You Can Hide but You Can’t Verify:
On Side-channel Countermeasure Verification

Patrick Schaumont
Worcester Polytechnic Institute, Worcester, MA 01609

Abstract—Power-based Side-channel Leakage (SCL) is a hardware vulnerability that may be mitigated using hardware countermeasures. We present a hardware-based hiding countermeasure for a small RISC-V (RV32IMC) based SoC, and apply it to hide the leakage of a software implementation of AES. We evaluate SCL by gate-level simulation of the design files and by measuring a 180nm ASIC prototype. We then compare these two approaches and optimize the hiding countermeasure. Our results highlight the specific challenges of the security verification of a hiding-based countermeasure.

Index Terms—hardware security, cryptographic engineering, reverse engineering

I. INTRODUCTION

Power-based Side-channel Leakage (SCL) has been exploited in a growing variety of settings, even by remote monitoring of power consumption of cloud-based FPGA [1]. There is a perceived need that secure hardware designs must be protected by countermeasures to mitigate the risk of SCL exploitation. Such countermeasures come in two flavors. Masking countermeasures use secret-sharing to make the lower-order statistics of the SCL independent of the secret intermediate variables [2]. Hiding countermeasures reduce the contribution of the secret intermediate variables’ power consumption to the overall power consumption variations. Neither technique is perfect, and either can fail due to countermeasure implementation errors, or to more powerful side-channel attacks. Hence, verification of side-channel countermeasures is mandatory. In current practice, this verification is almost always done post-silicon, namely when the actual device is available. Recently, considerable attention has been given to the case of pre-silicon security verification [3]. The compelling arguments for pre-silicon security verification include reducing the time-to-market by eliminating or simplifying post-silicon security testing, increasing the understanding of side-channel leakage root causes, and design-space exploration involving security-versus-cost trade-offs.

In this contribution we look at a concrete case of security verification for a hiding countermeasure, comparing pre-silicon simulations with post-silicon measurements. We highlight the importance (and difficulty) of taking all factors into account into SCL verification. Fig. 1 shows the layout of a small SoC with a RISC-V core, on-chip RAM, several cryptographic accelerators and peripherals, and an array of 32 ring oscillators (ROs). Table I summarizes the main properties of the chip. The ROs are distributed over the standard cell area, and they are configurable from the RISC-V core through a memory-mapped interface. Each RO has a 32-bit counter to measure the RO oscillation frequency through integration. The ROs serve multiple objectives [4]. They can sense electromagnetic fault injection and voltage glitching, since these fault vectors affect the immediate RO frequency. Next, the ROs can indirectly sense the on-chip activity, as the voltage of each RO is taken from the power distribution network of surrounding components. Finally, the ROs can serve as a hiding side-channel countermeasure, by switching the ROs randomly on and off. The objective of this contribution is to verify (only) the efficiency of the ROs when operating as a side-channel countermeasure.

This paper is organized as follows. Section II discusses related work and highlights the challenge of security verification of a countermeasure. Section III describes the measurement setup and parameters. Section IV summarizes our experimental results, followed by the lessons learned. We then conclude the paper.

II. RELATED WORK

Unlike masking-based side-channel countermeasures which are based on secret-sharing, hiding-based countermeasures have no formal notion of security. That may explain why there are relatively fewer publications exploring hiding-based countermeasures. The objective of a hiding based countermeasure is to degrade the Signal-to-Noise Ratio (SNR) of side-channel

![Fig. 1. Layout of the PICO test chip, a small SoC with 32 ring oscillators (ROs) working as a hiding countermeasure. The ROs are distributed in two groups of 16 RO each, marked with blue and yellow squares.](image-url)
leakage. The S-part represents side-channel leakage power, while the N-part represents the noise power added by measurement and by unrelated activities. There are two strategies to develop a hiding countermeasure. One can either decrease the application-dependent S-part, or one can increase the platform/environment dependent N-part. In the first category, Wave Dynamic Differential Logic (WDDL) was arguably the first generic, logic-level solution for hiding. Moreover, WDDL was demonstrated on a test chip [5]. Similar hiding techniques have been explored using architecture-reconfiguration [6], and as random delays and operation-shuffling in software [7]. More recently, energy buffering is proposed as an application-dependent hiding technique [8]. Our work belongs to the second category, and we degrade the SNR using randomly switched ring oscillators. In reconfigurable platforms, noise generation was previously demonstrated using shift registers, artificial short-circuit and memory-write collisions [9]. The generic nature of a noise generator also makes it appealing as an add-on technique as described by patents [10].

The evaluation of a countermeasure’s quality is challenging, because there is no universally accepted technique to quantify SCL [11]. In this paper, we evaluate the hiding countermeasure by testing its effectiveness to protect a software AES implementation running on the RISCV processor. We measure the power of the complete SoC without and with the countermeasure. We then aim to show that AES SCL can no longer be exploited when the countermeasure is turned on. To demonstrate leakage, we use traditional correlation power analysis (CPA) as well as a non-specific random-versus-fixed data test (TVLA). We do not apply the measurement-to-disclosure (MTD) metric. Indeed, we are primarily interested in relative comparisons: we check if the countermeasures are effective using a fixed number of traces (in our case: 10000 traces). An additional system-level security design requirement is then to replace the ephemeral keys every 50,000 encryptions.

While traditional SCL evaluation is done on prototypes, there is a rising interest in the use of simulation techniques [3]. By combining logic simulation and power estimation, side-channel traces are computed over a set of test vectors. The power traces are then assessed for SCL. Side-channel leakage can be simulated at different abstraction levels (RTL to transistor), and there is a steep trade-off between the simulation speed and the modeling detail [12]. We perform our power simulations at gate-level, which we found to be an acceptable trade-off between accuracy and speed [13]. While the exact interpretation of acceptable is application-dependent, it is based on an overall objective of completing an SoC-level SCL assessment within 24 hours.

### III. Measurement Setup

Our measurement setup includes a prototype (Fig. 2) and a corresponding simulation flow. The prototype Saidoyoki board supports chip-level testing and allows direct measurement of power-based side-channel leakage [14]. We use a Chipwhisperer Husky kit to feed test vectors over a serial connection to the PICO test chip, and then sample its power dissipation. The Husky kit generates the clock for the PICO test chip, which allows efficient synchronous sampling of side-channel leakage [15]. The PICO test chip runs at 4 MHz and its power is synchronously sampled at 16 MHz (4 samples per cycle). The setup in Fig. 2 captures around 20 traces per second at 16000 points per trace, which comfortably spans several SBOX operations of the AES. These SBOX operations are the focus of the SCL assessment.

We also use a simulation flow that captures the PICO test chip with gate-level accuracy (TSMC 180nm standard cells), and that performs gate-level power estimation using Cadence Joules. The power measurement in Cadence Joules is performed over frames, where a frame collects the average power consumption of the chip in a given time window. Joules practically limits the number of frames per simulation run to 1000, and we map the power estimation to one frame per two clock cycles. Because of the integration process during power estimation, no aliasing occurs [16]. While Joules is by far the slowest step in the power simulation flow, the overall flow is embarrassingly parallel. We typically run 20 concurrent Joules sessions on a design server, each working on a test vector. It takes around 150 minutes to produce 1000 power traces at gate-level accuracy for the entire PICO test chip. Despite the low sample rate of simulated traces, we still need far less simulated traces compared to the measured traces. The primary reason is the noiseless nature of the simulation.

<table>
<thead>
<tr>
<th>TABLE I</th>
<th>PICOSOC CHIP CHARACTERISTICS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Technology</td>
<td>TSMC 180nm (tcb018gbwp7l_270a)</td>
</tr>
<tr>
<td>Comb. Cells</td>
<td>43,724 cells</td>
</tr>
<tr>
<td>Seq. Cells</td>
<td>13,947 cells</td>
</tr>
<tr>
<td>Core Area</td>
<td>10.69 sq mm</td>
</tr>
<tr>
<td>Chip Area</td>
<td>13.35 sq mm</td>
</tr>
<tr>
<td>SRAM</td>
<td>64 KByte (8 x 8K macro)</td>
</tr>
<tr>
<td>Core</td>
<td>PicoRV (RV32IMC)</td>
</tr>
<tr>
<td>Crypto</td>
<td>AES, ASCON</td>
</tr>
<tr>
<td>Peripherals</td>
<td>UART, SPI, GPIO</td>
</tr>
<tr>
<td>Countermeasure</td>
<td>32 Configurable RO</td>
</tr>
</tbody>
</table>

Fig. 2. PICO test chip on the Saidoyoki board (left) with ChipWhisperer Husky (right)
TABLE II

<table>
<thead>
<tr>
<th>Test Vectors: Correlation Power Analysis (CPA) and Test Vector Leakage Assessment (TVLA)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPA</td>
</tr>
<tr>
<td>TVLA</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

Fig. 3. StdDev over traces from simulation (left) and measurement (right), as power over sample points. The plots are uncalibrated which explains the absence of Y-axis labels.

IV. EVALUATING THE COUNTERMEASURE

The hiding countermeasure works by turning on a random number (0 to 32) of ring oscillators before each security-sensitive operation. We run the ring oscillators at their maximum frequency (about 120 MHz), 30 times the chip’s clock frequency. Turning on and off the ring oscillators is done through a memory-mapped interface. It takes two memory-write operations to turn on or off an arbitrary number of ROs out of a group of 32. Each memory-write controls 16 ROs.

A. Unprotected Software AES

The reference implementation is a byte-wise software AES. A standard check to identify the location of potential side-channel leakage is to compute the standard deviation (STD) over a set of traces. Figure 3 shows the STD over 500 simulated traces and 10,000 measured traces for the CPA test vectors. One can clearly identify the SBOX lookup operations of AES, which computes $SBOX[state[i][j]]$ using the secret state. To assess the leakage, we use two different test techniques as listed in Table II. First, we compute a Welch-t statistic using the TVLA test set (Fig. 4). The t-test clearly marks leakage at every SBOX access with high confidence. Next, we also verified that a CPA easily recovers every key byte correctly on both simulated traces and measured traces.

B. Hiding-based Countermeasure

To protect the SBOX lookup operations, we enable a random number of RO at each SBOX lookup.

```c
static void SubBytes() {
    for(uint8_t i = 0; i < 4; ++i) {
        for(uint8_t j = 0; j < 4; ++j) {
            enable_ro_config(ct[i*4 + j]);
            state[i][j] = sbox[state[i][j]];
            disable_ro();
        }
    }
}
```

Fig. 5 (top) shows the impact of the RO on the simulated STD on 500 traces. The RO power variations are 2 orders of magnitude bigger than the power variations of the SBOX lookup. No keys can be recovered from the simulated traces. Fig. 5 (middle) shows the impact of the RO on the measured STD on 50K traces. The plot reveals the switch-on and switch-off events of the ROs for each SBOX lookup. However, once the RO power has stabilized, the measured STD drops. This is an artefact of the measurement setup, which measures power as a voltage drop over a shunt resistor. To remove the common mode, the signal is passed through a band-pass filter, masking out the DC component. As the RO power variation drops, the side channel leakage of the SBOX lookup becomes visible. Fig. 5 (bottom) shows that the design is still vulnerable. A time correlation plot on key bytes shows that the bytes can be recovered when the ROs have stabilized, just before they are switched off. We conclude that, while the design appears secure from a simulation point of view, the implementation shows this countermeasure design is flawed.

C. Hiding-based Countermeasure, revisited

To solve this flaw, we must realize that a hiding countermeasure must use broadband noise (N) to be effective.
Hence, we must switch a random number of ROs for (at least) each sensitive instruction. A manual study of the data dependencies in the assembly code of SBOX lookup shows that the sensitive operations can be reduced to just three instructions, highlighted in light gray below. We then surround these sensitive instructions with RO switch commands.

```c
static void SubBytes()
{
    lui a4, 0  // clear working register
    sw RO1, random1  // switch group 1 of 16 RO
    lbu a4, [state]  // read secret state
    add a4, a4, base  // add sbox base address
    sw RO2, random2  // switch group 2 of 16 RO
    lbu a4, [a4]  // read sbox[16]
    sw RO1, random3  // switch group 1 of 16 RO
    sb a4, [state]  // overwrite secret state
    sw RO2, random4  // switch group 2 of 16 RO
    lui a4, 0  // clear secret state
    disable_ro();
}
```

Fig. 6 indicates that distributing the RO switching commands keeps the power variation high throughout the sensitive SBOX lookup. The first three key bytes are now effectively hidden at 50K traces. However, keybyte 12 can still be recovered by CPA. This keybyte is accessed just at the last j-loop iteration (See code block in IV.C), and we suspect that leakage may be caused by an micro-architecture effect outside of the S-box access and will require further analysis. A further fixed-versus-random TVLA test on 50K traces demonstrates no leakage during Sbox access. However, the CPA test and the TVLA test used a different key.

V. CONCLUSION

We conclude by highlighting the significant challenge that exists to systematically test a countermeasure against side-channel leakage. The discrepancy between simulation and measurement, is not caused a chip-level modeling mistake but by a measurement effect. In hindsight, the flaw of the countermeasure and its fix seems obvious. But that didn’t prevent it from happening. Security verification of a cryptographic SoC must extend beyond the chip boundary.

REFERENCES