuring a small area footprint.

978-3-9819263-7-8/DATE23/© 2023 EDAA
Authorized licensed use limited to: KU Leuven Libraries. Downloaded on August 09,2023 at 13:07:43 UTC from IEEE Xplore. Restrictions apply.

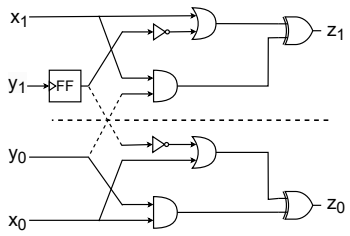


Fig. 1. `secAND2` gate with internal FF.

## III. Case Study : DES

We chose the Data Encryption Standard (DES) as our target algorithm because it is the main building block of TDES or 3DES, which is still widely used today. The main difficulty in protecting the DES implementation lies in the S-boxes, which are the only non-linear components. A hardware-friendly way to implement them is to represent them as four 4-bit permutations (the so-called *mini S-boxes*) and a multiplexer (MUX). Each mini S-box/MUX can be represented through equations in Algebraic Normal Form (ANF) that consist only of AND and XOR gates. These equations can be split into two stages: AND stage and XOR stage. In the AND stage, the inputs of the mini S-box (or MUX) are multiplied using our `secAND2` gadget. Then, in the XOR stage, the outputs of the AND stage are combined to produce the outputs of the mini S-Box (or MUX).

Our final DES implementation is constructed by adapting a round-based architecture to incorporate first-order Boolean masking. The performance results are summarized in Table I. We also provide the numbers from the work in [4], reporting a 3DES implementation protected with DOM, in the table. Note that the number of cycles reported in [4] is scaled down to compare with our DES implementation.

TABLE I
Utilization results of DES implementations.

| Version | | ASIC [GEs] | FPGA [FFs] | FPGA [LUT6s] | Randomness [bits/round] | Cycles |
|---|---|---|---|---|---|---|
| *this work* | | 14488* | 819 | 2129 | 14 | 7*16+3 |
| without recycling | | - | - | - | 14*8 = 112 | 7*16+3 |
| 2* [4] | DOM-indep | 13800† | - | - | 22*8 = 176 | 5*16+4 |
| | DOM-dep | 22400† | - | - | 22*3*8=528 | 5*16+4 |

† Calculated using a 28 nm RVT standard cell library from Global Foundries.
* Calculated using NanGate 45nm Open Cell Library.

From the area results, we can see that our design is more compact than the DOM-dep version in [4] but slightly larger than the DOM-indep version. Our design uses 30 `secAND2` gates per DES S-Box, compared to 22 DOM-indep gates in [4]. While `secAND2` gates are by design smaller than the DOM-indep gates, the additional modules in our implementation (e.g. refreshing gadgets, input/output S-box registers, etc.) compensate the gain to some extent. An important remark here is that our GE results include a masked key schedule whereas the results in [4] include the cost of an unmasked key schedule. The overhead caused by this module further explains the difference between both designs. In terms of randomness usage, our design requires less bits per round as we reuse randomness across the 8 different S-boxes. This decision does not have an impact on the first-order security (see below) and hence we opt to use it in our reference implementation. If our design did not recycle randomness, or if the designs in [4] did, then our random bits requirements would still be lower. We also note that there is no security evaluation of the DOM-indep version in [4]. They evaluate only the DOM-dep version which consumes 3 random bits per refreshing. Finally, our implementation takes two more cycles per round compared to [4].

We lastly evaluate evaluate the side-channel security of our protected DES implementation using a SAKURA-G board equipped with a Spartan-6 FPGA. We use the non-specific Test Vector Leakage Assessment (TVLA) methodology, and we perform three fixed vs. random tests (50 million traces each) with three different fixed plaintexts. Figure 2 shows from top to bottom: a power trace, first, second and third-order univariate t-test results, for one of the fixed plaintexts. As we can see there is no evidence of first-order leakage. There are very few and minor crossings of the 4.5 threshold in the first-order t-test values, but they are not consistent across the three different plaintexts, i.e. the threshold is not exceeded at the same time indexes, as is required by the TVLA methodology in order to deem a device leaking.

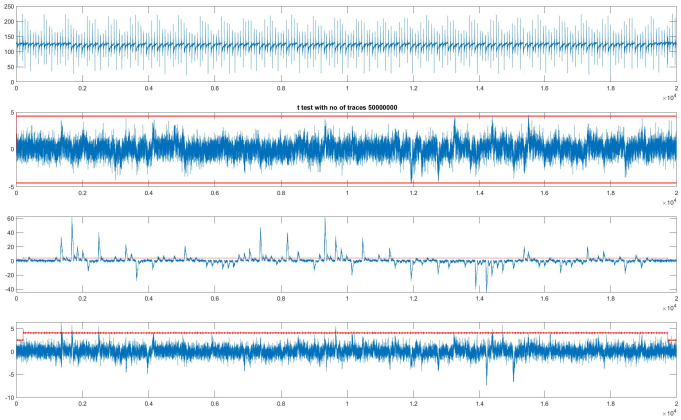


Fig. 2. TVLA FvR result using 50 Million Traces.

## References

[1] S. Nikova, C. Rechberger, and V. Rijmen, "Threshold implementations against side-channel attacks and glitches," in *Information and Communications Security - ICICS 2006*, ser. LNCS, P. Ning, S. Qing, and N. Li, Eds., vol. 4307. Springer, 2006, pp. 529–545.

[2] H. Groß, S. Mangard, and T. Korak, "Domain-Oriented Masking: Compact Masked Hardware Implementations with Arbitrary Protection Order," in *Theory of Implementation Security - TIS@CCS 2016*, B. Bilgin, S. Nikova, and V. Rijmen, Eds. ACM, 2016, p. 3.

[3] A. Biryukov, D. Dinu, Y. L. Corre, and A. Udovenko, "Optimal first-order boolean masking for embedded iot devices," in *Smart Card Research and Advanced Applications - CARDIS 2017*, ser. LNCS, T. Eisenbarth and Y. Teglia, Eds., vol. 10728. Springer, 2017, pp. 22–41.

[4] P. Sasdrich and M. Hutter, "Protecting triple-des against DPA - A practical application of domain-oriented masking," in *Constructive Side-Channel Analysis and Secure Design - COSADE 2018*, ser. LNCS, J. Fan and B. Gierlichs, Eds., vol. 10815. Springer, 2018, pp. 207–226.

Authorized licensed use limited to: KU Leuven Libraries. Downloaded on August 09,2023 at 13:07:43 UTC from IEEE Xplore. Restrictions apply.