SCIS2000 The 2000 Symposium on Cryptography and Information Security
Okinawa, Japan, January 26-28, 2000

Permutation Network を利用したより効率的な Mix-net
More Efficient Mix-Network on Permutation Networks

星野 文学* 阿部 正幸*
Fumitaka Hoshino Masayuki Abe

あらまし 阿部が提案した, 小中規模無記名電子投票向き Universally Verifiable Mix-net の 2 つの構成法 (1-pass 方式 と 2-pass 方式) の, それぞれの利点を継承する方式を提案する. 新しい方式は従来同様, 検証性, 匿名性, 頑健性などの安全性の要件を満たしている. さらに新しい方法は阿部の 2-pass 方式と計算量的に同等でありながら, 1-pass を実現している. 本論文ではさらに演算方法の検討も行い, 現実的にどこまで速くプロトコルを実行できるか追求する.

キーワード Mix-net, 電子投票, 匿名性, 置換網

1 はじめに

Mix-net は確率暗号による暗号文の列を入力とし, 各暗号文を復号して得られる平文をランダムな順序で出力する, 一種の匿名通信路である. インターネットのような必ずしも匿名性が保証されないネットワーク上でもこのプロトコルを用いることによって匿名性を確保することができる. 従来, 匿名性の他に検証性や頑健性を持つ方式は理論的には可能であったが, 実用性が乏しかった. しかし近年, 検証性や頑健性だけでなく実用性をも兼ね備えた方式に対する関心が高まっており, 阿部により提案された方式 (1-pass 方式と 2-pass 方式) はこれに答えるものである. (文献 [1][2]) 本論文では匿名性, 検証性, 頑健性を保持したまま 1-pass 方式よりも, さらに実用性を追求したプロトコルの提案を行うとともに, 演算方法の改善等による効率の追求も行う.

2 安全性

本論文では安全性に関する議論は行わないので形式的な定義は行わず, 用語の意味が分かる程度に直観的な説明を行う. 以下, 暗号文の解読にはセキュリティーパラメタの準指数時間の計算が必要であるとし, プロトコルの参加者は多項式時間の計算能力しかないと仮定する.

2.1 正当性

Mix-net や Mix サーバへの入力暗号文の列とその出力に復号規則に基づく "入出力対応" が存在すれば, Mix-net や Mix サーバを正当という.

ここで注意すべき事は, 例えば yes/no voting のような場合には, 復号規則に基づく "入出力対応" は一意ではなく複数存在するということである. どの "入出力対応" も正当性の根拠となる[1] .

2.2 検証性

Mix-net や Mix サーバが "正当か否か" を任意の第三者が圧倒的確率で正しく出力できる多項式時間アルゴリズムが存在すれば Mix-net は検証可能であるという.

2.3 匿名性

n 入力の Mix-net において, 任意の t_u 個の入力暗号文及び任意の t_m 個の Mix サーバを操作出来るとき, 残りの $n-t_u$ 個の入力暗号文に関して, この Mix-net が $n-t_u$ 入力の Mix-net のように振舞うならこの Mix-net は (t_u, t_m)-匿名性をもつという. $1 \leq t_u \leq n-2$ なる t_u に対して (t_u, t_m)-匿名ならば $(t_u - 1, t_m)$-匿名性は自明であるので, $(n-2, t_m)$-匿名性の事を単に t_m-匿名性と呼ぶ.
ここで注意すべきことは, 0 入力 や 1 入力 の Mix-net は存在しないということである. 0 入力 や 1 入力 の Mix-net が存在しない以上, (n, t_m)-匿名性 や $(n-1, t_m)$-匿名性は無意味な概念である. 2 人で『ババヌキ』をやるとつまらないのと同じ理屈である.

2.4 頑健性

任意の t_u 個の入力暗号文及び任意の t_m 個の Mix サーバを操作出来るとき, どのような操作を行っても多項式時間でプロトコルが完了し, 圧倒的確率で出力が正当

* NTT 情報流通プラットフォーム研究所, 〒 239-0847 神奈川県横須賀市光の丘 1-1, NTT Information Sharing Platform Laboratories, 1-1 Hikarinooka, Yokosuka-Shi, Kanagawa 239-0847, JAPAN

[1] 実際に行われた置換とは異る" 入出力対応" でもその置換が分からない限り正しい "入出力対応" と見做される. 量子力学に似ている.

である時 Mix-net は (t_u, t_m)-頑健であるという．特に (n, t_m)-頑健のことを単に t_m-頑健という．

3 準備

2-pass 方式, 1-pass 方式, 提案法は, いずれも効率的な**置換網** (Permutation Network) の存在 (文献 [4]) が実用性の基礎となっている．置換網とは, 任意の置換 Π 及び n 個の文の列を入力して, 入力文の列を Π によって置換した文の列を出力する回路または通信網のことである．

3.1 t-匿名 Mix-net

n 入力の置換網を $t+1$ 個直結した通信路を考える．但し, この通信路は 暗号文の列を入力として, 各暗号文を復号した結果をランダムに並べ換えた文の列を出力とする付加的な機能の付いた通信路 (Mix-net) であるとする．この通信の処理を, どの Mix サーバも 2 つ以上の置換網の処理を行わないよう m 個の Mix サーバ が分担して実行することとする．従って $m \geq t+1$ である．

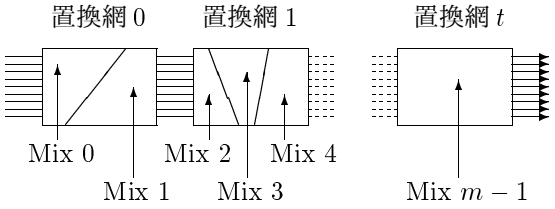

図 1: t-匿名 Mix-net

定理 1 上記 *Mix-net* は t-匿名である．

証明は省略. $m = t+1$ なら従来の Mix-net である．

3.2 検証可能 Mix-net

Mix-net において, 各 Mix が, 置換に関する情報を一切漏らさずに入出力対応の存在を任意の者に対して証明できればこの Mix-net は検証可能である．

3.3 t-頑健 Mix-net

m 個の Mix サーバを持つ Mix-net を t-頑健にする為には, m 個のうち t 個の Mix サーバが停止するか, 不正な出力を行った場合でもプロトコルが完了し Mix-net の出力が圧倒的な確率で正当でなくてはならない．従って, t-頑健 Mix-net を実現するためには少なくとも Mix サーバが 他の Mix サーバの正当性を検証できなくてはならないが, 検証可能 Mix-net であればこの事は問題ではない[2] . Mix-net を使って暗号文の復号処理を行う場合は, 正しく動作する $m-t$ 個が不正な t 個の代替処理を行なう必要がある．即ち, どの $m-t$ 個の Mix サーバが集まっても全ての暗号文を復号できる必要がある．この目的のためには, 全ての Mix サーバの秘密鍵を, どの $m-t$ 個の Mix サーバが集まっても復元できるように VSS で秘密分散すれば十分である．

[2] Mix サーバだけが正当性を検証できるとするモデルもあるが, この弱い意味での検証性は本論文では検証可能とは呼ばない.

3.4 t-匿名 t-頑健 Mix-net

上記 t-頑健 Mix-net は少なくとも $(m-t)$-匿名ではあり得ない．この意味で頑健性は匿名性と対立する概念である．この Mix-net が少なくとも t-匿名である為には $m-t-1 \geq t$ が必要である．即ち t-匿名 t-頑健 Mix-net を実現するには $m \geq 2t+1$ として上記の n 入力の置換網を $t+1$ 個直結した通信路をどの Mix サーバも 2 つ以上の置換網の処理を行わないよう m 個の Mix サーバ が分担し, 全ての秘密鍵を, どの $t+1$ 個の Mix サーバが集まっても復元でき, どの t 個の Mix サーバが集まってもどの秘密鍵も復元できないように VSS で秘密分散すれば良い．

3.5 置換器

置換器とは 2 入力の置換網のことであり, 1 bit の制御信号 $b \in \{0,1\}$ 及び 2 個の文を入力して, $b=0$ の 場合 2 個の 入力文そのものを出力し, $b=1$ の 場合入力文を入れ換えた 2 個の文を出力する装置のことである．制御信号 b を置換パラメタと呼ぶ.(図 2 参照)

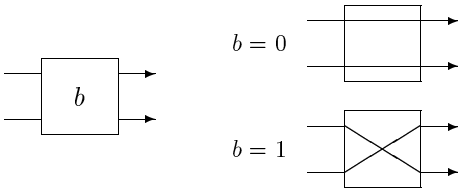

図 2: 置換器

3.6 置換網

置換器を幾つか組み合わせて入力数がより大きい置換網を作ることができる．

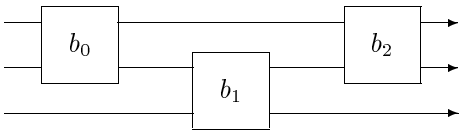

図 3: 置換器で作った 3 入力置換網

入力数が大きい時, 幾つかの 置換器の処理は同時に実行できることがある．同時に実行される (と見做される) 複数または単一の置換器は 1 段と数える.(文献 [1][2])

定理 2 n が 2 の冪であるとき $n\log_2 n - n + 1$ 置換器からなる, $2\log_2 n - 1$ 段の n 入力置換網が存在する．

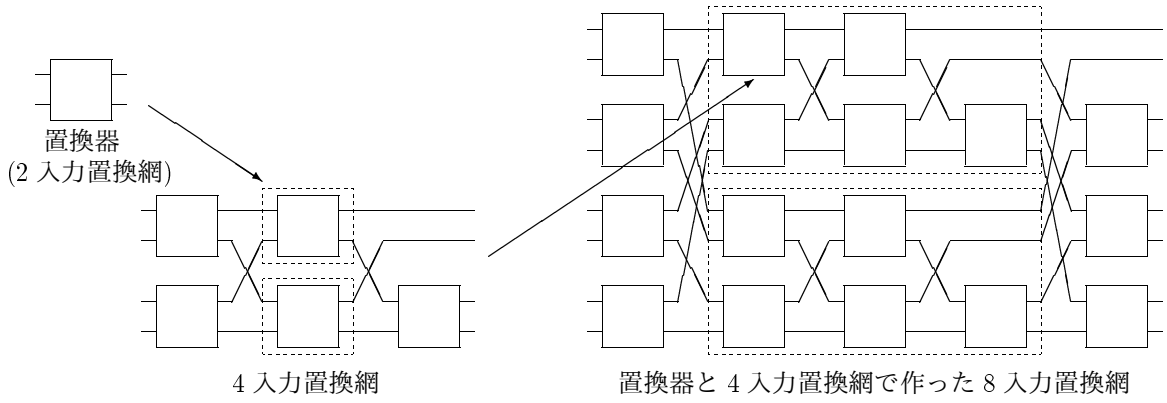

図 4: 2 冪入力置換網の構成

証明及び置換網の具体的構成方法は文献 [4] 参照. 一般に n 入力置換網と置換器を用いて効率的に $2n$ 入力置換網を作ることが出来る. 従って 2 入力置換網 (置換器) から始めて, 4 入力,8 入力,16 入力,$\cdots$ と次々に大きい置換網を作ることが出来る. (図 4 参照)

3.7 プロトコルの設計方針

もし ゼロ知識証明で入出力対応の存在を証明できる, 2 入力の Mix サーバを構築できれば, それを置換器として, 上記手法に従い置換網や Mix-net を構築することにより, 計算量が $O(n \log n)$ の実用的な t-匿名 t-頑健 検証可能 Mix-net を構築することができる. 従って検証可能な置換器の設計を行えば十分である. 以下, 置換器等への入力とその出力, 証明は全て公開されるとする.

4 従来法

入力暗号文に使用する暗号は ElGamal 暗号として, 主に基本単位となる置換器の構成を示す.

4.1 2-pass 方式

本方式では, 簡単のため上記 Mix-net において, 置換およびランダム化処理のみを行い, 復号処理は別途行う. いま平文 m, 生成元 g, 公開鍵 y に対する ElGamal 暗号文 (M, G) は 乱数 t を用いて $(M, G) = (my^t, g^t)$ と暗号化されているとする. (M, G) を受け取った Mix サーバは秘密の乱数 r を用いて $(M', G') = (My^r, Gg^r)$ を計算することが出来る. (M, G) に対して (M', G') は, 同じ平文 m , 同じ生成元 g, 同じ公開鍵 y に対する異る乱数 $t' = t + r$ による ElGamal 暗号文となっている. この処理をランダム化という.

本方式では基本単位たる 置換器は 2 つの入力 ElGamal 暗号文 を秘密の乱数 r_0, r_1 を各々に用いてランダム化した ElGamal 暗号文を 1 ビットの秘密の乱数 b (置換パラメタ) を用いて置換を行う. ランダム化を行う

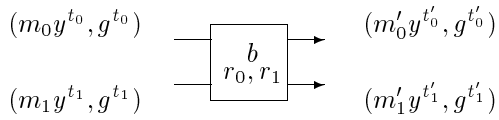

図 5: 置換及びランダム化

ことによって入力と出力の値から置換パラメタ b の値が推定出来ることを防いでいる. ランダム化置換処理において, 入出力対応の存在を証明するには

$$(m_0 = m'_0 \wedge m_1 = m'_1) \vee (m_0 = m'_1 \wedge m_1 = m'_0)$$

をゼロ知識証明で示せば良い. 詳細は文献 [1][2] 等参照.

4.2 1-pass 方式

本方式では, 上記 Mix-net において, 置換およびランダム化処理及び復号処理を行う. いま平文 m, 生成元 g, 公開鍵 y に対する ElGamal 暗号文 (M, G) は 乱数 t を用いて $(M, G) = (my^t, g^t)$ と暗号化されているとする. (M, G) を受け取った Mix サーバは秘密の乱数 r を用いて $(M', G') = (My^r, Gg^r)$ を計算し, 秘密鍵 x' を用いて $(M'', G'') = (M'G'^{-x'}, G')$ を計算出来る. (M, G) に対して (M'', G'') は, 同じ平文 m , 同じ生成元 g に対する異る公開鍵 $y' = yg^{-x'}$ 異る乱数 $t' = t + r$ による ElGamal 暗号文となっている. この処理をランダム化復号という.

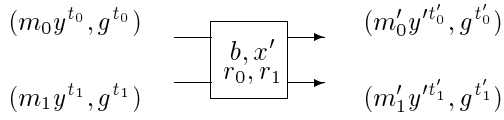

図 6: 置換及びランダム化復号

本方式では基本単位たる置換器は 2 つの入力 ElGamal 暗号文 を秘密の乱数 r_0, r_1 及び秘密鍵 x' を各々に用いてランダム化復号した ElGamal 暗号文を , 1 ビットの秘密の乱数 b(置換パラメタ) を用いて置換する. 上記処理において, 入出力対応の存在を証明するには

$$(m_0 = m'_0 \wedge m_1 = m'_1) \vee (m_0 = m'_1 \wedge m_1 = m'_0)$$

及び部分復号に正しい x' が使われていることをゼロ知識証明で示せば良い. 同じ段における部分復号は全て同じ秘密鍵を用いる. また, 段の中で置換が行われない経路では, 同じ部分復号を補償する装置 (復号器) を用いる. 復号器は, 部分復号に正しい x' が使われたことのみをゼロ知識証明で示せば良い. 詳細は文献 [1][2] 等参照.

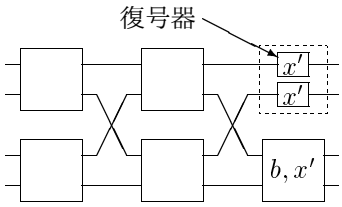

図 7: 部分復号の補償 (4 入力置換網の場合)

5 提案法

5.1 基本的なアイディア

1-pass 方式で用いられるゼロ知識証明は 2-pass 方式で用いられるものより幾分複雑である. 提案法はこの複雑さを解消する為の方法である. まず, 1-pass 方式で用いられる置換器と等価な回路を 2-pass 方式の置換器 1-pass 方式の復号器の組合せで作ることができる.

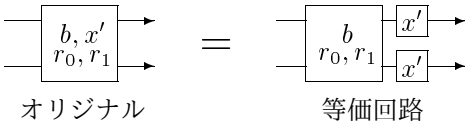

図 8: 1-pass 置換器と等価な回路

この回路で 1-pass 方式に相当する Mix-net を作ったとする. この Mix-net の復号器は, 同じ段の復号器を全て同時に移動するとして, 段を飛び越して後ろに移動しても安全性は損なわれない (図 9). また, 連続する復号器は 1 つの復号器にまとめることができる. もし, 全ての復号器を Mix-net の一番後ろに持って行けば, それは 2-pass 方式に相当する. 提案法では 1-pass 方式の 1-pass 性を損なうことがないよう Mix-net の一番後ろではなく Mix-サーバの一番後ろにこれを移動する. 従って, この方式の計算量は 2-pass 方式と同じであると考えて良い. また, この方法は 2-pass 方式の最初の pass において, Mix-サーバの一番後ろに部分復号処理を挿入

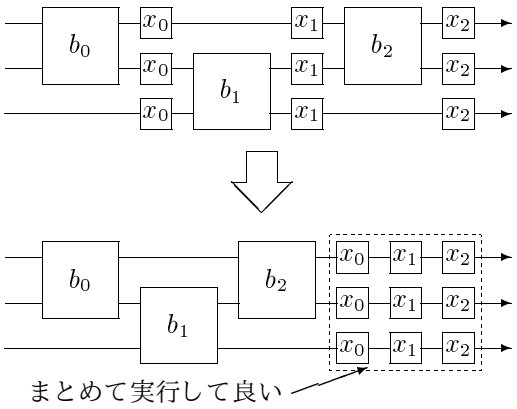

図 9: 部分復号はまとめて良い

した方式と考えることも出来る. 従って置換網の分担の割り当ては 2-pass 方式同様自由にすることが出来る[3] .

5.2 ゼロ知識証明の効率化

各プロトコルのゼロ知識証明に関して

$$(m_0 = m'_0 \wedge m_1 = m'_1) \vee (m_0 = m'_1 \wedge m_1 = m'_0)$$

を示す代わりに

$$(m_0 = m'_0 \vee m_0 = m'_1) \wedge m_0 m_1 = m'_0 m'_1$$

を示せば十分であり, 効率が良い (文献 [3]).

5.3 ランダム化置換プロトコル詳細

(M_0, G_0) を平文 m_0 生成元 g 公開鍵 y 乱数 t_0 に対する ElGamal 暗号文 $(M_0, G_0) = (m_0 y^{t_0}, g^{t_0})$ とする. (M_1, G_1) を平文 m_1 生成元 g 公開鍵 y 乱数 t_1 に対する ElGamal 暗号文 $(M_1, G_1) = (m_1 y^{t_1}, g^{t_1})$ とする. $r_0, r_1, r, R_0, R_1, R, e$ を秘密の乱数とする. b を 1 bit の秘密の乱数 (置換パラメタ) とし, $\bar{b}$ を $\bar{b} = b \oplus 1$ とする. h は公開されたハッシュ関数とする.

入力: $(M_0, G_0), (M_1, G_1)$
出力: $(M'_b, G'_b) = (M_0 y^{r_0}, G_0 g^{r_0})$
$(M'_{\bar{b}}, G'_{\bar{b}}) = (M_1 y^{r_1}, G_1 g^{r_1})$
証明: (1) $(m_0 = m'_0 \vee m_0 = m'_1)$ の部分

$$\begin{aligned}
(T_b, W_b) &= (y^{R_b}, g^{R_b}) \\
(T_{\bar{b}}, W_{\bar{b}}) &= (y^{R_{\bar{b}}} (M'_{\bar{b}}/M_0)^e, g^{R_{\bar{b}}} (G'_{\bar{b}}/G_0)^e) \\
e_b &= -e + h(T_0, T_1, W_0, W_1) \\
e_{\bar{b}} &= e \\
z_b &= R_b - e_b r_0 \\
z_{\bar{b}} &= R_{\bar{b}}
\end{aligned}$$

[3] 1-pass 方式でも部分復号の回数を増やせば可能.

として $(T_0, T_1, W_0, W_1, e_0, e_1, z_0, z_1)$ を出力

証明: (2) $(m_0 m_1 = m'_0 m'_1)$ の部分

$$\begin{aligned}(T, W) &= (y^R, g^R)\\ c &= h(T, W)\\ z &= R - c(r_0 + r_1)\end{aligned}$$

として (T, W, z) を出力

検証: $c = h(T, W)$ として

$$\begin{aligned}e_0 + e_1 &= h(T_0, T_1, W_0, W_1) &\wedge\\ 1 &= y^{z_0}(M'_0/M_0)^{e_0} T_0^{-1} &\wedge\\ 1 &= y^{z_1}(M'_1/M_0)^{e_1} T_1^{-1} &\wedge\\ 1 &= g^{z_0}(G'_0/G_0)^{e_0} W_0^{-1} &\wedge\\ 1 &= g^{z_1}(G'_1/G_0)^{e_1} W_1^{-1} &\wedge\\ 1 &= y^z (M'_0 M'_1/M_0 M_1)^c T^{-1} &\wedge\\ 1 &= g^z (G'_0 G'_1/G_0 G_1)^c W^{-1}\end{aligned}$$

ならば出力は正当.

5.4 部分復号プロトコル詳細

(M, G) を平文 m 生成元 g 公開鍵 Y 乱数 t に対する ElGamal 暗号文 $(M, G) = (mY^t, g^t)$ とする. x を秘密鍵とし, $y = g^x$ とする. r を秘密の乱数とする. h を公開されたハッシュ関数とする.

入力: (M, G)

出力: $(M', G') = (MG^{-x}, G)$

証明:

$$\begin{aligned}(T, W) &= (G^r, g^r)\\ c &= h(T, W)\\ z &= r - cx\end{aligned}$$

として (T, W, z) を出力.

検証: $c = h(T, W)$ として

$$\begin{aligned}1 &= G^z (M/M')^c T^{-1} &\wedge\\ 1 &= g^z y^c W^{-1}\end{aligned}$$

ならば出力は正当.

5.5 検証の効率化

提案法や 1-pass 方式, 2-pass 方式の律速は出力結果の検証である. 従ってこの部分には特に工夫を凝らす. 置換網を使った方式は, Mix-net 全体で大量の検証処理を行う. 検証処理は基本的に Schnorr 署名の検証処理の組み合わせであり, 大量の冪乗計算が発生する. K 個の冪乗を別々に計算するコストを 1 とすると, K 基底の冪乗積を計算するコストは K が十分大きい場合には 1/3 に過ぎない. さらに, 同じ基底がある場合は, これをまとめて計算する事が可能である. 即ち大量の検証は一括して行う事 (batch verification) により, これを効率的に実行出来る. 詳細については文献 [5] 参照.

5.5.1 事前計算の導入

あらかじめ基底の分かっている冪乗は table を作成する事によって計算コストを 1/3 に軽減できる. さらに指数も予め計算出来るなら冪乗は事前計算するのが効率的である.

6 効率

例えば ElGamal 暗号の定義群に楕円曲線上の有理点群を用いる場合は指数部 (スカラー部) で用いられる積は軽微であると見なせる. 定義群に Sophie Germain 素体の乗法群を用る場合は, 指数部で用いられる積と定義群上の積とは等価なコストを必要とするが, それでもセキュリティパラメタに依存しない項は軽微であると見なせる. 従って, 暗号のセキュリティパラメタに依存しない項を無視して, 効率の計算を行う. 指数部のサイズを $k+1$ bit として乗算 k 回のコストを m とし, 自乗 k 回のコストを s とする. 使用する冪乗アルゴリズムは簡単の為, 下表の 2 通りのみとする.

表 1: 冪乗算の平均コスト

冪乗アルゴリズム	平均コスト
(1) binary 法	$s + m/2$
(2) 冪乗 table 参照法	$m/2$
冪乗 table 作成	s

各処理に対して, アルゴリズム (2) を適用している部分で指数部を事前に作成出来る場合は前処理が可能である. 前処理を行った場合の評価には (前) が付いている.

表 2: ランダム化置換器 1 個当たり平均コスト

処理	平均コスト	内訳
出力	$2m$	$(2) \times 4$
出力 (前)	0	
証明	$2s + 4m$	$(1) \times 2 + (2) \times 6$
証明 (前)	$2s + m$	$(1) \times 2$

表 3: 部分復号器 1 個当たり平均コスト

処理	平均コスト	内訳
出力	$s + m/2$	$(1) \times 1$
証明	$s + m$	$(1) \times 1 + (2) \times 1$
証明 (前)	$s + m/2$	$(1) \times 1$

検証は batch verification で行うこととする. 置換器を N 個使うとし, 復号器を M 個使うとする. Mix-net 全体の正当性を検証する為のコストを表 4 に示す.

表 4: 検証コスト

処理	N 置換 M 復号コスト	内訳
検証	$s + (6N + 2M + 1)m$	batch verification

上記 出力, 証明, 検証のコストを置換器 1 個当たりのコストに換算して文献 [1] の 1-pass 方式の値と比較したい. 比較の便宜を考慮して, 文献 [1] と同様に $s = m$ とし, 冪乗算法 (1) の処理時間を単位とし, $\lambda = M/N$ とする. 事前演算を許すとして各処理コストを置換器 1 個当たりのコストに換算したものを表 5 に示す.

表 5: 平均換算コスト

処理	1 置換当りコスト
出力	λ
証明	$2 + \lambda$
検証	$4 + \frac{4}{3}\lambda$

単位:算法 (1) の平均コスト

入力数 n を 2 の冪 $n = 2^w$ とする. Mix 数 を $2t + 1$ とし, $t = 5$, $w = 10$ (即ち $n = 1024$) で計算し, 文献 [1] の値と比較する.

$$\begin{aligned} M &= (2t+1) \times 2^w &&= 11264 \\ N &= (t+1) \times ((w-1)2^w + 1) &&= 55302 \end{aligned}$$

より $\lambda = 0.20$ として表 5 の値を計算すれば良い.(表 6)

表 6: 平均コスト

方式	出力	証明	検証
1-pass[1]	4.4	9.52	10.24
提案法	0.20	2.20	4.27

単位:算法 (1) の平均コスト

21264 500MHz 上で 3 次 OEF を使った 160 bit 楕円 ElGamal 暗号を想定した場合, 算法 (1) の実行時間は 488.4 μsec と見積もられている (文献 [6]). ネットワークの遅延を無視した正常系で, このプロセッサを用いた場合, 提案法の Mix-net の実行時間の雑な見積は

$$(0.20 + 2.20 + 4.27) \times 488.4\mu\text{sec} \times 55302 = 180.2\text{sec}$$

である. 最初の Mix が稼働してから 約 3 分でプロトコル全体が完了する見積となる[4] . また, Mix-net 全体の正当性の検証にかかる時間は

$$4.27 \times 488.4\mu\text{sec} \times 55302 = 115.3\text{sec}$$

であり, 約 2 分で正当性の検証が完了する.

[4] 前の Mix が証明を幾つか出力する度 (例えば段毎) に検証を行うなら検証だけの時間が実質的な実行時間となり約 2 分で完了する.

7 参考文献

[1] M.Abe. Mix-Networks on Permutation Networks — Advances in Cryptology– ASIACRYPT '99, LNCS 1716,p258-273. Springer-Verlag, 1999

[2] 阿部 正幸. Permutation Network を利用した Mix-net — TECHNICAL REPORT OF IEICE ISEC 99-10 (1999-05), p61-68. 電子情報通信学会

[3] Markus Jakobsson, Ari Juels. Millimix: Mixing in Small Batches — DIMACS Technical Report 99-33, June 1999.

[4] A. Waksman. A permutation network — Journal of the Association for Computing Machinery, 15(1):159-163, January 1968.

[5] Mihir Bellare, Juan A. Garay, Tal Rabin. Fast Batch Verificaion for Modular Exponentiation and Digital Signatures — Advances in Cryptology– EUROCRYPT '98, LNCS 1403,p236-250. Springer-Verlag, 1998

[6] 青木 和麻呂, 小林 鉄太郎, 星野 文学. 楕円曲線暗号の最高速実装 — SCIS2000-B05