秘密カウンタプロトコルを用いた電子投票システムの携帯情報端末への実装と評価

中里純二[†] 菊池浩明[††] 中西祥八郎[†††]

集計者に投票内容が見えないことが特徴的な秘密カウンタプロトコルを現在，普及している携帯情報端末の上で実装した．携帯情報端末の限られた容量や計算力を克服するために，我々は，任意精度の剰余算術演算に特化した容量の小さい Java クラスの `ModInt` を設計した．本論文では，提案システムの評価や，`BigInteger` クラスとの比較を行う．

An Evaluation and Implementation of Electronic Voting System Using Oblivious Counter Protocol on Personal Digital Assistance

Junji Nakazato,[†] Hiroaki Kikuchi[††] and Shohachiro Nakanishi[†††]

We implemented the oblivious counter protocol on a platform of personal digital assistance (PDA), which is able to keep a tally from encrypted ballots without decrypting them. In order to overcome the restricted storage and computational power of PDA, we have developed a Java class, `ModInt`, which is designed for light-weight arbitrary-precision integer computation and especially for modular arithmetic operations. In this paper, we evaluate the processing time of the system and show a comparison with the built-in class BigInteger.

1. はじめに

2001 年 11 月制定された電子投票法案により，地方選挙での電子投票の利用を認められた．それにともない，2002 年 6 月には全国ではじめて岡山県新見市において電子投票が行われた．この選挙では，実際に投票所に行き，投票端末を利用して投票を行う形式で行われた．つまり，まったくのオフラインで行われた．これは，電子投票法案の中で，オンラインによる集計が認められていないためである．しかし，投票後には，国政選挙への利用やオンラインによる集計などを視野に入れるといった内容の発表がされた[1)]．

現在，インターネットなどのオンラインを利用して行うことを前提とした研究は多くされている．インターネットを用いた電子選挙においては，正規の投票資格を持つ投票者（有権者）であることを証明すること（認証）と，だれが何に投票したかを分からなくすること（匿名性）をいかに両立させるかが大きな課題であった．

この課題を解決する技術の 1 つに，ブラインド署名と匿名通信路を用いたシステムが藤岡らによって開発されている[2)]．しかし，匿名通信路は，多重に公開鍵暗号を繰り返す必要があり，その大きなコストを削減するために，Abe によるハイブリット MIX[3)] や，Furukawa らによるシャッフル[4)] などの多くの提案がなされている研究途上の技術である．一方，我々は，文献 5), 6) で，匿名通信路を用いない Linear Feedback Shift Register（LFSR）を利用した Oblivious LFSR Counter プロトコルを提案している．本プロトコルでは，ElGamal 暗号を採用し，秘密鍵を閾値法で分散することがより容易である．n bit の線形フィードバックシフトレジスタ（LFSR）を用いることで，投票者数 m に対して，通信量および，計算量の効率化を図っている．

本研究では，現在普及している携帯情報端末に文

† 東海大学大学院工学研究科電気工学専攻
Course of Electrical Engineering, Graduate School of Engineering, Tokai University
†† 東海大学電子情報学部情報メディア学科
Department of Information Media Technology, School of Information Technology and Electronics, Tokai University
††† 東海大学電子情報学部情報科学科
Department of Human and Information Science, School of Information Technology and Electronics, Tokai University

献 5) を実装することを試みる．なぜ PDA を選んだかの理由は次のとおりである．

1. 若年層の投票促進効果
   内閣府の報告[7] によると，2002 年，20 代の 89.4% がパソコンや，携帯情報端末から Internet を利用していることが示されている．したがって，携帯情報端末での投票ができると，従来は不参加であった，若年層を大きく増やす可能性があるためである．
2. 電子政府からの要請
   電子政府を実現するためには，国民 1 人 1 人が一意な ID を持ち，電子署名を実行できる情報端末を持つ必要がある．これにともない，PDA で電子投票を実行する環境が整う．
3. 身分証明の手段としての適合性
   1 人 1 台を常時身につけているため，パスポートなどよりも本人認証を確に行うことが可能である．さらに，通信手段を兼ね備えており，近年の多機能化によって，計算手段を持っている．
4. 双方向参加への応用
   持ち運びが容易なため，選挙だけではなく，イベント会場などでのアンケートなどに応用可能である．

以上の理由により，投票端末の候補として用いることがふさわしいと考える．

実装言語として近年の携帯情報端末の多くが対応している Java を用いた．本実装では，Java 言語を利用可能な携帯情報端末として最も普及していると思われる，NTT Docomo の i-appli を用いた．

Java で暗号計算を実現するには，`BigInteger` クラスが JDK に標準で用意されている．ところが，携帯情報端末を代表とする Java VM を搭載したほとんどのプラットフォームでは，このクラスが組み込まれていない．しかも，ソースコードを基にこのクラスを追加しようにも，バイトコード用の記憶容量には制限があり，約 40 k byte も費す `BigInteger` クラスを実現できない☆．

そこで，`BigInteger` よりもコンパクトで，より暗号計算用に特化した `ModInt` クラスをスクラッチから開発することとした．開発目標は次のとおりとした．

1 必要最小限のコンパクトなクラス
   他のコード用にメモリを残すため．
2 剰余計算への特化
   不要な任意長への対応をなくし，四則演算は初めから mod をとる．

このように，汎用的な携帯情報端末を用いて，暗号プロトコルを実現するには，いくつかの課題がある．特に，多倍長演算によって，どのくらいのオーバヘッドが生じるかは不明であり，実用的な電子投票システムを実現するためには，端末数や暗号強度などの適用可能な条件を明らかにする必要がある．

したがって，本論文の主旨は，現在普及している携帯情報端末上に，電子投票システムを実装し，そのフィジビリティを評価することである．

本論文では `ModInt` クラスの原理と仕様を示し，これを用いて実装した電子投票システムのパフォーマンスを報告する．本論文の構成は次のとおりである．

2 章では LFSR および，秘密カウンタ[8] を用いた文献 5) について説明する．3 章では実装システムについて説明し，`ModInt` クラスの詳しい説明を 3.3 節に記述する．4 章で本システムの評価を行い，文献 2) のシステムとの比較を示す．

2. Oblivious LFSR [5]

本論文では，文献 5) で提案した秘密線形フィードバックシフトレジスタプロトコルの概要を示す．なお，LFSR は，疑似乱数生成方式として知られている．LFSR を準同型性暗号を用いて実現することによって，シフトするかしないかを秘密にしたまま内部レジスタを更新することを可能とする．ここで，ElGamal 暗号の暗号化を $E[\cdot]$，復号を $D[\cdot]$ とする．

なお，LFSR の内部状態の変化において，票の集計を行うため，以後，これをカウンタと参照する．

2.1 LFSR

n 段の線形フィードバックシフトレジスタは，1 bit を記憶する n 個のフリップフロップ $A_1, \ldots, A_n$ で構成されている．レジスタは単位時間あたり左に 1 bit シフトし，特性多項式

$$f(x) = 1 + a_1 x + a_2 x^2 + \cdots + a_n x^n$$

について，内部レジスタを $i = 1, \ldots, n$ について，

$$A'_i = A_{i-1} \oplus a_{i-1} A_n \qquad (1)$$

により更新する（これをシフトアップと呼ぶ）．ただし，$A_0 = 0$，$A'_1 = A_n$，$a_j \in \{0, 1\}$ とする．$f(x)$ が原始（既約）多項式のとき，内部レジスタの状態は $2^n - 1$ の周期を持つ．

2.2 Oblivious LFSR

Step 1：（カウンタの初期化） 集計者 $\mathcal{C}$ は，カウンタの内部レジスタを $A_1 = E[-1]$，$A_2 =$

☆ なお，バイトコードの容量制限は 504i シリーズで 30 k byte，503i シリーズで 10 k byte である．また，503i シリーズの i-appli では，すべてのオブジェクトが継通している，clone() メソッドも動作しない．

$E[1], \ldots, A_n = E[1]$ と初期化する．ここで，m を投票者数とすると，$n = \lceil \log m \rceil$ とする．

Step 2：(投票) 投票者 $\mathcal{V}$ は，賛成・反対を表す投票内容 $b \in \{-1, 1\}$ によって，内部レジスタの更新値 C_i $(i = 1, \ldots, n)$ を

$$C_i = \begin{cases} E_{r_i}[A'_i] & (b = -1) \\ E_{r_i}[A_i] & (b = 1) \end{cases}$$

と定めて，カウンタへ送信する．ここで，賛成のとき，

$$C_i = E_{r_i}[A'_i] = E_{r_i}[1] \times A_{i-1} \times A_n^{a_i - 1} \quad (2)$$

となることに注意せよ．また，LFSR のシフトを行ったか行っていないかを集計者に秘密にするために再暗号化を行う．再暗号化とは，シフトを行った値または，行っていない値に $E[1]$ をかける処理とする．

Step 3：(正当性の証明) $\mathcal{V}$ は，投票内容 C_i すべてに対して，賛成・反対のどちらかを正しく投票しているということを式 (3) の知識の証明（Proof of Knowledge）を用いて証明する．

$$\begin{aligned} &PK\{\{(r_1, \ldots, r_n) : \\ &\quad C_1 = E_{r_1}[A_1] \wedge \cdots \wedge C_n = E_{r_n}[A_n]\} \\ &\vee\{(r_1, \ldots, r_n) : \\ &\quad C_1 = E_{r_1}[A'_1] \wedge \cdots \wedge E_{r_n}[A'_n]\}\} \quad (3) \end{aligned}$$

Step 4：(正当性の検証) $\mathcal{C}$ は $\mathcal{V}$ より受け取った $\{C_1, \ldots, C_n\}$ と PK，およびカウンタの内部状態 $\{A_1, \ldots, A_n\}$ を用いて，カウンタを LFSR に従ってシフトアップした結果 $\{A'_1, \ldots, A'_n\}$ と，$\{C_1, \ldots, C_n\}$ が矛盾していないか検証する．検証が正しければ投票内容 $\{C_1, \ldots, C_n\}$ を正しい票とし，新しいカウンタの値として保存する（$A_1 = C_1, \ldots, A_n = C_n$ とする）．間違っていれば票を破棄する．

Step 5：(開票) 開票者 $\mathcal{S}$ は，投票締切り後，カウンタの内部状態（$A_1, \ldots, A_n$）を受け取り，$B_1 = D[A_1], B_2 = D[A_2], \ldots, B_n = D[A_n]$ と復号する．ここで，各ビットを復号した平文を B_i とする．このままでは，結果は符号化されている．そこで，復号結果が LFSR の初期値から何回目の状態と同じになるかをカウントする．その回数が賛成者数となる．

3. 実装方法

本論文では 2.2 節のプロトコルを NTT Docomo の i-mode 対応携帯電話端末向けに行った実装について説明する．i-mode 対応携帯電話端末では，Java 言語を用いたアプリケーション i-appli[9] が動作可能である．

表 1 システム仕様
Table 1 Specification for the system.

開発言語	JDK1.3，J2ME Wireless SDK for the DoJa release 2.2
暗号化アルゴリズム	ElGamal 暗号
鍵サイズ	可変長（128〜512）
入出力プロトコル	HTTP/CGI

3.1 開発環境

i-appli 用開発ツールを文献 10) より入手して用いた．開発を行ったシステム仕様を**表 1** に示す．

3.2 構成

ここでは，i-appli の構成について説明する．

本実装では 7 つのクラスにより構成した．以下に実装クラスの説明をする．

- `ModInt` クラス
  暗号計算を提供．1,024 bit などの大きな整数を扱うことができる．3.3 節に詳細を述べる．
- `ElGamal_i` クラス
  ElGamal 暗号の暗号化を行う．必要最小限（暗号化のみ）のメソッドしか持たないような，i-appli 用に特化してある．特に，多倍長演算に `ModInt` クラスを用いている．
- `LFSR_i` クラス
  LFSR の更新処理を行う．賛成を投票するときはシフトアップの処理をしながら再暗号化を，反対を投票するときは再暗号化のみを行う．
- `Read2` クラス
  HTTP Request を利用してテキストデータの受信を行う．これにより，公開鍵やカウンタの内部状態を受信する．本実装では，16 進数表現されたテキストデータを読み込む．改行コード CR LF を区切り文字として用いる．
- `Write` クラス
  `HttpConnection` の POST を利用して投票内容の送信を行う．すべての値を文字列表現に変化し，区切り文字に改行コード CR LF を用いる．
- `Vote` クラス
  投票処理全般を行う．
- `i_vote` クラス
  i-appli 本体．ユーザインタフェースなどを提供する．

公開鍵や，秘密鍵はサーバで管理を行う．公開鍵は i-appli 起動時に毎回サーバにアクセスし取得する．これにより，鍵の変更が容易になりアプリケーションの更新を必要とせずに他の投票にも利用可能である．通

```
ModInt                                          BigInteger
p = new ModInt("71a04e8b02 … ");    p = new BigInteger("71a04e8b02 … ", 16);
q = new ModInt("e0ae02e788 … ");    q = new BigInteger("e0ae02e788 … ", 16);
m = p.getInstance("dedf322 … ");    m = new BigInteger("dedf322 … ", 16);
y = p.getInstance("12af7cc … ");    y = new BigInteger("12af7cc … ", 16);
r = q.getInstance("98af3df … ");    r = new BigInteger("98af3df … ", 16);
c = m.multiply(y.power(r));         c = m.multiply(y.modPow(r, p)).mod(p);
```

図 1 ModInt と BigInteger による ElGamal 暗号化のプログラム例
Fig. 1 The example of ElGamal encryption by ModInt and BigInteger.

表 2 BigInteger と ModInt の違い
Table 2 Difference between BigInteger and ModInt.

	BigInteger	ModInt
クラスサイズ	40 k byte	約 6 k byte
public メソッド数	43	15
整数型	符号付・可変長	符号なし・固定長（p 指定）
コンストラクタ入力	10 進，文字列，byte 配列	16 進，int 配列
オブジェクト内部表現	BigEndian，int 配列 × 1 符号，ビット長，ビット数他	BigEndian，int 配列 × 2

信プロトコルとしては HTTP プロトコルを採用し，実装の簡易化を行った．

本実装では，2.2 節のプロトコル Step3（正当性の証明）の処理を省略している．なぜならば，本論文の主題である i-appli では，セキュリティの観点から，プログラムをダウンロードしたサーバ以外へのデータの送信が不能なように制約されている．したがって，クライアントが，不正な投票値を送信する脅威は起こりにくいためである．加えて，我々が，文献 5) で報告しているように，知識の証明の処理は，証明時に暗号化の約 2.05 倍の時間と，検証時に約 2.03 倍の時間がかかることが分かっている．

3.3 ModInt クラス

ModInt クラスは，任意長の剰余演算を実行するオブジェクトとメソッドを提供する．必要最低限のコンパクトなクラスにするという目標を達成するため，BigInteger で用意されていたビット演算や符号のメソッドを取り除き，さらに，暗号計算に必要な部分にも次の制約を加えた．

- すべての入出力は，16 進表現のみ．
- 素数生成などは，サーバでオフラインで実行すればよいので，これも省略．
- 復号処理は，サーバでオフラインで実行するので，逆元を求める必要がないため，これも省略．

以上の工夫により，基本機能のみの 15 メソッドを約 6 k byte におさえたクラスを実現した．暗号計算ではすべての演算を法 p の下で行う．そこで，すべての整数オブジェクトには，対応するモジュロ mod の値を指定し，算術演算は自動的に剰余をとることとした．BigInteger との違いを明確にするため，ElGamal 暗号の一部

$$c = my^r \bmod p$$

を計算する例を考えよう．これを BigInteger で実行するためには，まず平文 m，公開鍵 y，法 p，乱数 r を各々コンストラクタから作っておき，

```
c=m.multiply(y.modPow(r, p)).mod(p)
```

と書く．p を明示的に 2 回指定しなくてはならず，記述性が悪い．しかも，m との積を先に実行するので，いったん $2p$ のサイズの値に拡張された後に，mod がとられるので，オーバヘッドが生じる．

これに対して，ModInt を用いた場合には，**図 1** のようになる．

オブジェクト m や y は，法 p の下での計算を行う整数として定義されているので，以後の計算では mod p の処理が省略されており，見通しがよい．

BigInteger と ModInt との違いを**表 2** に整理する．主なコンストラクタ，メソッドの説明を**表 3**，**表 4** に示す．

4. 評　　価

ここでは，実装システムのパフォーマンスの評価を行う．測定は，開発用エミュレータと，実際の携帯電話端末（NTT Docomo D503is）の両方の環境で行った．

4.1 エミュレータでの動作

表 5 にエミュレータの動作環境を示す．

実装システムを J2ME 付属の携帯電話端末エミュレータによってパフォーマンスの評価を行った．**表 6**，**表 7** にエミュレータ上での測定結果を示す．ここで，

表 3 主なコンストラクタ
Table 3 List of constructors.

コンストラクタ名	説明
`ModInt(String p)`	16 進表記された値 p を法に設定する
`ModInt(int[] a, int[] p)`	$a \bmod p$ のオブジェクトを作成する

表 4 主なインスタンスメソッド
Table 4 List of instance methods.

返り値	メソッド名	説明
`ModInt`	`getInstance(String s)`	16 進表記された値 s のインスタンスを作成する．p は，元の値をとる
`String`	`toString()`	インスタンスの値を 16 進表記にした文字列を返す
`ModInt`	`subtract(ModInt b)`	$this - b \pmod p$ の値を持つオブジェクトを返す
`ModInt`	`add(ModInt b)`	$this + b \pmod p$ の値を持つオブジェクトを返す
`ModInt`	`multiply(ModInt b)`	$this \times b \pmod p$ の値を持つオブジェクトを返す
`ModInt`	`shiftLeft()`	$this \ll 32 \pmod p$ した値を持つオブジェクトを返す
`ModInt`	`power(ModInt e)`	$this^e \pmod p$ の値を持つオブジェクトを返す
`long`	`compareTo(ModInt b())`	$this$ と b の大小比較を行う．$this = b$ ならば 0 を返す

表 5 エミュレータの実行環境
Table 5 Execution environment of emulator.

CPU	Pentium III 1.13 G
OS	Windows 2000 server
memory	512 MB
HDD	40 GB

表 6 鍵のビット $|p|$ の違いによる処理時間 [s]（$n = 3$，エミュレータ）
Table 6 Processing time for key size $|p|$ [s] ($n = 3$, Emulator).

$\|p\|$ [bit]	Key	Carry	Enc	Shift	Write	Total
128	0.6	0.3	0.7	0.00	0.03	6.4
256	0.5	0.4	2.7	0.01	0.05	8.6
512	0.7	0.6	18.9	0.10	0.10	27.4
1,024	0.8	1.1	40.0	0.17	0.16	47.1

Key を公開鍵ファイルの読み込み時間，Carry をレジスタの内部状態の読み込み時間，Enc をレジスタの内部状態の再暗号化時間，Shift を LFSR の Shift 処理時間，Write を投票内容の送信時間とする．Total は i-appli 起動から選択時間などの時間を除いた投票時間である．また，暗号に用いる法 p のビット数を $|p|$，LFSR のレジスタ数を n で表す．

表 6 は $n = 3$ にし，$|p|$ を変化させたときの測定結果である．表 7 は $|p|$ を 512 bit 一定にし，n を変化させたときの測定結果である．結果より，ファイルの入出力には $|p|$ のサイズや，n のサイズによって全体の処理時間に対する比率は，最大でも 15%程度で，最少だと 4%程度とほとんど影響していないことが分かる．しかし，暗号化時間は最少で全体の 10%程度だが，最大で 80%以上の処理時間を必要としている．また，$|p|$ を変化させる場合よりも，n を変化させた方が全体の処理時間に占める割合が多くなることが分かる．

表 7 レジスタのビット n の違いによる処理時間 [s]（$|p| = 512$，エミュレータ）
Table 7 Processing time for bit length of register n [s] ($|p| = 512$, Emulator).

n	Key	Carry	Enc	Shift	Write	Total
3	0.5	0.4	2.7	0.10	0.05	8.6
5	0.6	0.8	23.2	0.04	0.14	30.9
8	0.6	1.2	40.1	0.10	0.22	48.9

4.2 藤岡らのシステム[2] との比較

ここで，藤岡らのシステム[2] との比較を行う．

藤岡らは，ブラインド署名と匿名通信路を用いた電子投票システムを開発した．本システムは，選挙管理者の署名に Schnorr 署名を，開票者の暗号化に閾値付き ElGamal 暗号，匿名通信路に，ElGamal 暗号を用いた MIX-NET と，Diffie-Hellman 鍵共有と DES によるハイブリッド MIX を用いて実装されている．

一方，本論文の投票システムは，秘密カウンタを用いることで，匿名通信路を不要にしている．逆に，カ

表 8 コスト比較
Table 8 Order.

	2)	本システム
投票者	$O(1)$ 0.5 sec	$O(\log m)$ 23.3 sec
匿名通信路	$O(m)$ 2.52 sec	N.A.
開票者	$O(1)$ 3.05 sec	$O(\log m)$ 1 sec

図 2 動作画面
Fig. 2 Screen shot of the system.

表 9 鍵のビット $|p|$ の違いによる処理時間 [s]（$n = 3$, D503is）
Table 9 Processing time for key size $|p|$ [s] ($n = 3$, D503is).

$\|p\|$ [bit]	Key	Vote	Total
128	2.8	25.4	41.7
256	5.4	92.7	145.8
512	3.2	426.7	443.6

ウンタ値の更新においては投票者の負担が大きい．そのため，単純に両システムを比較することができないが，違いを明確にするために，ここでは，通信（計算）コストのオーダと，単一の実処理時間（コンスタント）を**表 8** に整理する．この比較では，文献 2) の表 1 のデータの Simle MIX を用いている☆．本システムは，図 4 より，$m = 10$ のとき（$n = 4$）の処理時間を試算した．

表 8 より，本システムは，全ユーザ数に比例した通信コストがかかる．MIX に相当する処理が存在せず，開票時のコストも，対数オーダであり，効率が良い．その反面，投票者の処理オーダも実時間も，文献 2) よりもはるかに高く，制約が多い．

4.3 実機での動作

次に，NTT Docomo D503is を用いた動作テストを行った（**図 2**）．**表 9**，**表 10** に測定結果を示す．ここで，Vote をカウンタの内部状態の読み込み，再暗号化，送信を含んだ投票処理の時間とする．4.1 節と同様に，表 9 は $n = 3$，表 10 は $|p| = 512$ で測定した．以上の結果を**図 3** と**図 4** に図示する．図より，

☆ ユーザ数 $m = 10$, プラットフォームは, Pentium II 400 MHz, 掲示板（bubo）や管理者（Admin）での署名処理は省略している．

表 10 レジスタのビット n の違いによる処理時間 [s]（$|p| = 512$, D503is）
Table 10 Processing time for bit length of register n [s] ($|p| = 512$, D503is).

n	Key	Vote	Total
3	3.2	426.7	443.6
5	7.3	664.6	686.2
8	5.2	1055.4	1084.7

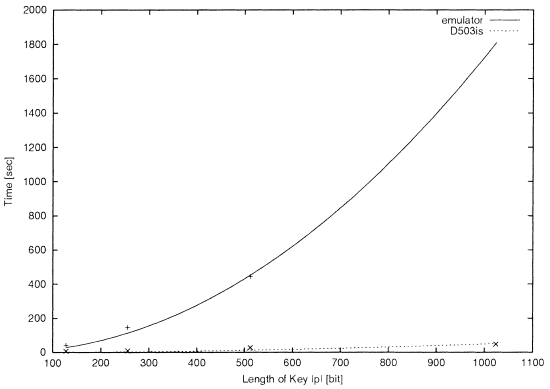


図 3 鍵のビット $|p|$ の違いによる処理時間
Fig. 3 Transaction time for key size $|p|$ [bit].

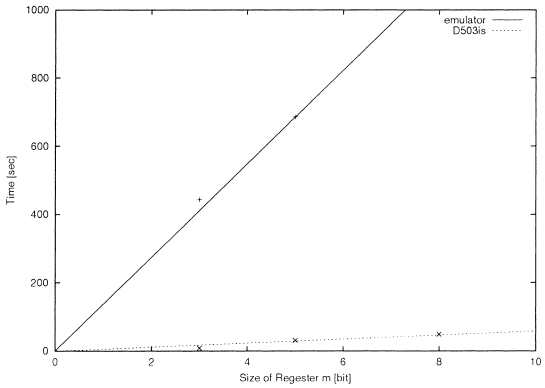


図 4 レジスタのビット n の違いによる処理時間
Fig. 4 Transaction time for bit length of register n.

レジスタのビット数 n を変化させた場合は n に比例して投票時間が変化していることが分かる．また，鍵のビット数 $|p|$ を変化させると線形以上の多項式オーダで投票時間が変化することが分かった．

図 4 より，エミュレータと実機での処理速度の違いは約 23 倍であることが分かる．ここで，ファイルの読み込み時間はエミュレータとの違いが 5 倍程度であるが，暗号化の処理に 20 倍以上の時間を要してしまっている．この結果より，携帯情報端末では冪乗計算などの処理に向いていないことが分かる．

表 12 運用レベルについて必要な計算機パワー向上の条件
Table 12 Estimated improvement of CPU power necessary for several scale.

運用レベル	ユーザ数 m	レジスタ n	事前処理なし		事前処理あり	
			エ［倍］	実機［倍］	エ［倍］	実機［倍］
クラスの学級委員	32	5	1	23.5	1	5.17
学生生徒会	1,000	10	1	45.7	1	10.1
市長選	1 万	14	2.6	60.7	1	13.4
県知事選	300 万	22	4.2	98.4	1	21.6

エ：エミュレータの略（Pentium III 1.13 GHz）
実機：D503is

表 11 削減された投票時間の見積り [s]（$|p| = 512$）
Table 11 The estimated reduction of voting time [s] ($|p| = 512$).

n	Original	Enc	Reduced
3	443.6	348.8	94.8
5	686.2	539.5	146.7
8	1084.7	852.8	231.9

Original：投票時間の計測値
Enc：表 7 より試算した暗号化時間
Reduced：削減された投票時間

ここで，鍵のサイズ $|p|$ を 512 ビットで，$m =$ 約 10 万人規模の選挙に適応した場合の処理時間を見積もってみよう．レジスタのビット数 $n = 17$ ($> \log_2 100{,}000$) となるため，携帯電話端末での投票は，1 人あたり約 37 分の投票時間であることが分かる．

このままでは，実用化は困難であるので，処理削減の可能性を検討する．表 6，表 7 のエミュレータの処理時間より，そのほとんどの処理が暗号化（再暗号化）の処理であることが分かる．処理時間のほとんどは冪乗演算に費されている．この処理は，2.2 節で説明した式 (2) のレジスタ更新式

$$C_i = E[1] \times A_{i-1} \times A_n^{a_i - 1}$$

における，1 の暗号化 $E[1]$ の計算に相当する．平文 “1” は，カウンタの値にも，投票値にも依存しないので，事前処理が可能である．この事前処理は，PDA 内部に蓄えた暗号文を参照されない限り，プロトコルの安全性を保っている．これにより，大幅な効率化が期待できる．この大きさを，エミュレータ上の測定結果の表 7 のデータに基づいて，見積もった結果を**表 11** に示す．すなわち，事前処理によって，約 22%にまで時間を縮めることができていることを示している．

以上のデータを基にして，本システムの運用可能条件を見積もろう．ここで，512 ビットの ElGamal 暗号を用いたとき，投票待ち時間の上限を 30 秒と仮定する．このとき，ユーザ数として，学級委員選挙の $m = 32$ から，県知事選挙の $m = 300$ 万人までの 4 レベルについて，条件を満たすための CPU パワーを実験データから試算した．結果を**表 12** に示す．前述

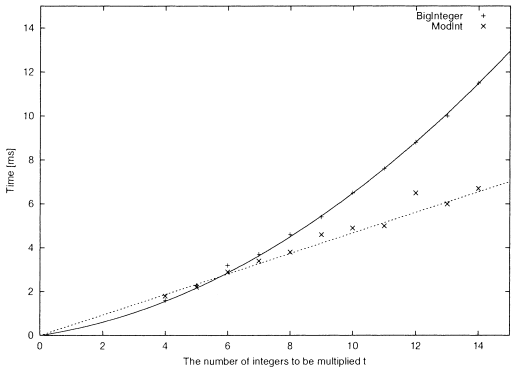


図 5 `ModInt` と `BigInteger` の処理時間 [ms]
Fig. 5 Processing time for `ModInt` and `BigInteger` [ms].

の事前処理により，実測値の 22%まで処理時間が短縮されていると見なしている．

たとえば，1,000 人規模の選挙を行うためには，現在の携帯電話の CPU パワーを 10.1 倍高める必要がある．さらに，`ModInt` などの多倍長演算の改良をすることによって，さらなるパフォーマンスの向上が考えられる．

4.4 `ModInt` のパフォーマンス

`ModInt` が `BigInteger` に対して最も効果的なのは，乗算を繰り返す場合である．`BigInteger` では，まず乗算を行ってから剰余をとるので，整数長がいったん伸びてしまう．たとえば，1,024 ビットの数 a，b，c をかけるとき，`BigInteger` で

```
a.multiply(b).multiply(c).mod(p)
```

と書くと，3,072 ビットの数に対して $\bmod p$ がとられる．一方，`ModInt` の場合は 1,024 のままである．そこで，乗算の回数 t（上の例では $t = 3$）についての，処理時間を測定してみた．**図 5** に測定結果を示す．

この結果より，$t > 6$ では，`ModInt` の方が効果的であることが明らかになった．6 未満でその処理が劣るのは，主にコードの最適化による．

本システムに関しては，中心的な役割を果たしている ElGamal 暗号の高速化に応用されている．ただし，式 (2) の $t = 3$ の場合が最大であり，$t > 6$ が生じる

ことはない．

5. おわりに

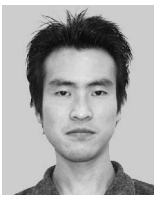
本論文では，文献 5) を携帯情報端末上で動作可能なように Java を用いて実装を行った．暗号演算のためのクラスを携帯情報端末用に開発した．本実装により，携帯電話端末のような CPU パワーの低い環境からでも，ある条件の下でセキュアな電子投票を実現するための基本動作が正しく行えることを確かめた．実験によると，300 万人規模の電子選挙を携帯情報端末から実行するためには，現行の約 22 倍の CPU パワーを必要とすることを示した．また，事前処理により，投票者の処理時間が約 22%削減されることが明らかになった．開発した `ModInt` クラスは，法 p の値を暗黙的に持っており，`BigInteger` クラスより，高い記述性を実現した．

謝辞 本研究の遂行にあたって多大な協力をいただいた東海大学工学部電気工学科大久保学氏に感謝する．

参考文献

1) 国政選への導入は?，YOMIURI ON LINE. http://www.yomiuri.co.jp/atmoney/net/net02062403.html（2002 年 12 月現在）
2) 藤岡 淳，阿部正幸，大久保美也子，星野文学：実用的な電子投票方式の実装および実験，*Proc. Symposium on Cryptography and Information Security*, Vol.48 (2000).
3) Abe, M.: Universally Verifiable Mix-Net with Verification Work Independent of the Number of Mix-Servers, *IEICE Trans. Fundamentals*, Vol.E83-A (2000).
4) Furukawa, J. and Sako, K.: An Efficient Scheme for Proving a Shuffle, *Proc. CRYPTO '01*, LNCS 2139, pp.368–387 (2001).
5) 中里純二，菊池浩明，中西祥八朗：秘密線形フィードバックシフトレジスタを用いた電子選挙システムの実装，情報処理学会マルチメディア・分散・協調とモバイル（DICOMO2002），pp.97–100 (2002).
6) 中里純二，菊池浩明，中西祥八郎：秘密線形フィードバックシフトレジスタを用いた電子選挙システムの提案，情報処理学会論文誌（投稿中）(2003).
7) 平成 13 年度 IT による家族への影響実体調査. http://www5.cao.go.jp/seikatsu/2002/0405it-chousa/index.html（2003 年 4 月現在）
8) 菊池浩明，中里純二，中西祥八朗：秘密カウンタ，信学技報 ISEC2001-25, pp.45–51 (2001).
9) NTT Docomo オフィシャルページ. http://www.nttdocomo.co.jp/（2002 年 9 月現在）
10) i-mode オフィシャルページ. http://www.nttdocomo.co.jp/p_s/imode/（2002 年 9 月現在）
11) 中里純二，菊池浩明，中西祥八朗：秘密カウンタによる電子投票システム，情報処理学会 DPS ワークショップ（DPSW2001），pp.115–119 (2001).
12) Kikuchi, H.：秘密カウンタ 2，情報処理学会コンピュータセキュリティシンポジウム（CSS2001），pp.367–372 (2001).
13) Katz, J., Myers, S. and Ostrovsky, R.: Cryptographic Counters and Applications to Electronic Voting, *Proc. EUROCRYPT '02*, pp.78–92 (2001).
14) Crammer, R., Damgard, I. and Schoenmakers, B.: Proofs of partial knowledge and simplified design of witness hiding protocols, *Proc. CRYPTO '94*, pp.174–187 (1994).
15) 岡本龍明，山本博資：現代暗号，産業図書 (1997).

(平成 14 年 11 月 29 日受付)
(平成 15 年 6 月 3 日採録)



中里 純二（学生会員）

平成 13 年東海大学工学部電気工学科卒業．平成 15 年同大学院博士前期課程修了．現在，同大学院工学研究科博士後期課程在学中．コンピュータセキュリティ，暗号・情報セキュリティの研究に従事．電子情報通信学会会員．

菊池 浩明（正会員）

1988 年明治大学工学部電子通信工学科卒業．1990 年同大学院博士前期課程修了．1990 年（株）富士通研究所入社．1994 年東海大学工学部電気工学科助手．1995 年同専任講師．1999 年同助教授，1997 年カーネギーメロン大学計算機科学部客員研究員．2000 年東海大学電子情報学部情報メディア学科助教授，現在に至る．博士（工学）．ファジィ論理，多値論理，ネットワークセキュリティに興味を持つ．1990 年日本ファジィ学会奨励賞，1993 年情報処理学会奨励賞，1996 年 SCIS 論文賞．電子情報通信学会，日本ファジィ学会，IEEE，ACM 各会員．

中西祥八郎

1967 年東海大学工学部電気工学科卒業．1969 年同大学院博士前期課程修了．同年同大学工学部電気工学科助手．1971 年同専任講師，札幌校舎勤務，1973 年同湘南校舎勤務．1985 年同助教授．1991 年同教授，2000 年同電子情報学部情報科学科教授，現在に至る．工学博士．日本ファジィ学会，電気学会，計測自動制御学会，システム制御情報学会，日本神経回路学会，日本経営工学会，IEEE，IFSA 各会員．