Computer Security Symposium 2018
22 - 25 October 2018

DPA 対策マスクの変換アルゴリズムの改良と認証方式への適用

駒野 雄一[1]　清水 秀夫[1]　三宅 秀享[1]

概要：DPA などの電力解析に対して、内部処理に算術加算を含むアルゴリズムを保護するためには、算術マスクとブールマスクを相互変換して使用して、処理データを攪乱する必要がある。Coron らは、従来とは異なる原理で算術マスクをブールマスクに変換する手法を提案した。本稿は、Coron らの変換手法を改良し、内部処理に算術加算を含む認証方式への適用結果を報告する。

キーワード：サイドチャネル攻撃, マスク変換, IoT, 組込み端末

Integrative Acceleration of First-Order Boolean Masking for Embedded IoT Devices

Yuichi Komano[1]　Hideo Shimizu[1]　Hideyuki Miyake[1]

Abstract: Physical attacks, especially side-channel attacks, are threats to IoT devices which are located everywhere in the field. For these devices, the authentic functionality is important so that the IoT system becomes correct, and securing this functionality against side-channel attacks is one of our emerging issues. Toward that, Coron et al. gave an efficient arithmetic-to-Boolean mask conversion algorithm which enables us to protect cryptographic algorithms including arithmetic operations, such as hash functions, from the attacks. Recently, Biryukov et al. improved it by locally optimizing subroutines of the conversion algorithm. In this paper, we revisit the algorithm. Unlike Biryukov et al., we improve the Coron et al.'s algorithm with integrative optimizations over the subroutines. The gains against these algorithms are about 22.6% and 7.0% in the general setting. We also apply our algorithm to HMAC-SHA-1 and have an experiment to show that the implementation on a test vehicle smartcard leaks no sensitive information with the ISO/IEC17825 test.

Keywords: side-channel attack, mask conversion, IoT, embedded device

1. はじめに

Internet of Things (IoT) 端末は、フィールドに散在して、センサデータや制御命令を互いに通信している。このような端末は、攻撃者が自由に入手できるために攻撃にさらされ易いうえ、低コスト化のためにセキュリティ実装が不十分となりやすいので、セキュリティの確保が急務となっている。特に、電力解析 [4], [9], [10] を含むサイドチャネル攻撃は、これらの端末に対する深刻な脅威となりうる。

IoT 端末のセキュリティ機能に関しては、守秘や匿名性などに比べて、通信内容や通信端末の真正性を保証する認証機能が重要である。そこで、本稿は、IoT 端末への応用を想定して、サイドチャネル攻撃に対して安全な認証機能を実現するための、対策技術を提案する。

1.1 1 次のブールマスクと算術マスク

ハッシュ関数ベースのメッセージ認証子 (HMAC, [2], [17]) は、様々なシステムで利用されている。ブロック暗号 AES[16] と異なり、Secure Hash Algorithm (SHA, [18]) の各アルゴリズムは、ブール演算に加えて、算術加算を内部処理に含む。SHA を利用した HMAC に対しても、サイドチャネル攻撃が可能となることが知られてい

[1] 株式会社東芝 研究開発センター
Corporate R&D Center, Toshiba Corporation

る [1], [12], [14]。

サイドチャネル攻撃から HMAC を保護するためには、ブール演算と算術演算で処理されるデータを、それぞれ異なる型のランダムマスクで撹乱すればよい。そのためには、それぞれの型のランダムマスクを、互いに変換しあう技術が必要である。

Goubin[6] は、ブールマスクから算術マスクへの (B2A) 変換法とその逆向きの (A2B) 変換法を提案した。それ以来、さらに効率の良い変換法や、高次ランダムマスクの変換法などが提案されている。しかし、1 次ランダムマスクの B2A 変換に関しては、現時点において Goubin の手法が最も効率が良い手法である。

A2B 変換に関しては、Coron ら [5] が、変換の基盤となる算術加算アルゴリズムを見直すことにより、効率の良い変換法を提案した。Ripple-carry adder に基づいて A2B 変換法を構築した Goubin のアプローチとは異なり、Coron らは Kogge-Stone carry look-ahead adder[11] を利用した。この手法により、算術加算のビット幅を k とするとき、変換に要する計算量を $\mathcal{O}(k)$ から $\mathcal{O}(\log k)$ へと減らすことができた。さらに、Coron らは、ブールマスクで撹乱された二つのデータから、その一方のブールマスクで撹乱された和を計算するための、マスク付き算術加算アルゴリズムも提案した。

その後、Biryukov ら [3] は、Coron らのマスク付き算術加算の改良方式を提案した。Biryukov らは、形式手法を用いて Coron らの方式のサブルーチンを個別に最適化して、高速に処理できるマスク付算術加算アルゴリズムを提案した。Biryukov らは、軽量暗号 SIMON, SPECK, RECTANGLE に改良方式を適用し、その効果も示した。

1.2 本稿の成果

本稿は、Coron らの変換法とマスク付き算術加算アルゴリズムの改良方式を提案する。個別のサブルーチンを最適化した Biryukov らとは異なり、我々は、複数のサブルーチンで引き渡されるマスクの処理量を減らして、統合的なアプローチにより高速化を図る。

提案法は、Coron らと Biryukov らの変換法に比べて、変換に要する計算量 (表 1 の $\log_2 k$ の係数) を 7 および 1 減らすことができた。Coron らの変換法に対して、Biryukov らは、OR(orn) 演算を必要とするが、AND と XOR の演算数を減らす変換法を提案した。一方、我々の変換法は、Coron らの変換法に対して、XOR の演算数を大幅に減らし、全体として計算量を抑えた (表 2)。例えば、$k = 32$、すなわち 32 ビット加算器を仮定するとき、Coron らと Biryukov らの変換法に対して、提案法は 22.6% および 7.0% の高速化が可能となる[*1]。

さらに、我々は、HMAC-SHA-1 への提案法の適用例を示す。既存の変換法を用いた HMAC-SHA-1 実装よりも、提案法を用いることで、演算数を 17.1% および 4.9% だけ減らすことができる (表 3)。我々は、xorshift[13] をマスク生成関数として利用し、IPA が作成した IC カードテストビークル [7] の上に HMAC-SHA-1 を実装して、ISO/IEC 17825[8] の評価により脆弱な処理が含まれないことを確認した。

1.3 構成

本稿の構成は、以下のとおり。2 節では、関連研究を紹介する。3 節において、提案法のアイデアとアルゴリズムを紹介する。その後、4 節では、HMAC-SHA-1 への提案法の適用結果と、IPA カードテストビークルカードを利用した耐性評価結果を示す。そして、5 節では、考察を述べる。最後に、6 節でまとめを述べる。

2. 関連研究

ランダムマスクの変換法に関する既存研究を紹介する。

2.1 Goubin のマスク変換法 [6]

はじめに、Goubin の B2A 変換法 [6] を復習する。この変換は、Algorithm 1 に示すように、$x' = x \oplus r$ と r から、$A = x - r$ を計算する。4 節の実験では、この変換を B2A 変換として利用する。文献 [6] では、Goubin は逆向きの A2B 変換法も提案したが、本稿では説明を省略する。

Algorithm 1 Goubin's Boolean-to-arithmetic Conversion Algorithm

Input: $x', r \in \{0,1\}^k$ such that $x' = x \oplus r$ for secret $x \in \{0,1\}^k$
Output: $A \in \mathbb{F}_{2^k}$ and r such that $A = x - r$

1: $\gamma \leftarrow \{0,1\}^k$
2: $t \leftarrow x' \oplus \gamma$
3: $t \leftarrow t - \gamma$
4: $t \leftarrow t \oplus x'$
5: $\gamma = \gamma \oplus r$
6: $a \leftarrow x' \oplus \gamma$
7: $a \leftarrow a - \gamma$
8: $a \leftarrow a \oplus t$
9: **return** a, r

2.2 Coron らのマスク変換法 [5]

Coron ら [5] は、1 次 DPA への対策に適用可能な、効率的な A2B 変換法を提案した。Goubin は Ripple-carry adder に基づいて A2B 変換法を構成していたが、Coron らは Kogge-Stone carry look-ahead adder[11] に基づく A2B 変換法を構成した。Coron らは、A2B 変換法に加えて、

[*1] 本研究とは独立に、Jungk ら [19] は、Biryukov らのマスク付き算術加算アルゴリズムを改良することで、さらに効率の良いマスク付き算術加算アルゴリズムを提案した。本稿では、Jungk らのアルゴリズムの構成や安全性は述べず、結果のみを比較する。

ブールマスクが施された二つのデータを入力として、それらの一方のブールマスクで保護された和を出力する、Kogge-Stone マスク付き算術加算も提案した。

Coron らは、文献 [5] において、Kogge-Stone ベースの A2B 変換法と Kogge-Stone マスク付き算術加算のそれぞれを利用して、HMAC-SHA-1 の 1 次 DPA 対策を実装評価した。Coron らの結果によると、A2B 変換法を用いる場合に比べて、マスク付き算術加算を用いると 2.28 倍の処理時間が必要となり、効率が悪い。そこで、本節では Kogge-Stone ベースの A2B 変換法のみを Algorithm 2 に示す。

Algorithm 2 Kogge-Stone Arithmetic-to-Boolean Conversion [5]

Input: $A, r \in \{0,1\}^k$ and $n = \max(\lceil \log_2(k-1) \rceil, 1)$ such that $A = x - r \in \mathbb{F}_{2^k}$
Output: x' such that $x' \oplus r = A + r \mod 2^k$
1: Let $s \leftarrow \{0,1\}^k, t \leftarrow \{0,1\}^k, u \leftarrow \{0,1\}^k$
2: $P' \leftarrow A \oplus s$
3: $P' \leftarrow P' \oplus r$
4: $G' \leftarrow s \oplus ((A \oplus t) \wedge r)$
5: $G' \leftarrow G' \oplus (t \wedge r)$
6: **for** $i := 1$ to $n-1$ **do**
7: $H \leftarrow \mathsf{SecShift}(G', s, t, 2^{i-1})$
8: $U \leftarrow \mathsf{SecAnd}(P', H, s, t, u)$
9: $G' \leftarrow \mathsf{SecXor}(G', U, u)$
10: $H \leftarrow \mathsf{SecShift}(P', s, t, 2^{i-1})$
11: $P' \leftarrow \mathsf{SecAnd}(P', H, s, t, u)$
12: $P' \leftarrow P' \oplus s$
13: $P' \leftarrow P' \oplus u$
14: **end for**
15: $H \leftarrow \mathsf{SecShift}(G', s, t, 2^{n-1})$
16: $U \leftarrow \mathsf{SecAnd}(P', H, s, t, u)$
17: $G' \leftarrow \mathsf{SecXor}(G', U, u)$
18: $x' \leftarrow A \oplus 2G'$
19: $x' \leftarrow x' \oplus 2s$
20: **return** x'

Algorithm 2 は、Shift と AND と XOR を、ランダムマスクを用いて安全に計算するサブルーチン SecShift,SecAnd,SecXor を使用する [5]。

2.3 Biryukov らのマスク付き算術加算 [3]

Biryukov ら [3] は、Coron らのサブルーチンのうち、AND と OR を安全に計算するサブルーチンの最適な構成を網羅的に探索し、それらを利用したマスク付き算術加算とマスク付き算術減算を提案した。

Algorithm 3 に、Biryukov らのマスク付き算術加算アルゴリズム [3] を示す。このアルゴリズムは、SecAnd を改良した SecAnd2 と、二つのサブルーチン SecShift2,SecXor2 を使用する。

Algorithm 3 Biryukov et al.'s Masked Addition Algorithm [3]

Input: $x_1, x_2, y_1, y_2 \in \{0,1\}^k$ and $n = \max(\lceil \log_2(k-1) \rceil, 1)$ such that $x = x_1 \oplus x_2$ and $y = y_1 \oplus y_2$
Output: z_1, z_2 such that $z = z_1 \oplus z_2 = (x + y) \mod 2^k$
1: $p_1, p_2 \leftarrow \mathsf{SecXor2}(x_1, x_2, y_1, y_2)$
2: $g_1, g_2 \leftarrow \mathsf{SecAnd2}(x_1, x_2, y_1, y_2)$
3: $g_1, g_2 \leftarrow ((g_1 \oplus x_2) \oplus g_2, x_2)$
4: **for** $i := 1$ to $n-1$ **do**
5: $h_1, h_2 \leftarrow \mathsf{SecShift2}(g_1, g_2, 2^{i-1})$
6: $u_1, u_2 \leftarrow \mathsf{SecAnd2}(p_1, p_2, h_1, h_2)$
7: $g_1, g_2 \leftarrow \mathsf{SecXor2}(g_1, g_2, u_1, u_2)$
8: $h_1, h_2 \leftarrow \mathsf{SecShift2}(p_1, p, 2^{i-1})$
9: $h_1, h_2 \leftarrow ((h_1 \oplus x_2) \oplus h_2, x_2)$
10: $p_1, p_2 \leftarrow \mathsf{SecAnd2}(p_1, p_2, h_1, h_2)$
11: $p_1, p_2 \leftarrow ((p_1 \oplus y_2) \oplus p_2, y_2)$
12: **end for**
13: $h_1, h_2 \leftarrow \mathsf{SecShift2}(g_1, g, 2^{n-1})$
14: $u_1, u_2 \leftarrow \mathsf{SecAnd2}(p_1, p_2, h_1, h_2)$
15: $g_1, g_2 \leftarrow \mathsf{SecXor2}(g_1, g_2, u_1, u_2)$
16: $z_1, z_2 \leftarrow \mathsf{SecXor2}(y_1, y_2, x_1, x_2)$
17: $z_1, z_2 \leftarrow (z_1 \oplus 2g_1, z_2 \oplus 2g_2)$
18: **return** z_1, z_2

3. 提案

本節では、まず、Coron らの A2B 変換法とマスク付き算術加算を改良するアイデアを紹介する。そして、具体的な処理手順を与えて、既存方式と処理量を比較する。

3.1 構成のアイデア

Coron らの方式を改良するにあたり、二つのアイデアを採用した。

一つ目は、Coron らの変換法における第三のランダムマスク (Algorithm 2 の t) を他の二つのランダムマスクの XOR 値に (Algorithm 5 の 2 行目のように) 置き換えることで、変換に必要となるランダムマスクの数を一つ減らした。これにより、高次 DPA への耐性は 1 次元分だけ下がってしまう。しかし、1 次 DPA への対策としては、内部データが一つまたは独立な二つのランダムマスクで撹乱されていれば十分であり、本アイデアを採用しても 1 次 DPA への耐性は損なわれない。このアイデアにより、ランダムマスクの数を削減できるだけでなく、それらの XOR 演算 (Algorithm 2 の Step 12,13 など) も削減できる。

二つ目は、サブルーチンをまたいでマスク処理の回数を減らす。Coron らの変換法では、SecShift の後に SecAnd を実行する。このとき、SecShift と SecAnd のそれぞれで、出力にかかるランダムマスクの値を調整するために、XOR 演算が余分に実行される。本アイデアは、SecShift と SecAnd を統合して (Algorithm 4 の SecShiftAnd)、XOR の演算数を減らす。

それらに加えて、Coron らの変換法の初期化ステップ

(Algorithm 2 の Step 3 から 5) を見直して、XOR の演算数を減らした。

これらのアイデアにより Coron らの変換法の改良が可能のように思えるが、これだけでは不十分である。実際、ランダムマスクの数を減らしたことにより、そのままではランダムマスクがキャンセルされることがある。そのため、Coron らの変換法とは異なる順番でランダムマスクを作用させる必要がある。そこで、次節に示すように、処理の対称性を崩し、二つの系列 (Step 8 から 11 と、13 から 15) を用意して、変換を通じてランダムマスクがキャンセルされないように工夫した。

3.2 提案アルゴリズム

Algorithm 4 に示すように、二つのサブルーチンを SecShiftAnd に統合して SecShift 内の XOR 演算を省略し、A2B 変換の効率を上げる。

Algorithm 4 SecShiftAnd

Input: $x'_1, s_1, j, x'_2, s_2, u$ such that $x'_i, s_i, u \in \{0,1\}^k$ and $j \in \mathbb{Z}$ where $x'_i = x_i \oplus s_i$
Output: z' such that $z' = ((x_1 << j) \wedge x_2) \oplus u$
1: $y \leftarrow x'_1 << j$
2: $s' \leftarrow s_1 << j$
3: $z' \leftarrow u \oplus (x'_2 \wedge y)$
4: $z' \leftarrow z' \oplus (x'_2 \wedge s')$
5: $z' \leftarrow z' \oplus (s_2 \wedge y)$
6: $z' \leftarrow z' \oplus (s_2 \wedge s')$
7: **return** z'

Algorithm 5 に、提案する A2B 変換法の手順を示す。前節で説明したとおり、2 つの処理列を用意して、i(ループカウンタ) または n を条件として処理を選択する。これらの条件は公開情報であり、選択処理により内部の秘密情報が (サイドチャネル攻撃により) 漏洩する恐れはないことに注意しよう。

3.3 既存方式との比較

表 1 に、既存の変換法と提案法のそれぞれが必要とする演算数をまとめる。文献 [3] では、Biryukov らはマスク付き算術加算アルゴリズムを与えており、A2B 変換法を与えていない。しかし、付録 A.1 の Algorithm 7 に示すように、Biryukov らのマスク付き算術加算アルゴリズムから、A2B 変換法を構成できる。本節では、Algorithm 7 を Biryukov らの A2B 変換とよぶことにする。

この表から、提案法は既存方式よりも少ない演算数で処理できることが分かる。例えば、$k = 32$ の場合には、Coron らと Biryukov らの変換法に比べて、提案法は演算数を 31 回 ($\approx 22.6\%$) および 8 回 ($\approx 7.0\%$) 削減できる。

次に、変換法の詳細を見てみよう。表 2 は、$k = 32$ のときの必要演算数の詳細をあらわす。Coron らの変換法に比べて、Biryukov らの変換法は、orn 演算を 20 回余分に必要とするが、and と eor 演算を減らすことができ、全体として 23 回の演算を減らすことができる。

Algorithm 5 Our Arithmetic-to-Boolean Conversion

Input: $A, r \in \{0,1\}^k$ and $n = \max(\lceil \log_2(k-1) \rceil, 1)$ such that $A = x - r \in \mathbb{F}_{2^k}$
Output: x' such that $x' \oplus r = A + r \mod 2^k$
1: $s, u \leftarrow \{0,1\}^k$
2: $t \leftarrow s \oplus u$
3: $P' \leftarrow P' \oplus r$
4: $G' \leftarrow t \oplus (P' \wedge r)$
5: $G' \leftarrow G' \oplus (s \wedge r)$
6: $P' \leftarrow A \oplus s$
7: **for** $i := 1$ to $n-1$ **do**
8: **if** i is odd **then**
9: $U \leftarrow \mathsf{SecShiftAnd}(G', t, 2^{i-1}, P', s, u)$
10: $G' \leftarrow G' \oplus U$
11: $P' \leftarrow \mathsf{SecShiftAnd}(P', s, 2^{i-1}, P', s, t)$
12: **else**
13: $U \leftarrow \mathsf{SecShiftAnd}(G', s, 2^{i-1}, P', t, u)$
14: $G' \leftarrow G' \oplus U$
15: $P' \leftarrow \mathsf{SecShiftAnd}(P', t, 2^{i-1}, P', t, s)$
16: **end if**
17: **end for**
18: **if** n is odd **then**
19: $U \leftarrow \mathsf{SecShiftAnd}(G', t, 2^{n-1}, P', s, u)$
20: **else**
21: $U \leftarrow \mathsf{SecShiftAnd}(G', s, 2^{n-1}, P', t, u)$
22: **end if**
23: $G' \leftarrow G' \oplus U$
24: $x' \leftarrow A \oplus 2G'$
25: **if** n is odd **then**
26: $x' \leftarrow x' \oplus 2s$
27: **else**
28: $x' \leftarrow x' \oplus 2t$
29: **end if**
30: **return** x'

一方、提案法は、Coron らの変換法に対して、eor 演算 (だけ) を 31 回削減できる。提案法は、Biryukov らの変換法よりも多くの and を必要とするが、eor 演算を大幅に減らしているため、全体として効率が良い。

4. 実験

次に、提案した A2B 変換法を用いて、1 次 DPA 耐性をもつ HMAC-SHA-1 を実装する。Coron らの結果を踏まえ、本稿はマスク付き算術加算ではなく、Goubin の B2A 変換とそれぞれの A2B 変換を利用する。すなわち、(HMAC-)SHA-1 の内部データは、基本的にはブールマスクで撹乱される。そして、各ラウンドにおいて、高速処理可能な Goubin の B2A で算術マスクに変換して加算処理を実行し、A2B 変換を一度だけ適用してブールマスクに戻す。

表 1　各変換法の必要演算数

Algorithm	rand	$k=8$	$k=16$	$k=32$	$k=64$	k
Coron et al.'s conversion	3	81	109	137	165	$28\log_2 k-3$
Biryukov et al.'s conversion	2	70	92	114	136	$22\log_2 k+4$
Our conversion	2	64	85	106	127	$21\log_2 k+1$
Coron et al.'s addition	2	88	116	144	172	$28\log_2 k+4$
Biryukov et al.'s addition	0	70	92	114	136	$22\log_2 k+4$
Our addition	1	69	90	111	132	$21\log_2 k+6$
Jungk et al.'s addition	0	68	87	106	125	$19\log_2 k+11$

Algorithm 6 Our Masked Addition Algorithm

Input: $x', y', r, s \in \{0,1\}^k$ and $n = \max(\lceil \log_2(k-1) \rceil, 1)$ such that $x' = x \oplus r$ and $y' = y \oplus s$
Output: z' such that $z \oplus r = x + y \mod 2^k$

1: $u \leftarrow \{0,1\}^k$
2: $t \leftarrow s \oplus u$
3: $z' \leftarrow x' \oplus y'$
4: $P' \leftarrow z' \oplus r$
5: $z' \leftarrow z' \oplus s'$
6: $G' \leftarrow \mathsf{SecAnd}(x', y', s, r, t)$
7: **for** $i := 1$ to $n-1$ **do**
8: **if** i is even **then**
9: $U \leftarrow \mathsf{SecShiftAnd}(G', t, 2^{i-1}, P', s, u)$
10: $G' \leftarrow G' \oplus U$
11: $P' \leftarrow \mathsf{SecShiftAnd}(P', s, 2^{i-1}, P', s, u)$
12: **else**
13: $U \leftarrow \mathsf{SecShiftAnd}(G', s, 2^{i-1}, P', t, u)$
14: $G' \leftarrow G' \oplus U$
15: $P' \leftarrow \mathsf{SecShiftAnd}(P', t, 2^{i-1}, P', t, s)$
16: **end if**
17: **end for**
18: **if** n is even **then**
19: $U \leftarrow \mathsf{SecShiftAnd}(G', t, 2^{n-1}, P', s, u)$
20: **else**
21: $U \leftarrow \mathsf{SecShiftAnd}(G', s, 2^{n-1}, P', t, u)$
22: **end if**
23: $G' \leftarrow G' \oplus U$
24: $z' \leftarrow z' \oplus 2G'$
25: **if** n is even **then**
26: $z' \leftarrow z' \oplus 2s$
27: **else**
28: $z' \leftarrow z' \oplus 2t$
29: **end if**
30: **return** z'

表 2　必要演算数の詳細 ($k=32$ の場合)

Algorithm	and	orn	sft	eor	total
Coron et al.'s conversion	38	0	20	79	137
Biryukov et al.'s conversion	20	20	20	54	114
Our conversion	38	0	20	48	106

4.1　実験環境

実験では、HMAC-SHA-1[2], [17], [18] を Keil MDK-lite for Windows, version 5.24.1 を用いて開発した。C 言語で記述したソースコードは、最適化オプション O3 をつけて、armcc v5.06 update 5 (build 528) でコンパイルし、アセンブラのコードを生成した。そして、コンパイル時にマスク処理が取り除かれていないことをアセンブラレベルで確認し、必要な修正を加えた。最後に、アセンブラのコードを armasm v5.06 update 5 (build 528) でコンパイルして、実行ファイルを作成した。

次に、実行ファイルを IPA が作成した IC カードテストビークル [7] にダウンロードした。IC カードテストビークルは、28MHz で動作する ARM7 ベースの SC100 と、512KB のフラッシュメモリ、18KB の RAM を搭載しており、通信インタフェースは ISO7816-3 (T=0) に準拠している。

そして、SASEBO-W[15] とデジタルオシロスコープ LECROY WavePro715Zi を使用して、IC カードテストビークルで HMAC-SHA-1 を実行しているときの消費電力波形を収集した。IC カードテストビークルには、外部から 2.5V の電圧と 3.57MHz のクロックを供給し、Windows ベースの PC で IC カードテストビークルを制御した。オシロスコープのサンプリングレートは 1G Samples/s に設定した。

4.2　HMAC-SHA-1 の実装

4.2.1　Python によるプロトタイプ

HMAC-SHA-1 を C 言語で実装する前に、Python で一から実装した。表 3 に、各変換法を用いた場合に必要となる乱数と演算の数をまとめる。演算数に関しては、ARM プロセッサで 1 クロックサイクルで実行可能な、add, sub, and, or, eor, orr(Biryukov らの変換法のみ), shift, rot の数の和をまとめた。

表 3　HMAC-SHA-1 の Python 実装に必要な乱数と演算の数

Implementation	♯rand	♯ops	ratio
Without countermeasure	0	4,004	1
With Coron et al.'s conversion	313	63,358	15.82
With Biryukov et al.'s conversion	72	55,173	13.78
With our conversion	72	52,493	13.11

表から、無対策実装に対して、1 次 DPA 対策を施すと必要な演算数が 10 倍以上となることが分かる。中でも、提案法を用いた場合が最も演算数が少なく、Coron ら

と Biryukov らの変換法を用いた場合に比べて、演算数を 10,865(≈ 17.1%) および 2,680(≈ 4.9%) 削減できる。

乱数に関しては、Biryukov らと我々の変換法は 72 個のランダムマスクを使用する。内訳は、各ハッシュ関数の 5 ワードのそれぞれで 5 個 (2 回のハッシュ処理で 10 個)、各 16 ワードのメッセージのそれぞれで 16 個 (3 回のダイジェスト計算で 48 個)、最後のダイジェストで 11 個 (最初の 5 ワードは inner hash の出力であり、既にマスクが施されている)、B2A 変換用に 1 個、A2B 変換用に 2 個である。

4.2.2 IPA カードテストビークルへの C 実装

次に、無対策実装と 1 次 DPA 対策実装のそれぞれについて、C 言語で実装してアセンブラでマスク処理を修正し、IC カードテストビークルカードに搭載した。ランダムマスク用の乱数は、xorshift[13] で生成した。表 4 に、各実装で必要なサイクル数をまとめた。

表 4 IC カードテストビークル上での HMAC-SHA-1 の実行サイクル数

Implementation	♯cycles	ratio
Without countermeasure	12,391	1
With Coron et al.'s conversion (opt0)	68,711	5.55
With Biryukov et al.'s conversion (opt0)	66,344	5.35
With our conversion (opt0)	63,546	5.13
With Coron et al.'s conversion (opt1)	41,914	3.38
With Biryukov et al.'s conversion (opt1)	40,913	3.30
With our conversion (opt1)	39,471	3.19
With Coron et al.'s conversion (opt2)	29,150	2.35
With Biryukov et al.'s conversion (opt2)	28,629	2.31
With our conversion (opt2)	27,862	2.25

表において、"opt0" は、固定鍵に対する圧縮処理は事前処理しておくが、80 ラウンドの全てでマスク処理を施した実装をあらわす。一方、"opt1" と "opt2" は、SHA-1 の中間の 40 または 60 ラウンドは、マスク処理を施さない実装をあらわす。"opt0" の対策実装は無対策実装の 5 倍以上のサイクルを必要とする。仮に、固定鍵に対する 2 つの圧縮処理を事前計算しない場合には、10 倍程度のサイクルが必要になることとなり、表 3 の結果に一致する。Coron らの変換法を用いた場合に比べて、提案法を用いると、サイクル数を、5,165(≈ 7.5%, "opt0"), 2,443(≈ 5.8%, "opt1"), 1,288(≈ 4.4%, "opt2") 削減できる。Biryukov らの変換法を用いた場合に比べると、2,798(≈ 4.2%, "opt0"),1,442 (≈ 3.5%, "opt1"),767 (≈ 2.7%, "opt2") 削減できる。

4.3 1 次 DPA 耐性の確認

IC カードテストビークルに、提案した変換法 (opt2) を利用した HMAC-SHA-1 の 1 次 DPA 対策を実装し、ランダムな鍵に対して 100,000 個の消費電力波形を収集した。

鍵 key とメッセージ msg を入力とする HMAC-SHA-1 は、$\mathsf{sha1}((key \oplus opad)||\mathsf{sha1}((key \oplus ipad)||msg))$ を計算する。ただし、$ipad$ と $opad$ は、それぞれ、定数 0x3636$\cdots$36 と 0x5c5c$\cdots$5c である。既に説明したとおり、各 sha1 の実行において、固定鍵の圧縮処理は事前計算した。実装したソフトウェアは、事前計算した圧縮値を初期ベクトルとみなし、msg を入力として inner hash の 2 回目の圧縮処理を実行する。耐性を評価するにあたり、この圧縮処理の最初のラウンドの出力値を評価対象に設定した。

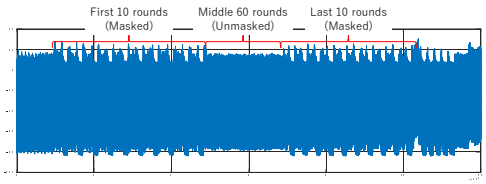

図 1 $\mathsf{sha1}(msg)$ with IV= $\mathsf{sha1}(key \oplus ipad)$ の平均波形

図 1 は、最初の 1000 個の消費電力波形の平均波形であり、横軸と縦軸はそれぞれ時間と消費電力 (電圧値) をあらわす。 図に示したように、消費電力波形から、各ラウンドの繰り返し処理の様子をみることができる。各消費電力波形から、評価対象となる初段を含む部分波形を切り出してきて、処理のタイミングを揃えるための事前処理を施した。

次に、100,000 個の部分波形を評価対象の最初のバイト値のハミング重みが 16 より大きいグループと小さいグループに分類し、ISO/IEC 17825[8] の評価を実行した。図 2 は評価結果であり、横軸と縦軸はそれぞれ時間と T 検定値をあらわす。この図から、いずれの点も T 検定値が 4.5 を超えることがなかったため、HMAC-SHA-1 の 1 次 DPA 対策が有効であることが確認できた。

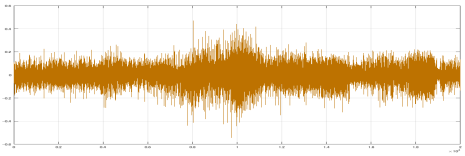

図 2 ISO/IEC 17825 での評価結果

5. 考察

3.3 節で説明したとおり、提案法は既存の方式よりも少ない演算数で A2B 変換を実行可能である。さらに、4.3 節で確認したとおり、1 次 DPA への耐性も保証できる。

5.1 処理分岐

既存の A2B 変換と異なり、提案した変換法は、Algorithm 5 に示したとおり、i と n を条件とする分岐処理を含む。これらの条件は秘密を含まないため、サイドチャネル情報からも秘密データの漏洩は起こらない。しかし、そのまま

では処理時間の増加に繋がってしまう。コードサイズとのトレードオフは生じるが、ループを展開してアンロール処理とすることで、処理時間の増加を抑えることが可能である。前節で紹介した実験では、アンロール処理で A2B 変換を実装した。

5.2 高次マスクへの拡張

本稿は、1 次 DPA に対抗可能な、1 次ランダムマスクの A2B 変換法を議論した。文献 [5], Section 6 の議論に従えば、Coron らと同様に、提案法も高次ランダムマスクの変換へと拡張可能である。

6. まとめ

本稿は、Coron らの A2B 変換法の改良方式を提案した。IPA が作成した IC カードテストビークルを利用した実験では、提案した変換法が 1 次 DPA の対策として有効であることが確認できた。プロファイル型などの洗練された攻撃への耐性評価や対策実装は今後の課題である。

謝辞

本研究は、IPA の IC カードテストビークルを使用しました。

参考文献

[1] Sonia Belaïd, Luk Bettale, Emmanuelle Dottax, Laurie Genelle, and Franck Rondepierre: Differential power analysis of HMAC SHA-2 in the hamming weight model. In Pierangela Samarati, editor, *SECRYPT 2013 - Proceedings of the 10th International Conference on Security and Cryptography*, pages 230–241. SciTePress, 2013.

[2] Mihir Bellare, Ran Canetti, and Hugo Krawczyk: Keying hash functions for message authentication. In Neal Koblitz, editor, *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings*, volume 1109 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 1996.

[3] Alex Biryukov, Daniel Dinu, Yann Le Corre, and Aleksei Udovenko: Optimal first-order Boolean masking for embedded IoT devices. In Thomas Eisenbarth and Yannick Teglia, editors, *Smart Card Research and Advanced Applications - 16th International Conference, CARDIS 2017*, volume 10728 of *Lecture Notes in Computer Science*, pages 22–41. Springer, 2018.

[4] Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi: Towards sound approaches to counteract power-analysis attacks. In Michael J. Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 398–412. Springer, 1999.

[5] Jean-Sébastien Coron, Johann Großschädl, Mehdi Tibouchi, and Praveen Kumar Vadnala: Conversion from arithmetic to Boolean masking with logarithmic complexity. In Gregor Leander, editor, *Fast Software Encryption - 22nd International Workshop, FSE 2015*, volume 9054 of *Lecture Notes in Computer Science*, pages 130–149. Springer, 2015.

[6] Louis Goubin: A sound method for switching between Boolean and arithmetic masking. In Çetin Kaya Koç, David Naccache, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2001*, volume 2162 of *Lecture Notes in Computer Science*, pages 3–15. Springer, 2001.

[7] Toru Hashimoto and Boutheina Chetali: High level CC certification in Japan. In *The 2013 International Common Criteria Conference, ICCC 2013*, 2013.

[8] ISO/IEC: *ISO/IEC 17825. Information technology – Security techniques – Testing methods for the mitigation of non-invasive attack classes against cryptographic modules.* ISO/IEC, 2016.

[9] Paul C. Kocher: Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In Neal Koblitz, editor, *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer, 1996.

[10] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun: Differential power analysis. In Michael J. Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.

[11] Peter M. Kogge and Harold S. Stone: A parallel algorithm for the efficient solution of a general class of recurrence equations. *IEEE Trans. Computers*, 22(8):786–793, 1973.

[12] Kerstin Lemke, Kai Schramm, and Christof Paar: DPA on n-bit sized Boolean and arithmetic operations and its application to IDEA, RC6, and the HMAC-construction. In Marc Joye and Jean-Jacques Quisquater, editors, *Cryptographic Hardware and Embedded Systems - CHES 2004*, volume 3156 of *Lecture Notes in Computer Science*, pages 205–219. Springer, 2004.

[13] George Marsaglia: Xorshift RNGs. *Journal of Statistical Software*, 8:1–6, 2003.

[14] Robert P. McEvoy, Michael Tunstall, Colin C. Murphy, and William P. Marnane: Differential power analysis of HMAC based on SHA-2, and countermeasures. In Sehun Kim, Moti Yung, and Hyung-Woo Lee, editors, *Information Security Applications, 8th International Workshop, WISA 2007*, volume 4867, pages 317–332. Springer, 2007.

[15] National Institute of Advanced Industrial Science and Technology: Side-channel Attack Standard Evaluation Board (SASEBO), SASEBO-W. 2012.

[16] National Institute of Standards and Technology (NIST): Federal Information Processing Standards Publication (FIPS) 197, Advanced Encryption Standard (AES). 2001.

[17] National Institute of Standards and Technology (NIST): Federal Information Processing Standards Publication (FIPS) 198-1, The Keyed-Hash Message Authentication Code (HMAC). 2008.

[18] National Institute of Standards and Technology (NIST): Federal Information Processing Standards Publication (FIPS) 180-4, Secure Hash Standard (SHS). 2015.

[19] Jungk, B., Petri, R., Stöttinger, M.: Efficient Side-

Channel Protections of ARX Ciphers. CHES 2018, Cryptology ePrint Archive, Report 2018/693, 2018.

付　　録

A.1 Biryukov らの加算に基づく A2B 変換

Biryukov らのマスク付き算術加算アルゴリズム (Algorithm 3) に基づき、Algorithm 7 に示すように、A2B 変換法を構成することができる。この変換法は、我々の提案法と同様に、二つのランダムマスクを必要とする。

Algorithm 7 Arithmetic-to-Boolean Conversion based on Biryukov et al.'s Addition

Input: $A, r \in \{0,1\}^k$ and $n = \max(\lceil \log_2(k-1) \rceil, 1)$ such that $A = x - r \in \mathbb{F}_{2^k}$
Output: x' such that $x' \oplus r = A + r \mod 2^k$

$x_2, y_2 \leftarrow \{0,1\}^k$
$x_1 \leftarrow x \oplus x_2$
$y_1 \leftarrow y \oplus y_2$
$p_1, p_2 \leftarrow \mathsf{SecXor2}(x_1, x_2, y_1, y_2)$
$g_1, g_2 \leftarrow \mathsf{SecAnd2}(x_1, x_2, y_1, y_2)$
$g_1, g_2 \leftarrow ((g_1 \oplus x_2) \oplus g_2, x_2)$
for $i := 1$ to $n-1$ **do**
    $h_1, h_2 \leftarrow \mathsf{SecShift2}(g_1, g_2, 2^{i-1})$
    $u_1, u_2 \leftarrow \mathsf{SecAnd2}(p_1, p_2, h_1, h_2)$
    $g_1, g_2 \leftarrow \mathsf{SecXor2}(g_1, g_2, u_1, u_2)$
    $h_1, h_2 \leftarrow \mathsf{SecShift2}(p_1, p_, 2^{i-1})$
    $h_1, h_2 \leftarrow ((h_1 \oplus x_2) \oplus h_2, x_2)$
    $p_1, p_2 \leftarrow \mathsf{SecAnd2}(p_1, p_2, h_1, h_2)$
    $p_1, p_2 \leftarrow ((p_1 \oplus y_2) \oplus p_2, y_2)$
end for
$h_1, h_2 \leftarrow \mathsf{SecShift2}(g_1, g_, 2^{n-1})$
$u_1, u_2 \leftarrow \mathsf{SecAnd2}(p_1, p_2, h_1, h_2)$
$g_1, g_2 \leftarrow \mathsf{SecXor2}(g_1, g_2, u_1, u_2)$
$x' \leftarrow x \oplus 2g_1 \oplus 2g_2$
return x'