電力解析攻撃/故障利用攻撃耐性 RSA 復号回路の高位合成

High-Level Synthesis of Power Analysis Attack and Fault Attack Resistant RSA Decryption Circuit

太田 小百合　Sayuri Ota
由良 駿 †　Suguru Yura
石浦 菜岐佐　Nagisa Ishiura

関西学院大学 理工学部　School of Science and Technology, Kwansei Gakuin University

1　はじめに

近年増加する M2M 機器に暗号化機能を実装する場合, 厳しい消費電力制約を満たすために, その処理のハードウェア化が一つの選択肢になる. [1] では, 高位合成により RSA 暗号処理回路を設計しているが, セキュリティ確保のためには, サイドチャネル攻撃への対策も必須となる. 本稿では, [2] のアルゴリズムに基づき, 電力解析攻撃と故障利用攻撃に耐性のある RSA 復号回路を高位合成により設計する手法を提案する.

2　Fournaris の RSA 復号アルゴリズム

Fournaris のアルゴリズム [2] は RSA の電力解析攻撃と故障利用攻撃に対する脆弱性の総合的な解決を図ったものであり, 一般的な単純/差分電力解析攻撃, Bellcore 攻撃, KQ 攻撃, YLMH 攻撃に耐性を持つ. アルゴリズムは 図 1 に示すように, マスキングを伴う Montgomery 冪剰余算と, 故障挿入の検出に基づいている.

3　攻撃に耐性のある RSA 復号回路の高位合成

本稿では, [1] と同様, 多倍長整数演算ライブラリ GMP を高位合成用に書き換えたものを用いて, 図 1 のアルゴリズムを C 言語で記述し, バイナリ合成システム ACAP [3] で合成する. 例えば, $t = S_0 \cdot S_1 \bmod N$ は, 乗算と剰余算を行う関数 `mpz_mul` と `mpz_mod` を用いて,

```
mpz_mul(&tmp, &s0, &s1);
mpz_mod(&t, &tmp, N);
```

のように書ける. 本稿の設計記述は, データを格納する領域サイズを除いて復号処理のビット数に依存しない. ACAP は, MIPS 用 GCC で得られるアセンブリを CDFG に変換し, ここから Verilog HDL を生成する.

4　合成結果

実装したプログラムの動作を MIPS ソフトコアで確認した後, ACAP で合成した. GCC には -O2 を指定し, ALU 3 個, 乗算器 3 個でスケジューリングを行った. チェイニングは行っていない. 論理合成は Xilinx ISE (14.7) で FPGA (Spartan-6) をターゲットに行った. 合成結果を 表 1 に示す. “RSA” は攻撃に耐性のない [1] の回路であり, “SRR” が本稿の回路である. “cycles” は, 128 ビット復号処理のサイクル数である. MIPS に比べ約 7.9 倍の LUT 数で, 約 5.2 倍の高速化が実現した. SRR は RSA に比べて, 約 1.5 倍の LUT 数で, 処理時間は約 10.5 倍になった.

5　むすび

本稿では, 耐電力解析攻撃/故障利用攻撃 RSA 復号回路を合成した. 実行の高速化と回路規模の削減が今後の課題である.

† 現在, 防衛省

耐攻撃 Montgomery 冪剰余算

Function: FSCAME
Input: $c, b, b^{-1}, e = (1, e_{t-2}, ...e_0), M$
Output: (s_0, s_1, s_2, s_4)
// $s_0 = b^e \cdot c^e \bmod M$,　$s_1 = b^{\bar{e}+1} \cdot c^{\bar{e}+1} \bmod M$
// $s_2 = b^{2^t} \cdot c^{2^t} \bmod M$,　$s_4 = b^{-e} \bmod M$

$T = R^2 \bmod M$;
$b_R = b \cdot R \bmod M$;
$b_{R-1} = b^{-1} \cdot R \bmod M$;
$R = 2^{n+2}$;
$s_0 = s_1 = b_R$;
$T_R = T \cdot c \cdot R^{-1} \bmod M$;
$s_2 = b_R \cdot T_R \cdot R^{-1} \bmod M$;
$s_3 = s_4 = s_5 = b_{R-1}$;
for $(i = 0$ to $t-1)$ {
　if $(e_i = 1)$ {
　　$s_0 = s_0 \cdot s_2 \cdot R^{-1} \bmod M$;
　　$s_4 = s_4 \cdot s_3 \cdot R^{-1} \bmod M$;
　} **else** {
　　$s_1 = s_1 \cdot s_2 \cdot R^{-1} \bmod M$;
　　$s_5 = s_5 \cdot s_3 \cdot R^{-1} \bmod M$;
　}
　$s_2 = s_2^2 \cdot R^{-1} \bmod M$;
　$s_3 = s_3^2 \cdot R^{-1} \bmod M$;
}
$s_0 = s_0 \cdot b^{-1} \cdot R^{-1} \bmod M$;
$s_1 = s_1 \cdot c \cdot R^{-1} \bmod M$;
$s_2 = s_2 \cdot 1 \cdot R^{-1} \bmod M$;
$s_4 = s_4 \cdot b \cdot R^{-1} \bmod M$;
if (i and e are not modified **and**
　　$s_0 \cdot s_1 \cdot R^{-1} \bmod M = s_2 \cdot 1 \cdot R^{-1} \bmod M$)
　{ **return** (s_0, s_1, s_2, s_4); } **else** { **return** error; }

RSA 復号

Input: $c, b, b^{-1}, p, q, d_p, d_q, i_q = q^{-1} mod p, N$
Output: $c^d mod N$

$(s_0^p, s_1^p, s_2^p, s_4^p) = \text{FSCAME}(c, b, b^{-1}, d_p, p)$;
$(s_0^q, s_1^q, s_2^q, s_4^q) = \text{FSCAME}(c, b, b^{-1}, d_q, q)$;
$s_0 = s_0^q + q \cdot ((s_0^p - s_0^q) \cdot i_q \bmod p)$;
$s_1 = s_1^q + q \cdot ((s_1^p - s_1^q) \cdot i_q \bmod p)$;
$s_2 = s_2^q + q \cdot ((s_2^p - s_2^q) \cdot i_q \bmod p)$;
$s_4 = s_4^q + q \cdot ((s_4^p - s_4^q) \cdot i_q \bmod p)$;
if $(S_0 \cdot S_1 \bmod N = S_2$ **and** p, q not modified)
　{ **return** $(S_0 \cdot S_4 \bmod N)$; } **else** { **return** error; }

図 1　Fournaris のアルゴリズム [2].

表 1　合成結果.

code		resisters	LUTs	delay [ns]	cycles
RSA	(MIPS)	1,821	3,788	16.078	431,013
	(HW)	1,508	19,620	21.206	68,262
SRR	(MIPS)	1,821	3,788	16.078	3,745,369
	(HW)	2,658	29,986	25.055	714,635

参考文献

[1] 伊藤, 竹林, 神原, 石浦: “多倍長整数演算ライブラリをリンクしたバイナリコードからの RSA 暗号回路の高位合成,” 情処関西支部大会, A-04 (Sept. 2015).

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

[3] 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).