High-Level Synthesis of Side Channel Attack Resistant RSA Decryption Circuit

Naoki OSAKO Sayuri OTA Suguru YURA[†] Nagisa ISHIURA

School of Science and Technology, Kwansei Gakuin University
2-1 Gakuen, Sanda, Hyogo, 669–1337, Japan

Abstract—This paper presents a side channel attack resistant design of an RSA decryption circuit using high-level synthesis. An RSA encoder/decoder can be designed by high-level synthesis utilizing the GNU GMP library. With this methodology, an RSA decryption circuit is synthesized based on the Fournaris's algorithm which was designed against power analysis attacks and fault injection attacks. In order to reduce computation time and hardware size, Montgomery modular multiplication and parallelization of CRT-based modular exponentiation are applied. Results of synthesis targeting Xilinx Kintex-7 FPGA shows that the attack resistance has been implemented with the 1.94 times LUT count and 5.17 times execution cycles.

I. Introduction

The "Internet of Things (IoT)" is becoming an integral part of our daily life and explosively increasing number of devices are being connected to the Internet. To safeguard information across those devices, encryption technologies efficient both in hardware cost and power consumption are becoming important.

Efficiency of common key block cryptosystems such as AES (Advance Encryption Standard) may be drastically improved by manually designing dedicated hardware with streamlined architecture. On the other hand, public key cryptosystems such the RSA (RivestShamirAdleman) and ECC (Elliptic-Curve Cryptography) involve very long precision arithmetic which tend to end up with memory centric architecture. In this case, specialized hardware does not always exhibit overwhelming efficiency over processor implementation. In spite of this low return-of-investment, hardware implementation is still wanted to reduce hardware cost and power consumption of the IoT devices as much as possible. High-level synthesis is an attractive option to implement such complex computation into memory centric hardware.

Moreover, in the design of cryptosystem modules, extra care must be paid against side channel attacks [2]. Depending on desired security levels, sophisticated algorithms must be employed which hide power consumption changes associated with secret keys and do not emit significant information on fault injection. This further increases required design efforts.

To address this issue, in this paper, a side channel attack resistant RSA decryption circuit is designed by high-level synthesis. A side channel attack resistant algorithm proposed by Fournaris [1] is coded in the C language utilizing the GNU multiple precision arithmetic library (GMP), and RTL design is obtained using a high-level synthesizer ACAP [4]. In order to make the implementation further efficient, Montgomery reduction and parallelization on CRT-based modular exponentiation are applied. Synthesis targeting Xilinx Kintex-7 FPGA shows that attack resistance can be implemented at the cost of 1.94 times LUT count and 5.17 times execution cycles.

[†] Currently with Ministry of Defense, Kanagawa, Japan.

II. Protection against side channel attacks on RSA cryptosystem

In the RSA cryptosystem, the ciphertext c of message m is computed with public keys N and e according to:

$$c = m^e \bmod N \tag{1}$$

where $N = pq$ and p and q are prime numbers. Message m is recovered from c using private key d by computing:

$$m = c^d \bmod N \tag{2}$$

The high computation cost of modular exponentiation in (2) is often lessened utilizing the Chinese reminder theorem (CRT):

$$m_p = c^{d_p} \bmod p, \quad m_q = c^{d_q} \bmod q \tag{3}$$

$$m = m_q(q^{-1} \bmod p)q + m_p(p^{-1} \bmod q)p \tag{4}$$

A standard way to compute (3) is to iterate modular multiplications over the bits of d_p and d_q, which leads to different power consumption for different bit values. Moreover, changes on the outputs caused by intentionally injected faults on particular signal lines may be exploited to infer the secret keys.

Fournaris's algorithm [1] was designed to protect RSA decryption against those side channel attacks. It had resistance against the power analysis attacks as well as the fault injection attacks known at the point of publication, including Bellcore attack, KQ attack, and YLMH attack.

An outline of the algorithm is shown in Figure 1. The main function DECRYPT receives c, N, p, q, secret keys (d_p, d_q, i_q), and a random mask b and its inverse b^{-1}, to compute $c^d \bmod N$. Modular multiplication in equation (3) is computed by calling FSCAME (lines 4–5), and the output is computed according to the CRT (lines 6–9). Here, 4-tuple values instead of a single values are computed, which are used to detect fault injection; if fault injection is detected, *error* is returned not to leak the information. Function FSCAME computes modular exponentiation in encoded 4-tuple values. It is designed so that the amount of computation is independent of the bit values in the secret key d (lines 9–16). It also detects fault injection.

III. High-level synthesis of attack resistant RSA decryption circuit

A. Overview

In this paper, the algorithm in Figure 1 is coded in the C language to generate RTL design by high-level synthesis.

The GNU GMP library is used for multiple precision arithmetic. It is rewritten for the use of high-level synthesis [3]. Dynamic allocation is eliminated and statically allocated ar-

1: **Function** DECRYPT
2: **Input**: $c, N, p, q, d_p, d_q, i_q, b, b^{-1}$
3: **Output**: $c^d \bmod N$

4: $(s_0^p, s_1^p, s_2^p, s_4^p) = \text{FSCAME}(c, b, b^{-1}, d_p, p);$
5: $(s_0^q, s_1^q, s_2^q, s_4^q) = \text{FSCAME}(c, b, b^{-1}, d_q, q);$

6: $S_0 = s_0^q + q \cdot ((s_0^p - s_0^q) \cdot i_q \bmod p);$
7: $S_1 = s_1^q + q \cdot ((s_1^p - s_1^q) \cdot i_q \bmod p);$
8: $S_2 = s_2^q + q \cdot ((s_2^p - s_2^q) \cdot i_q \bmod p);$
9: $S_4 = s_4^q + q \cdot ((s_4^p - s_4^q) \cdot i_q \bmod p);$

10: **if** ($S_0 \cdot S_1 \bmod N = S_2$ **and** p, q not modified)
11: { **return** $S_0 \cdot S_4 \bmod N$; } **else** { **return** error; }

(a) RSA decryption.

1: **Function** FSCAME
2: **Input**: $c, M, d = (1, d_{t-2}, ...d_0), b, b^{-1}$
3: **Output**: (s_0, s_1, s_2, s_4)
4: // $s_0 = b^e \cdot c^e \bmod M$, $s_1 = b^{\bar{e}+1} \cdot c^{\bar{e}+1} \bmod M$
5: // $s_2 = b^{2^t} \cdot c^{2^t} \bmod M$, $s_4 = b^{-e} \bmod M$

6: $R = 2^{n+2}$; $T = R^2 \bmod M$; $T_R = T \cdot c \cdot R^{-1} \bmod M$;
7: $b_R = b \cdot R \bmod M$; $b_{R-1} = b^{-1} \cdot R \bmod M$;

8: $s_0 = s_1 = b_R$; $s_2 = b_R \cdot T_R \cdot R^{-1} \bmod M$; $s_3 = s_4 = s_5 = b_{R-1}$;
9: **for** ($i = 0$ to $t - 1$) {
10: **if** ($d_i = 1$) {
11: $s_0 = s_0 \cdot s_2 \cdot R^{-1} \bmod M$; $s_4 = s_4 \cdot s_3 \cdot R^{-1} \bmod M$;
12: } **else** {
13: $s_1 = s_1 \cdot s_2 \cdot R^{-1} \bmod M$; $= s_5 = s_5 \cdot s_3 \cdot R^{-1} \bmod M$;
14: }
15: $s_2 = s_2^2 \cdot R^{-1} \bmod M$; $s_3 = s_3^2 \cdot R^{-1} \bmod M$;
16: }
17: $s_0 = s_0 \cdot b^{-1} \cdot R^{-1} \bmod M$;
18: $s_1 = s_1 \cdot c \cdot R^{-1} \bmod M$;
19: $s_2 = s_2 \cdot 1 \cdot R^{-1} \bmod M$;
20: $s_4 = s_4 \cdot b \cdot R^{-1} \bmod M$;

21: **if** (i and d are not modified **and**
22: $s_0 \cdot s_1 \cdot R^{-1} \bmod M = s_2 \cdot 1 \cdot R^{-1} \bmod M$)
23: { **return** (s_0, s_1, s_2, s_4); } **else** { **return** error; }

(b) Attack resistant Montgomery modular exponentiation

Fig. 1. Fournaris's Algorithm [1].

rays are used. Once the maximum bit length of N is declared, the C code can compute decryption for N smaller than that.

B. Montgomery modular multiplication

Montgomery modular multiplication is a technique to eliminate division from modular multiplication. Let R be an integer where R and N are mutually prime and R^{-1} be an integer satisfying $RR^{-1} = 1 \bmod N$. Montgomery reduction is defined as $M(t) = tR^{-1} \bmod N$. Let $R_2 = R^2 \bmod N$. Then, modular multiplication $z = xy \bmod N$ is replaced by:

$$X = M(xR_2),\ Y = M(yR_2),$$
$$Z = MR(XY)$$
$$z = MR(Z)$$

If we choose R to be an exponential of 2, all these computation is performed only with multiplication, shift, and bit-wise *and* operations. We apply this transformation to all the modular multiplication in the algorithm to eliminate division.

C. Parallelization

There is no dependency between two modular exponentiation, so the calls to FSCAME in lines 4–5 in Figure 1 (a) can be computed in parallel. For this purpose, we synthesize another hardware module that compute only FSCAME. Then, function DECRYPT is modified so that it activates the FSCAME module for computation of line 4 before calling (its own) FSCAME in line 5, and collect the return values before proceeding to line 6. Since most computation time is spent in the FSCAME, substantial speed-up is expected by this parallelization.

TABLE I
SYNTHESIS RESULTS.

design	cycles	#LUT	freq. [MHz]
RSA (MIPS)	432,256	3,180	121.7
RSA	68,261	11,721	107.8
SRR (MIPS)	3,557,349	3,180	121.7
SRR	627,615	16,801	77.5
SRR+M	680,284	11,464	105.5
SRR+M+P	353,489	22,590	105.6

IV. SYNTHESIS RESULTS

The C code was synthesized into a Verilog HDL code by high-level synthesizer ACAP [4] and then mapped into FPGA (kintex-7 xc7k70) by Xilinx Vivado (2016.4). High-level synthesizer other than ACAP might also be used, but ACAP would need less modification on the C code because it can synthesize any C codes as long as their binaries can be executed on a bare-metal MIPS.

The synthesis results are summarized in TABLE I. The C code and the resulting circuits are independent of the number of bits for RSA decription, except for the size of arrays to store intermediate results. In this experiment, the parameters were set so that up to 2048-bit RSA decription can be executed. "Cycles" in the table indicate the number of cycles for RSA decryption of 128-bit data.

V. CONCLUSION

This paper has described high-level synthesis of a side channel attack resistant design of an RSA decryption circuit. Evaluation of the attack resistance is the future work. Even if the algorithm is resistant, high-level synthesizer might bring unexpected variation exploited as side channels [5]. We consider this an important topic to be investigated in details.

Acknowledgements—Authors would like to express their appreciation to Dr. H. Kanbara (ASTEM/RI), Prof. H. Tomiyama (Ritsumeikan Univ.), and Mr. T. Nakatani (formerly with Ritsumeikan Univ.) for their valuable comments. We would also like to thank to the members of Ishiura Lab. of Kwansei Gakuin Univ. for their cooperation. This work was partly supported by JSPS KAKENHI Grant #16K00088.

REFERENCES

[1] P. Fournaris, et al.: "Protecting CRT RSA against fault and power side channel attacks," in *Proc. VLSI 2012*, pp. 159–164 (Aug. 2012).

[2] Vlastímil Klíma and Tomáš Rosa: "Further results and considerations on side channel attacks on RSA," in *Proc. CHES 2002*, pp. 244–259 (Aug. 2002).

[3] N. Ito, et al.: "High Level synthesis from binary code linked with multiple precision arithmetic library to RSA encrypt/decrypt circuit," (in Japanese) in *Proc. Convention of IPSJ Kansai Section*, A-04 (Sept. 2015).

[4] N. Ishiura, H. Kanbara, and H. Tomiyama: "ACAP: Binary synthesizer based on MIPS object codes," in *Proc. ITC-CSCC 2014*, pp. 725–728 (July 2014).

[5] S. Peter and T. Givargis: "Towards a timing attack aware high-level synthesis of integrated circuits," in *Proc. ICCD*, pp. 452–455 (Nov. 2016).