複合暗号演算を行うグループ署名回路に対する SPA 対策オーバヘッドの基礎検討

森岡 澄夫 †

† 日本電気株式会社 システム IP コア研究所
211-8666 神奈川県川崎市中原区下沼部 1753
s-morioka@ak.jp.nec.com

あらまし ユーザ ID を特定せず認証が行えるグループ署名は，多ビット長のべき乗剰余演算や楕円曲線演算が 30〜40 回組み合わさって構成される．そのような複合暗号演算を行う回路に対するサイドチャネル攻撃の可能性や対策法については，従来，検討がほとんど行われていない．従って現時点では，全演算に対し一律に既知対策法を施すのが最も簡便かつ安全だが，平均計算時間や平均消費エネルギーの増加が懸念される（それぞれ 10〜30 %，12 %程度）．今回，それらのオーバヘッドを推定した結果，対策を施す演算を最少限に押さえれば，標準的な SPA 対策については平均計算時間の増加がほぼ 0（高々7 %），平均消費エネルギーの増加が約 3 %で済むことが分かった．

Basic Investigation on Overhead of SPA Countermeasures to Group Signature Circuit

Sumio Morioka†

†System IP Core Research Laboratories, NEC Corporation
1753 Shimonumabe, Nakahara-ku, Kawasaki-shi 211-8666, JAPAN
s-morioka@ak.jp.nec.com

Abstract Group signature algorithm is a combination of more than 30 elliptic curve (ECC) and modular (RSA) operations. Little research on side channel attacks to the group signature H/W has been done yet, while a lot of reports on ECC and RSA have been published. In this paper, we investigated the amount of computation time and power overhead when minimum SPA countermeasure is applied to the group signature H/W, and found that nealy zero percent (7 percent, at most) increase is expected.

1 はじめに

1991 年に Chaum と Heyst[1] により提案されたグループ署名は，署名者があるグループに所属することは誰でも検証できるが，具体的に誰であるかの特定は管理者しかできない，という匿名性を特徴として持つ（図 1）．その応用として，電子マネー，個人認証，機器認証等の事例が検討されている．

グループ署名はべき乗剰余演算（RSA 暗号と同じ），楕円曲線演算やハッシュ演算を約 30〜40 回組み合わせた処理であり，計算コストが高いが，計算機の性能向上によって実用速度で処理できるようになってきた．文献 [2, 3] では，

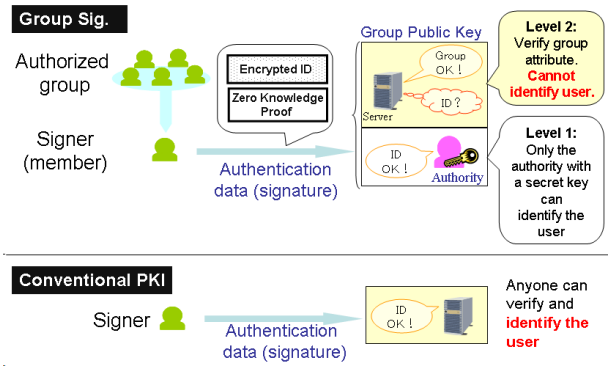

図 1: グループ署名と従来の電子署名の違い

3GHz 動作の高速 PC（シングル・コア）上のソフトウェアによって，署名生成・検証を 0.01〜0.1 秒程度で行った例が報告されている．また，著者らは，低クロック・少電源容量の組み込み機器や端末機器での利用を目的に，高位レベル回路設計法を活用しつつ回路実装を試み，低コスト 0.25um ASIC で上記 PC ソフトウェアと同等の速度（0.135 秒@100MHz）を達成した [5, 6]．その消費電力は 425mW@100MHz と PC 等と比べ 1/100 未満であり，より新しい半導体プロセスならば性能はさらに数倍，電力は 1/10 未満になると考えられる．

本稿では，グループ署名回路に対するサイドチャネル攻撃について検討を行う．近年，暗号処理系に対するサイドチャネル攻撃の脅威が広く認知されるようになり，RSA 暗号回路や楕円曲線暗号回路に対する攻撃法や対策法についても，多数の研究報告がなされている [7, 8, 9]．

いっぽう，グループ署名のような複合暗号演算を対象とした研究は著者らが知る限りほとんどなく，どのような攻撃が有り得るかも明確には判明していない．そこで文献 [5, 6] のグループ署名 ASIC 実装では暫定的に，内部で行う全ての RSA・楕円曲線演算に対し，標準的 SPA 対策 [7, 8, 9] を施すこととした[1]．それにより，グループ署名演算全体として，平均消費エネルギー（電力の時間による積分値）に約 12 %，平均計算時間に 10〜30 %の増加が生じた．

[1]べき乗剰余演算についてはバイナリ法に対する squaring-and-multiply-always 法を適用し，楕円曲線のスカラ乗算についても同様に double-and-add-always 法を適用した．なお，DPA 対策は実装上の都合で行わなかったが，対策の必要性がないと判断したわけではない．

しかしながら，全演算を一律に保護するのは過剰対策であると考えられ，グループ署名の演算コストが相当高いことからも，サイドチャネル攻撃対策のオーバヘッドは最低限に抑えたい．

そこで今回，対策がどうしても必要な演算を特定するとともに，対策のためにかかるオーバヘッドの見積もり評価を行った．その結果，標準的な SPA 対策については平均計算時間の増加がほぼ 0（高々7 %），平均エネルギーの増加が約 3 %で済むことが分かった．計算時間の増加分がとくに少ないのは，複合演算であるがゆえ，並列実行する他の演算でオーバヘッドを隠蔽できるからである．

以下，**2.** ではグループ署名の演算アルゴリズムと回路実装例について，**3.** では今回行ったサイドチャネル攻撃対策オーバヘッドの検討について述べる．

2 グループ署名の演算アルゴリズムと回路実装事例

本稿では文献 [2, 3] にある典型的なグループ署名アルゴリズムを用いる．

2.1 モデル

グループ署名スキームには，**ユーザ**，**発行者**，**追跡者**，**失効者**の四つのエンティティが参加する．ユーザは署名生成と検証を行う者であり，当該グループのメンバである．発行者はユーザをグループに追加する権限を，追跡者は署名者を特定する権限を，失効者はユーザをグループから除去する権限をそれぞれ持つ．署名スキームには鍵生成，ユーザ追加，ユーザ除去，署名生成，署名検証，署名者特定（追跡）の各手続きが含まれるが，本稿では，一般ユーザが利用する署名生成と検証の実装について考える．

2.2 セキュリティ・パラメータ

以下，セキュリティ・パラメータとして $\kappa = (\kappa_n, \kappa_\ell, \kappa_e, \kappa_{e'}, \kappa_q, \kappa_c, \kappa_S)$ を導入する．$\kappa_n, \kappa_\ell, \kappa_e, \kappa_{e'}$ のビット数はそれぞれ n, ℓ, e, e' であり，κ_q はある楕円曲線 $\mathcal{G}$ の位数のビット数，κ_c はあるハッシュ関数（Fiat-Shamir ヒューリスティ

クスを利用）のビット数である．κ_S はある乱数 r のビット数表記に用いる（r は a を任意の整数として $|a|+\kappa_S$ ビットであり，$a+r$ と r は統計的に識別不可能）．RSA-1024 と ECC-160 に相当する標準セキュリティレベルの場合，κ_n, κ_ℓ, κ_e, $\kappa_{e'}$, κ_q, κ_c, κ_S はそれぞれ 1024, 1024, 504, 60, 160, 160, 60 である[2]．

2.3 各エンテイティが持つ公開鍵と秘密鍵

$\mathcal{G}$ を位数 q（q はビット数 κ_q の素数）の有限巡回群とする．また，$QR(n)$, Λ, $[c]G$, $+_e$, $-_e$ をそれぞれ quadratic residue modulo n, $[0, 2^\lambda)$ の範囲中の整数の集合（$\lambda = \kappa_n + \kappa_q + \kappa_S$），楕円曲線 G 上のスカラ乗算，ポイント加算，ポイント減算とする．このもとで，各エンティティは以下の鍵ペアを持つ．*A.* **発行者**：公開鍵は $\mathsf{ipk} = (n, a_0, a_1, a_2)$, 秘密鍵は $\mathsf{isk} = (p_1, p_2)$. p_1 と p_2 は $\kappa_n/2$ ビットの safe prime number で，$n = p_1 p_2$ かつ $a_0, a_1, a_2 \in \mathrm{QR}(n)$．*B.* **追跡者**：公開鍵は $\mathsf{opk} = (q, G, H_1, H_2)$, 秘密鍵は $\mathsf{osk} = (y_1, y_2)$．$y_1, y_2 \in \mathbb{Z}_q$, $G \in \mathcal{G}$, $(H_1, H_2) = ([y_1]G, [y_2]G)$ である．*C.* **失効者**：公開鍵は $\mathsf{rpk} = (\ell, b, w)$, 秘密鍵は $\mathsf{rsk} = (\ell_1, \ell_2)$．$\ell_1$ と ℓ_2 は $\kappa_\ell/2$ ビットの safe prime number で，$\ell = \ell_1 \ell_2$ かつ $b, w \in \mathrm{QR}(\ell)$．*D.i* **番目のユーザ**：公開鍵は $\mathsf{mpk}_i = (h_i, A_i, e'_i, B_i)$, 秘密鍵は $\mathsf{msk}_i = x_i$. $x_i \in \Lambda$, $h_i = [x_i]G$, $B_i = b^{1/e'_i} \bmod \ell$, $e_i = 2^{\kappa_e} + e'_i$, $a_0 a_1^{x_i} \equiv A_i^{e_i} \bmod n$ である．

2.4 署名生成アルゴリズム

署名生成の入力は ipk, rpk, opk, mpk_i, msk_i とメッセージ m である．ここで $\mathsf{Hash} : \{0,1\}^* \to \{0,1\}^{\kappa_c}$ を衝突困難なハッシュ関数とし，$e_i = 2^{\kappa_e} + e'_i$ とする．具体的には，FIPS PUB 186-3 に記載の楕円曲線とハッシュ関数 SHA-224 を用いる．このもとで，以下のアルゴリズムで署名を生成する．

1. $\rho_E \in \mathbb{Z}_q$, $(\rho_m, \rho_r) \in \{0,1\}^{\kappa_n/2} \times \{0,1\}^{\kappa_\ell/2}$ $\mu_x \in \{0,1\}^{\lambda+\kappa_c+\kappa_S}$, $\mu_s \in \{0,1\}^{\kappa_e+(\kappa_n/2)+\kappa_c+\kappa_S}$, $\mu_{e'} \in \{0,1\}^{\kappa_{e'}+\kappa_c+\kappa_S}$, $\mu_t \in \{0,1\}^{\kappa_{e'}+(\kappa_\ell/2)+\kappa_c+\kappa_S}$, $\mu_E \in \mathbb{Z}_q$ をそれぞれランダムに選ぶ．

[2]RSA-2048 と ECC-224 に相当する高セキュリティレベルの場合，それぞれ 2048, 2048, 736, 60, 224, 224, 112 である．

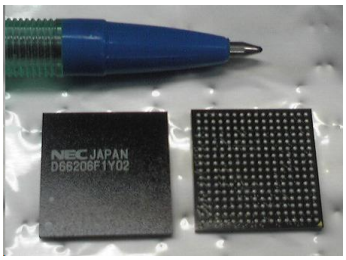

図 2: サンプル ASIC (0.25um ゲートアレイ)

最大動作周波数	100MHz (ワースト条件)
1024bit 署名生成速度	0.135 秒 (100MHz)
1024bit 署名検証速度	0.135 秒 (100MHz)
ゲートサイズ (ロジック部)	810K GE (DFT 後) 650K GE (DFT 前)
オンチップ SRAM 量	総計 320K ビット
平均消費電力	425mW (100MHz, 1.5V)

表 1: サンプル ASIC の実装評価結果 (0.25um ゲートアレイ)

2. $\boldsymbol{E} = (E_0, E_1, E_2) = ([\rho_E]G, h_i +_e [\rho_E]H_1, h_i +_e [\rho_E]H_2)$, $V_{\mathtt{ComCipher}} = ([\mu_E]G, [\mu_x]G +_e [\mu_E]H_1, [\mu_x]G +_e [\mu_E]H_2)$ をそれぞれ計算する．

3. $(A_{\mathrm{COM}}, B_{\mathrm{COM}}) = (A_i a_2^{\rho_m} \bmod n, B_i w^{\rho_r} \bmod \ell)$, $(V_{\mathtt{ComMPK}}, V_{\mathtt{ComRev}}) = (a_1^{\mu_x} a_2^{\mu_s} A_{\mathrm{COM}}^{-\mu_{e'}} \bmod n, w^{\mu_t} B_{\mathrm{COM}}^{-\mu_{e'}} \bmod \ell)$ をそれぞれ計算する．

4. $c = \mathsf{Hash}(\kappa, \mathsf{ipk}, \mathsf{opk}, \mathsf{rpk}, \boldsymbol{E}, A_{\mathrm{COM}}, B_{\mathrm{COM}}, V_{\mathtt{ComCipher}}, V_{\mathtt{ComMPK}}, V_{\mathtt{ComRev}}, m)$ を計算する．

5. $\tau_x = cx_i + \mu_x$, $\tau_s = ce_i\rho_m + \mu_s$, $\tau_t = ce'_i\rho_r + \mu_t$, $\tau_{e'} = ce'_i + \mu_{e'}$, $\tau_E = c\rho_E + \mu_E \bmod q$ をそれぞれ計算する．

6. 以上より，$(\boldsymbol{E}, A_{\mathrm{COM}}, B_{\mathrm{COM}}, c, \tau_x, \tau_s, \tau_{e'}, \tau_t, \tau_E)$ を署名として出力する．

2.5 署名検証アルゴリズム

署名検証の入力は ipk, opk, rpk, メッセージ m, m に添付された署名 $\sigma = (\boldsymbol{E}, A_{\mathrm{COM}}, B_{\mathrm{COM}}, c, \tau_x, \tau_s, \tau_{e'}, \tau_t, \tau_E)$ である．このもとで，以下のアルゴリズムで署名を検証する．

コア種類	DFT 前ロジックサイズ（GE 換算）
楕円曲線演算	140K
モジュロ演算	200K
多ビット長整数演算	80K
HASH ＋擬似乱数生成	120K
その他（全体制御等）	110K

表 2: 各演算コアの回路規模 (0.25um ゲートアレイ)

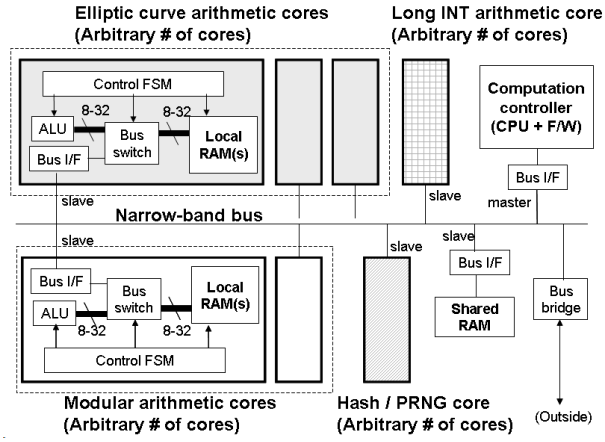

図 3: グループ署名回路のアーキテクチャ

1. $|\tau_x| \leq \lambda + \kappa_c + \kappa_S$ と $|\tau_{e'}| \leq \kappa_{e'} + \kappa_c + \kappa_S$ が両方成立するかを調べる．もし成立するなら次のステップへ進み，しなければ reject を結果として返す．

2. $V'_{\mathtt{ComCipher}} = ([\tau_E]G -_e [c]E_0,\ [\tau_x]G +_e [\tau_E]H_1 -_e [c]E_1,\ [\tau_x]G +_e [\tau_E]H_2 -_e [c]E_2)$ を計算する．

3. $V'_{\mathtt{ComMPK}} = a_0^c a_1^{\tau_x} a_2^{\tau_s} A_{\mathrm{COM}}^{-(c2^{\kappa_e}+\tau_{e'})} \bmod n$ と $V'_{\mathtt{ComRev}} = b^c w^{\tau_t} B_{\mathrm{COM}}^{-\tau_{e'}} \bmod \ell$ を計算する．

4. $c = \mathsf{Hash}(\kappa,\ \mathsf{ipk},\ \mathsf{opk},\ \mathsf{rpk},\ \boldsymbol{E},\ A_{\mathrm{COM}},\ B_{\mathrm{COM}},\ V'_{\mathtt{ComCipher}},\ V'_{\mathtt{ComMPK}},\ V'_{\mathtt{ComRev}},\ m)$ が成立するかを調べる．もし成立するなら accept を，しなければ reject を結果として返す．

2.6 グループ署名回路の ASIC 実装事例

著者らは文献 [5, 6] において，上記の署名生成・検証アルゴリズムの回路実装を行った（図 2）．サンプル ASIC の性能等を表 1 に，内部構成を図 3 に，各演算コアのサイズを表 2 にそれぞれ示す[3]．たとえば署名生成では表 3 に挙げた演算を行うが，それぞれモジュロ演算コア（RSA と同じ），楕円曲線演算コア，多ビット長整数演算コア，および擬似乱数生成/ハッシュ演算コア[4]に割り付けて処理される．

[3]モジュロ演算コアや楕円曲線演算コアは，一般的な RSA 暗号回路や楕円曲線暗号回路よりも大きいが，内部演算の並列度を上げたりセルのストレングスを上げたりして高速化したためである．また，モジュロ演算と楕円曲線演算の並列実行を可能とするため，両者の回路共有は行っていない．

[4]擬似乱数生成でハッシュ関数を使用するので，同一コアにまとめてある．

演算種類	演算ビット長	1回あたりクロック数(100MHz)	実行回数	総時間比率
楕円・スカラ乗算	160	970,055	7	31.4%
楕円・ポイント加算	160	4,861	4	< 0.1%
モジュロ・乗剰余	1024 x 1024	1,841	5	< 0.1%
モジュロ・べき乗剰余	1024 ^ 280	521,439	2	4.8%
	1024 ^ 512	943,291	2	8.7%
	1024 ^ 792	1,500,187	1	6.9%
	1024 ^ 1236	2,341,599	1	10.8%
	1024 ^ 1464	2,804,294	1	13.0%
モジュロ・逆元	1024	1,713,022	2	15.8%
整数・乗算	160 x 60	84	1	< 0.1%
	160 x 160	125	1	< 0.1%
	160 x 576	301	1	< 0.1%
	160 x 1016	489	1	< 0.1%
	160 x 1244	584	1	< 0.1%
	512 x 50	239	1	< 0.1%
	512 x 504	736	1	< 0.1%
整数・剰余	321 % 160	4,611	1	< 0.1%
	1464 % 160	34,146	1	0.16%
ハッシュ (SHA-224)	-----	< 10,000	1	< 0.1%
擬似乱数生成	160	29,720	2	0.2%
	280	35,608	1	0.16%
	512	35,688	2	0.33%
	792	47,475	1	0.22%
	1236	53,342	1	0.25%
	1464	69,070	1	0.32%
データ転送	------	1,286,806	------	5.9%
合計(逐次実行時)	------	**21,633,992**	------	**100%**

表 3: 用いるプリミティブ演算の性能例（0.25um ゲートアレイ）

注意すべき点として，表 1 中の時間比率は，演算を逐次実行したと仮定した場合の値，すなわち延べ演算時間中の比率であるが，実際には演算実行を並列化し高速化する（次章でも触れる）．サンプル ASIC ではコアの個数は各 1 個であるが，複数のコアを用意して高速化を図ることもできる．コア数と演算性能の関係については文献 [5, 6] 等を参照されたい．

3 SPA 対策箇所と対策オーバヘッドの検討

3.1 全演算に対策を施した場合のオーバヘッド

前章で述べたサンプル ASIC では，モジュロ演算コアでのべき乗剰余演算の実装に，標準的な計算アルゴリズム（バイナリ法とモンゴメリ

乗算）と SPA 対策法（squaring-and-multiply-always 法）[7, 9] を利用した．楕円曲線演算コアについても同様で，スカラ乗算に add-and-double-always 法を適用した [8]．また，グループ署名演算に対してどのようなサイドチャネル攻撃が可能であるか不明であったため，個々の演算ごと対策の必要性を検討することはせず，全演算一律に SPA 対策を施した．表 3 における各演算のクロック数も，SPA 対策後の値である．

その結果，平均消費エネルギー（平均電力の時間による積分値）と平均計算時間の両方にオーバヘッドが生じた．

まず，平均消費エネルギーの増加分について推定する．SPA 対策後のグループ署名生成においては，表 3 に示したとおり，延べ演算時間の 44.2 %を上記べき乗剰余算が，31.4 %を上記スカラ乗算が占めている．ここで，コアのダイナミック電力が表 2 の回路規模に比例すると仮定すると（コア内部論理のトグル率は高いので，不自然な仮定ではない），ダイナミック電力による総エネルギーのうち 46.5 %をべき乗剰余算が，23.0 %をスカラ乗算が占めることとなる．また，標準的 RSA 暗号回路や楕円曲線暗号回路においては，SPA 対策を施すことで平均 33 %の消費エネルギー増になる[5]．よって，べき乗剰余算やスカラ乗算に SPA 対策を一切施さなかった場合，ダイナミック電力による総エネルギーは 17.375 %減少すると考えられる．

以上より，総エネルギーのうちダイナミック電力に由来する成分の比率を 60 %と仮定した時（近年の半導体プロセスでは標準的な仮定である），全ての上記演算に SPA 対策を施すことで総エネルギーが 12.6 %増加する，と推定できる．

次に，平均計算時間の増加分について記す．計算時間は，文献 [5, 6] の高位アーキテクチャ設計ツールに対し，表 3 中のべき乗剰余算およびスカラ乗算の時間を 25 %削減した値を与え，演算

[5]ダイナミック電力によるエネルギーに着目する．バイナリ法によるべき乗剰余算において，乗算と自乗算に同エネルギー E が必要で，鍵における 0/1 の発生確率が同じであると仮定する．すると，鍵 1 ビットあたりの平均使用エネルギーは 1.5E である．これに対し，SPA 対策後のエネルギーは 2E で，33 %増加する．計算時間や楕円曲線のスカラ乗算についても同様.

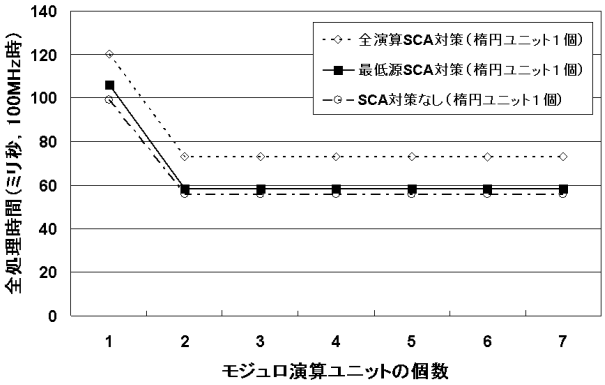

図 4: SPA 対策の有無の違いによる性能変化（楕円コア数＝ 1）

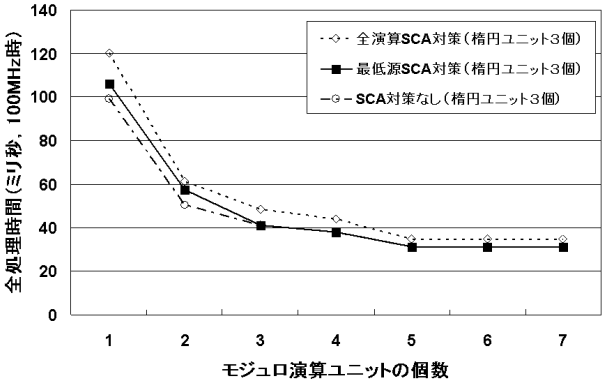

図 5: SPA 対策の有無の違いによる性能変化（楕円コア数＝ 3 以上）

スケジューリング（並列化）をする事によって求めた．図 4 と図 5 がその評価であるが，3.3 で説明する．結論のみ先に述べると，11〜30 %の増加となる．

3.2 SPA 対策が必要な演算の特定

署名検証については，秘密鍵を利用しないため対策不要と考えられる．

署名生成については，ユーザ秘密鍵 x_i が漏洩しないよう保護しなければならない．また，2.4 の計算ステップ 5 において $\tau_x = cx_i + \mu_x$ を計算するが，τ_x と c は署名として直接出力されるため，μ_x（ステップ 1 で生成する乱数値）も漏洩しないよう保護する必要がある．

よって，最低限以下の演算について，サイドチャネル攻撃対策が必要と考えられる：ステップ 1 の μ_x 生成，ステップ 2 の $[\mu_x]G$，ステップ 3 の $a_1^{\mu_x} \bmod n$, ステップ 5 の $cx_i + \mu_x$.

これらは表 3 の灰色部分に相当する．なお，擬似乱数生成や整数演算についても何らかの攻撃対策が必要ということになるが，それは今後の課題である．本稿ではべき乗剰余算とスカラ乗算について考える．

3.3 最低限の対策を施した場合のオーバヘッド

3.2 で述べた演算には SPA 対策を行い，他の演算については SPA 対策を外した場合のオーバヘッドについて評価した．

平均消費エネルギーについては，3.1 と同様の計算で，ダイナミック電力に由来する成分の比率を 60 %と仮定した時，3.1 %増加すると推定できる．

平均計算時間の増加分については，コアの使用数によって変動が生じる．演算並列度が変わるからである．署名生成演算の速度は，モジュロ演算コアが 1〜7 個，楕円曲線演算コアが 1〜3 個の範囲にあるとき変化するが（それ以上増やしても変化しない），そのもとで SPA 対策の有無による全体速度の違いをプロットしたのが図 4 と図 5 である．コア数が少なく演算並列度が低いときは約 7 %の増加であるが，コア数が増えて並列度が高くなると計算時間の増加は 0 となる．

4 おわりに

本稿では，グループ署名演算に標準的な SPA 対策を施した場合の電力・計算時間オーバヘッドを評価した．その結果，オーバヘッドはほとんどないとの見通しを得たが，暗号演算間の相関等を利用した新しい攻撃の可能性は，依然未検討課題として残っている．

参考文献

[1] D.Chaum and E.van Heyst, ”Group signatures,” EUROCRYPT ’91, LNCS Vol.547, pp.257-265, 1991.

[2] J.Camenisch and J.Groth, ”Group signatures: Better efficiency and new theoretical aspects,” SCN 2004, LNCS Vol. 3352, pp.120-133, 2004.

[3] T.Isshiki, K.Mori, K.Sako, I.Teranishi, and S.Yonezawa, ”Using Group Signature for Identity Management and its Implementation,” DIM2006, 2006.

[4] 一色俊幸, 森健吾, 尾花賢, 佐古和恵，”グループ署名のスマートフォンへの実装,” 2008 年 暗号と情報セキュリティシンポジウム (SCIS2008) 3C3-3, 2008.

[5] 森岡澄夫，荒木俊則，一色寿幸，尾花賢，佐古和恵，寺西勇，“ ESL 設計法を活用したグループ署名アルゴリズムの ASIC 化，” 2010 年 暗号と情報セキュリティシンポジウム（SCIS2010）3C4-5, 2010.

[6] S.Morioka, T.Isshiki, S.Obana and K.Sako, ”Flexible Architecture Optimization and ASIC Implementation of Group Signature Algorithm using a Custimized HLS Methodology,” IEEE Int. Symp. on Hardware-Oriented Security and Trust (HOST 2011), pp.57-62, 2011.

[7] N.Homma, A.Miyamoto, T.Aoki, A.Satoh, A.Samir, ”Comparative Power Analysis of Modular Exponentiation Algorithms,” IEEE Trans. on Computers, Vol.59, No.6, pp.795-807, 2010.

[8] F.Junfeng, G.Xu, E.De Mulder, P.Schaumont, B.Preneel, I.Verbauwhede, ”State-of-the-art of secure ECC implementations: a survey on known side-channel attacks and countermeasures,” IEEE Int. Symp. on Hardware-Oriented Security and Trust (HOST 2011), pp.76-87, 2011.

[9] M.Izumi, K.Sakiyama, K.Ohta, ”A New Approach for Implementing the MPL Method toward Higher SPA Resistance,” Intl. Conf. on Availability, Reliability and Security, 2009 (ARES ’09), pp.181-186, 2009.