SCIS 2005 The 2005 Symposium on Cryptography and Information Security
Maiko Kobe, Japan, Jan.25-28, 2005
The Institute of Electronics, Information and Communication Engineers

汎用的結合可能な電子投票方式について
On Universally Composable Voting

土居 誠司 * 真鍋 義文 * † 岡本 龍明 * ‡
Seiji Doi Yoshifumi Manabe Tatsuaki Okamoto

あらまし 本稿では Universal Composability framework の中で理想的な電子投票機能を定義する．理想機能実現のために，送信者認証機能を持つ Mix-Net(Authenticated Mix-Net:AMN) の理想的な機能および集計の理想機能を定義し，2 つの理想機能が存在する hybrid モデルにおいて電子投票の理想機能を実現するプロトコルを提案する．また，AMN の理想機能および集計の理想機能を実現するプロトコルを示す．

キーワード 汎用的結合可能性, Mix-Net, 電子投票

1 はじめに

電子投票は将来その実現が望まれる技術の 1 つである．電子投票は現在実際に行われている投票と比べ，投票者の利便性が大きく改善される．これまで電子投票を実現する様々な方式が提案されてきた．Mix-net は電子投票の実現の際にによく用いられるプロトコルの 1 つである [10][7][9]．Mix-net は入力としていくつかのパーティから値を受け取り，それらをシャッフルして出力する．出力からはどの値が誰からの入力かがわからないのでメッセージに匿名性を持たせることができる．

Canetti は汎用的結合可能性 (Universal Composablity: UC) と呼ばれる，暗号プロトコルの安全性の新しい概念を提案し，安全性を評価するための枠組みを示した [1]．この枠組みで安全性が証明されたプロトコル (UC 安全なプロトコル) は他のプロトコルと組み合わせて用いてもその安全性が保証される．UC 結合定理 [1] は非常に強力であり，UC 安全なプロトコルを結合して作ったプロトコルは UC 安全であることを保証する．

本稿では UC Framework の概念を用いて Mix-Net を用いた汎用的結合可能な電子投票方式を提案する．まず最初に電子投票の理想的な機能 $\mathcal{F}_{\mathrm{VOTE}}$ と AMN の理想的な機能 $\mathcal{F}_{\mathrm{AMN}}$，集計の理想的な機能 $\mathcal{F}_{\mathrm{TALLY}}$ を定義し，$\mathcal{F}_{\mathrm{AMN}}$ と $\mathcal{F}_{\mathrm{TALLY}}$ が存在する hybrid モデルにおいて $\mathcal{F}_{\mathrm{VOTE}}$ を実現するプロトコルを提案する．続いて，いくつかの理想機能を用いる hybrid モデルで $\mathcal{F}_{\mathrm{AMN}}$ を実現するプロトコルを提案し，$\mathcal{F}_{\mathrm{TALLY}}$ を実現するプロトコルを示す．

* 京都大学大学院情報学研究科, 〒 606-8501 京都市左京区吉田本町, Graduate School of Informatics,Kyoto University, Yoshidahon-machi, Sakyo-ku,Kyoto-shi,Japan, doi@lab7.kuis.kyoto-u.ac.jp
† NTT サイバースペース研究所, 〒 239-0847 横須賀市光の丘 1-1, NTT Cyber Space Laboratories, 1-1 Hiakrinooka,Yokosuka-shi,Japan, manabe.yoshifumi@lab.ntt.co.jp
‡ NTT 情報流通プラットフォーム研究所, 〒 239-0847 横須賀市光の丘 1-1, NTT Information Sharing Platform Laboratories, 1-1 Hiakrinooka,Yokosuka-shi,Japan,okamoto@sucaba.isl.ntt.co.jp

2 理想機能の定義

2.1 $\mathcal{F}_{\mathrm{VOTE}}$

本章では，理想的な投票機能について考える．理想的な投票機能は，票を集計して集計結果を公表する，という通常の選挙機能に加えて，無証拠性と全体検証性の 2 つの性質を満たす必要がある．無証拠性とは，投票者が自分の投票内容を第三者に証明することができない性質である．全体検証性とは，選挙が正しく行われたことをどの参加者も検証することができる性質である．図 1 に投票の理想機能 $\mathcal{F}_{\mathrm{VOTE}}$ を定義する．なお，通常，理想機能を記述する際，理想機能がやりとりをするメッセージには全て，現在呼び出している理想機能をパーティが識別するための番号 *sid* が含まれるが，本稿では記述を簡単にするため，特に必要が無い限り *sid* を省略している．$\mathcal{F}_{\mathrm{VOTE}}$ は，有権者リストを保持しており投票者から票を受け取ると有権者リストに含まれる投票者からの投票のみを集計する．また，参加者から要求があると集計結果を渡す．$\mathcal{F}_{\mathrm{VOTE}}$ は明らかに無証拠性と全体検証性を満足する．

2.2 $\mathcal{F}_{\mathrm{AMN}}$

ここでは AMN の理想的な機能を定義する．これまで様々な Mix-Net プロトコルが提案されてきた [6][7][9][10]．なかでも，Wikström は初めて理想的な Mix-Net の機能

理想機能 $\mathcal{F}_{\mathrm{VOTE}}$

投票者 $P_1, P_2, \cdots, P_n$，選挙 server $M_1, M_2, \cdots, M_k$，敵 S と動作する．また，候補者のリスト $L_C = \{C_1, C_2, \cdots, C_{N_c}\}$ と有権者のリスト $L_P = \{P_1, P_2, \cdots, P_{N_p}\}$ を保持している．

1. $J_M = 0$, $c_i = 0$ $(i = 1, \cdots, N_c)$, $c_{inv} = 0$.
2. P_i から $(\mathtt{Vote}, v_i)$ を受け取る時，$P_i \in L_P$ なら $(\mathtt{vote}, P_i)$ を S へ送信した後，以下の操作を行い，以降の P_i からのメッセージを無視する．$P_i \notin L_P$ であれば何もしない．
   - $v_i = C_j$ $(C_j \in L_C)$ であれば $c_j = c_j + 1$
   - それ以外の場合は $c_{inv} = c_{inv} + 1$
3. M_j から $(\mathtt{Start})$ を受け取ると，$J_M \leftarrow J_M \cup \{j\}$ を行い，$(\mathtt{Start}, M_j)$ を S へ送る．この時，$|J_M| \geq \frac{k}{2}$ であれば $L' = \{\{c_j\}_{j=1}^{N_c}, c_{inv}\}$ を計算する．
4. P_j から $(\mathtt{Read\text{-}Result})$ を受け取ると，3. で L' を計算しているならば，$(\mathtt{Result}, L')$ を返し，$(\mathtt{Read}, P_j)$, $(\mathtt{Result}, L')$ を S へ送る．

図 1: $\mathcal{F}_{\mathrm{VOTE}}$

理想機能 $\mathcal{F}_{\mathrm{AMN}}$

送り手 $P_1, P_2, \cdots, P_n$, Mix-server $M_1, M_2, \cdots, M_k$，敵 S と動作する．リスト L_P を持つ．

1. 各リストの初期化を行う．$L = \phi$, $J_P = \phi$, $J_M = \phi$.
2. $(\mathtt{Send}, m_i)$ を P_i から受け取と $i \notin J_P$ ならば，次の操作を行う．
   - $P_i \in L_P$ ならば，$J_P \leftarrow J_P \cup \{i\}$ を行って，m_i を L に加える．$(P_i, \mathtt{Send})$ を S に送る．
   - それ以外ならば，何もしない．
3. $(\mathtt{Run})$ を M_j から受け取ると，$J_M \leftarrow J_M \cup \{j\}$ を行い，以下の操作を行う．
   - $|J_M| \geq \frac{k}{2}$ ならば，L を辞書順に並び替えて L' を作る．続いて，$(M_j, \mathtt{List}, L')$ を S へ送り，$(\mathtt{List}, L')$ を全ての Mix-server へ送る．
   - それ以外であれば，$(M_j, \mathtt{Run})$ を S へ送る．

図 2: $\mathcal{F}_{\mathrm{AMN}}$

$\mathcal{F}_{\mathrm{MN}}$ を定義した [10]．我々は $\mathcal{F}_{\mathrm{MN}}$ を変更して，送信者認証機能を持つ Mix-Net (Authenticated Mix-Net:AMN) の理想機能 $\mathcal{F}_{\mathrm{AMN}}$ を定義する．図 2 に $\mathcal{F}_{\mathrm{AMN}}$ の動きを示す．$\mathcal{F}_{\mathrm{AMN}}$ はメッセージ送信者のリストを保持しており，入力として受け取ったメッセージのうちリストに載っている送信者からのメッセージのみをシャッフルする．また 1 人の送信者からはメッセージを 1 回しか受け付けない．$\mathcal{F}_{\mathrm{VOTE}}$ により，メッセージの匿名性と，重複投稿の防止を実現でき，また権利を持たない不正な送信者からのメッセージを取り除く．$\mathcal{F}_{\mathrm{AMN}}$ は，まず複数の送り手からメッセージを受け取る．$\mathcal{F}_{\mathrm{AMN}}$ は送り手のリスト L_P を持っており，リストに載っている送り手からのメッセージだけを辞書順に並び替えて出力する．

2.3 $\mathcal{F}_{\mathrm{TALLY}}$

集計の理想機能を図 3 に示す．$\mathcal{F}_{\mathrm{TALLY}}$ には公開鍵が存在し，その公開鍵で暗号化された要素からなるリストを入力として受け取る．集計サーバのうち過半数が協力することでリストの復号，集計することができる．

3 理想機能の実現

本章では，前章で定義した理想機能を実現するプロトコルを示す．各プロトコルはいくつかの理想機能が存在する hybrid モデルで実現される．UC framework では理想機能が動作する理想世界をプロトコルと理想機能が動作する hybrid モデルが模倣することができていれば，そのプロトコルは hybrid モデルにおいて理想機能を安全に実現している，と言う．プロトコルが模倣している

理想機能 $\mathcal{F}_{\mathrm{TALLY}}$

観測者 $P_1, P_2, \cdots, P_n$, 集計サーバ $T_1, T_2, \cdots, T_k$，敵 S と動作する．リスト $L_C = \{C_1, C_2, \cdots, C_{N_C}\}$ が存在する．

1. T_i から (`Tally`,L) を受け取ると，(`Tally`,T_i) を S へ渡す．U_L が存在するなら $T_L \leftarrow U_L \cup \{i\}$．$U_L$ が存在しないならば $U_L = \{i\}$ を作る．
2. $|U_L| \geq \frac{k}{2}$ なる U_L が存在するならば，L を復号して $L' = \{l'_1, l'_2, \cdots, l'_N\}$ を作成して以下の操作を行う．
   (a) $c_j = 0$ $(j = 1, 2, \cdots, N_c)$, $c_{inv} = 0$.
   (b) for j=1 to N
   - $l'_{i,j} = C_h (C_h \in L_C)$ ならば $c_h = c_h + 1$
   - それ以外なら $c_{inv} = c_{inv} + 1$

   (c) $L_{tally} = \{\{c_j\}_{j=1}^{N_c}, c_{inv}\}$ を計算して，(`Result`, L_{tally}) を S へ渡す．
3. P_j から (`Read`) を受け取ると，(`Result-tally`, L_{tally}) を返し，(`Read`, P_i) を S へ渡す．

図 3: 集計の理想機能 $\mathcal{F}_{\mathrm{TALLY}}$

かどうかの判定を行うのが環境 Z である．Z は，現実世界もしくは hybrid モデルのどちらかとインタラクションを行う．Z は，パーティや敵とインターフェイスを持っており，それらからの出力を参考にして，どちらとインタラクションを行ったかを当てる．UC framework においては，プロトコルが理想機能を実現していることの証明はシミュレーションベースで行われる．理想世界の敵 S(シミュレータと呼ぶ) が，hybrid モデルの敵 H の出力を完全にシミュレートすることができればプロトコルは理想機能を実現していると呼ぶ．

3.1 プロトコル π_{VOTE}

$(\mathcal{F}_{\mathrm{AMN}}, \mathcal{F}_{\mathrm{TALLY}})$-hybrid モデルにおいて，$\mathcal{F}_{\mathrm{VOTE}}$ を実現するプロトコル，π_{VOTE} を図 4 に示す．

全ての投票者 P_i は $\mathcal{F}_{\mathrm{TALLY}}$ の公開鍵 y で暗号化された投票を $\mathcal{F}_{\mathrm{AMN}}$ へ渡す．$\mathcal{F}_{\mathrm{AMN}}$ では投票者が認証されて投票がシャッフルされる，シャッフルされた暗号化投票を $\mathcal{F}_{\mathrm{TALLY}}$ が復号して集計する．投票者は $\mathcal{F}_{\mathrm{TALLY}}$ から集計結果を得る．

定理１： プロトコル π_{VOTE} は $(\mathcal{F}_{\mathrm{AMN}}, \mathcal{F}_{\mathrm{TALLY}})$-hybrid モデルにおいて，過半数の選挙サーバは honest であるという仮定の下で，adaptive な敵に対して $\mathcal{F}_{\mathrm{VOTE}}$ を実現する

略証： H を $(\mathcal{F}_{\mathrm{AMN}}, \mathcal{F}_{\mathrm{TALLY}})$-hybrid モデルにおける敵とする．$H$ が π_{VOTE} と動作する $(\mathcal{F}_{\mathrm{AMN}}, \mathcal{F}_{\mathrm{TALLY}})$-hybrid モデルをシミュレートする S を次のように構成する．S は $H, \mathcal{F}_{\mathrm{AMN}}, \mathcal{F}_{\mathrm{TALLY}}$ を内部で実行する．また理想世界では P_i に対応するダミーパーティ $\tilde{P}_i$ が存在する．Z からの S へのメッセージは H に渡され、H から S へのメッセージは Z へ渡される．

1. S が (`vote`,P_i) を $\mathcal{F}_{\mathrm{VOTE}}$ から受け取り，かつ，P_i が corrupt されていないとき，S は $P_i \in L_P$ ならば (P_i,`Send`) を $\mathcal{F}_{\mathrm{AMN}}$ から H に送らせる．
2. S が (`Start`,V_j) を $\mathcal{F}_{\mathrm{VOTE}}$ から受け取り，かつ，V_j が corrupt されていないとき，S は (`Tally`,V_j) を $\mathcal{F}_{\mathrm{TALLY}}$ から H に送らせる．
3. H が $P_i(V_j)$ を corrupt すると，S は対応するダミーパーティ $\tilde{P}_i(\tilde{V}_j)$ を corrupt して，以降 H への命令に従う．

このような S は過半数の選挙サーバが corrupt されないという条件の下で，adaptive な敵 H に対してシミュレーションを完璧に行うことは明らかである． □

3.2 プロトコル π_{AMN}

ここでは，$\mathcal{F}_{\mathrm{AMN}}$ を実現するプロトコル π_{AMN} を示す．Wikström は π_{MN} を定義すると同時に，π_{MN} を実現するプロトコル (**Protocol 1**) を提案した．図 7 に Protocol 1 の概要を示す．詳細については [10] を参照のこと．

Protocol1 は $(\mathcal{F}_{\mathrm{BB}}, \mathcal{F}_{\mathrm{KG}}, \mathcal{F}_{\mathrm{ZK}}^{R_{\mathrm{C}}}, \mathcal{F}_{\mathrm{ZK}}^{R_{\mathrm{DS}}})$-hybrid モデルで $\mathcal{F}_{\mathrm{MN}}$ を実現する．$\mathcal{F}_{\mathrm{BB}}$ は掲示板の理想機能であり，そこに書き込まれた情報は誰もが読むことができるが，一度書き込まれた情報を消去もしくは変更することはできない．図 5 に $\mathcal{F}_{\mathrm{BB}}$ の定義を示す．$\mathcal{F}_{\mathrm{KG}}$ は Mix-server がメッセージをシャッフルする際に用いる公開鍵と秘密鍵を生成，公開する理想機能である．また，正しく動作しない Mix-server の代わりにメッセージの復号も行う．$\mathcal{F}_{\mathrm{ZK}}^{R}$ は関係 R に関するゼロ知識証明の理想機能であり，R_{C} は平文と暗号文の関係で，R_{C} はシャッフル前の暗号文とシャッフル後の暗号文との関係である．Protocol1 では各送信者が暗号化したメッセージを入力して，各 Mix-server がメッセージのリストを再暗号化/シャッフ

プロトコル π_{VOTE}

投票者 $P_1, P_2, \cdots, P_n$，選挙サーバ $V_1, V_2, \cdots, V_k$, と動作する．また，候補者のリスト $L_C = \{C_1, C_2, \cdots, C_{N_c}\}$ と有権者のリスト $L_P = \{P_1, P_2, \cdots, P_{N_p}\}$ を保持している．y は $\mathcal{F}_{\mathrm{TALLY}}$ の公開鍵であり公開されていると仮定する．
投票者 P_i

1. 入力 (Vote, sid, m_i) を受け取ると, $v_i = Enc_y(m_i)$ を計算して，(Send, v_i) を $\mathcal{F}_{\mathrm{AMN}}$ へ送る．
2. 入力 (Read-result) を受け取ると，(Read) を $\mathcal{F}_{\mathrm{TALLY}}$ へ送る．
3. $\mathcal{F}_{\mathrm{TALLY}}$ から (Result-tally, L_{tally}) を受け取ると，(Result, L') を出力する．

選挙サーバ V_j (Start) の入力を受けると次の動作を行う．

1. (Run) を $\mathcal{F}_{\mathrm{AMN}}$ へ渡す．
2. $\mathcal{F}_{\mathrm{AMN}}$ から (List,L') を受け取ると (Tally, L') を $\mathcal{F}_{\mathrm{TALLY}}$ へ送る．

図 4: プロトコル π_{VOTE}

理想機能 $\mathcal{F}_{\mathrm{BB}}$

パーティ $P_1, P_2, \cdots, P_k$ と敵 S と動作する．

1. $c = 0$
2. (Write,m_i) ($m_i \in \{0,1\}^*$) を P_i から受け取ると，(P_i, m_i) をタグ c で保存し (Write,c, P_i, m_i) を S へ送る．
3. (Read, c) を P_j から受け取ると，(P_i, m_i) がタグ c で保存されいないかチェックする．あれば，(P_j, Read,c, P_i, m_i) を S へ送り，(Read,c, P_i, m_i) を P_j へ送る．保存されていなければ，(P_j, NoRead,c) を S へ送り，(NoRead,c) を P_j へ送る．

図 5: 掲示板の理想機能 $\mathcal{F}_{\mathrm{BB}}$

ルして次の server へ渡す，という操作を繰り返す．最後の Mix-server が再暗号化するとメッセージが完全に復号され平文となる．しかし，中間の各 Mix-server で順番が入れ替えられているので，どれがどの送信者からのメッセージなのかわからない．

π_{AMN} は Protocol1 を一部変更して記述する．送信者認証のために Canetti の定義した理想機能 $\mathcal{F}_{\mathrm{CERT}}$ を用いる [2]．$\mathcal{F}_{\mathrm{CERT}}$ は署名の検証ならびに署名者の認証を行う．図 6 に $\mathcal{F}_{\mathrm{CERT}}$ の定義を示す．$(\mathcal{F}_{\mathrm{BB}}, \mathcal{F}_{\mathrm{CERT}})$-hybrid モデルで $\mathcal{F}_{\mathrm{AMN}}$ を実現するプロトコル π_{AMN} を図 8 に示す．仮定として，有権者のリストが $\mathcal{F}_{\mathrm{BB}}$ 上に存在する．

π_{AMN} は A および過半数の Mix-server が honest であるという条件のもとで $\mathcal{F}_{\mathrm{AMN}}$ を実現する．$\mathcal{F}_{\mathrm{AMN}}$ と $\mathcal{F}_{\mathrm{MN}}$ の違いは，投票者の認証部分のみであり，そこを実現している検証者 A の動作の正しさは，全てのパーティが検証できる．よって A が corrupt されない限り π_{AMN} が $\mathcal{F}_{\mathrm{AMN}}$ を実現できるのは明らかである．

3.3 π_{TALLY}

過半数の集計サーバが corrupt されないという仮定の下で $\mathcal{F}_{\mathrm{TALLY}}$ を実現するプロトコルは，一般の (k,t)-threshold 暗号プロトコルと $\mathcal{F}_{\mathrm{CP}}$ を用いることで実現できる [3][5][4]．証明は省略する．$\mathcal{F}_{\mathrm{CP}}$ は 2 パーティ間でのコミットメントとその証明を行う理想機能である．$\mathcal{F}_{\mathrm{CP}}$ を用いて正しく集計したことを証明する．(k,t)-threshold 暗号プロトコルは，公開鍵で暗号化されたメッセージを k 人のうち t 人のパーティが集まれば復号できる．t 人未満のパーティが集まっても何の情報も得られない．過半数の集計サーバは honest であるので，$t = \frac{k}{2}$ と置くことで容易に実現が可能である．

4 おわりに

本稿では，汎用的結合可能性を満足する電子投票の理想機能を定義し，投票者認証機能を持つ Mix-Net と Mix-Net の出力を集計する機能を用いて実現するプロトコルを提案した．提案したプロトコルはある程度の長さのメッセージを投票内容とすることができ，一般的な投票に用いることができる．また，汎用的結合可能性を用いていくつかの理想機能を組み合わせて実現しているのでシステム全体の構成が単純で理解しやすくなっている．今後は簡潔さを失うことなく効率的なプロトコルを提案

理想機能 $\mathcal{F}_{\mathrm{CERT}}$

パーティ P, S と動作する．

署名生成 (`Sign`, sid, m) を S から受け取ると，$sid = (S, sid')$ かどうか検証する．違うならば無視する．そうであれば，(`Sign`, sid, m) を敵に送る．敵から (`Signature`, sid,m, σ) を受け取ると，$(m, \sigma, 0)$ が記録されていないか検証する．されていればエラーメッセージを S へ送り停止する．記録されていなければ (`Signature`,sid,m, σ) を S に出力して $(m, \sigma, 1)$ を記録する．

署名検証 (`Verify`, sid,m, σ) を P から受け取ると，(`Verify`,sid,m, σ) を敵へ送る．(`Verified`,sid,m, ϕ) を敵から受け取ると次のいずれかの操作を行い (`Verified`,sid,m, f) を P へ出力する．

1. $(m, \sigma, 1)$ が記録されているならば $f = 1$.
2. それ以外で，署名者が corrupt されていない，かつ，$(m, \sigma', 1)$ が任意の σ' に対しても記録されていなければ，$f = 0$ として $(m, \sigma, 0)$ を記録する．
3. それ以外で，(m, σ, f') が記録されていなければ，$f = f'$ とする．
4. それ以外であれば，$f = \phi$ として，(m, σ', ϕ) を記録する．

図 6: 署名と署名者認証の理想機能 $\mathcal{F}_{\mathrm{CERT}}$

を検討している．

参考文献

[1] Ran Canetti, ‘Universally Composable Security: A new Paradigm for Cryptgraphic Protocols‘, ’http://eprint.jacr.org/2000/067 and ECCC TR 01-24’

[2] Ran Canetti, ‘Universally Composable Signature, Certification, and Authentication‘, CSFW 2004, 219-, 2004

[3] Ran Canetti, Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, Tal Rabin, ‘Adaptive Security for Threshold Cryptosystems.‘, CRYPTO’99, LNCS, 1999.

[4] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, Amit Sahai, ‘Universally Composable Two-Party and Multi-party Secure Computation‘, STOC 2002, 494-503, 2002

[5] Yvo Desmedt, Yair Frqaankel, ‘Threshold cryptsystems‘, CRYPT ’89, LNCS 435, pp.307-315, 1990

[6] Yvo Desmedt, Kaoru Kurosawa, ‘How to Break a Practical MIX and Design a New One‘, EUROCRYPT 2000, LNCS 1807, pp.557-572, 2000

[7] Jun Furukawa, Kazue Sako, ‘An efficient scheme for proving shuffle‘, Crypto 2001, LNCS 2139, pp. 368-387, 2001

[8] Jens Groth, ‘Evaluating Security of Voting Schemes in the Universal Composability Framework‘, ACNS 2004

[9] Kazue Sako, Joe Killian, ‘Receipt-Free Mix-Type Voting Scheme -A practical solution to the implementation of a voting booth-‘, LNCS 921, pp. 393-403, 1995

[10] Douglas Wikström, ‘A Universally Composable Mix-Net‘, LNCS 2951, pp. 317-335. 2004

Protocol 1(概要)

送信者 $P_1, P_2, \cdots, P_n$，Mix-xserver $M_1, M_2, \cdots, M_k$ と動作する．
送信者 P_i

1. $\mathcal{F}_{\mathrm{KG}}$ から各 M_i の公開鍵を得て，システムの公開鍵を計算して投票内容を暗号化し票を生成，$\mathcal{F}_{\mathrm{BB}}$ に書き込む．
2. 暗号化の際に用いた乱数を $\mathcal{F}_{\mathrm{ZK}}^{R_{\mathrm{C}}}$ へ渡す．

Mix-server M_j

1. $\mathcal{F}_{\mathrm{KG}}$ から自分の秘密鍵と各 Mix-server の公開鍵を得る．
2. 過半数以上の Mix-server が (`Run`) を $\mathcal{F}_{\mathrm{BB}}$ へ書き込むと $\mathcal{F}_{\mathrm{BB}}$ 上の投票に対して，正当な票かどうか $\mathcal{F}_{\mathrm{ZK}}^{R_{\mathrm{C}}}$ を用いて検証する．
3. 自分の前の Mix-server がリストを $\mathcal{F}_{\mathrm{BB}}$ へ書き込むと，そのリストに対し，自分の秘密鍵を用いて再暗号化して $\mathcal{F}_{\mathrm{BB}}$ へ書き込む．暗号化の際に用いた秘密情報を $\mathcal{F}_{\mathrm{ZK}}^{R_{\mathrm{DS}}}$ へ送る．
4. 他の Mix-server が，シャッフルしたリストを $\mathcal{F}_{\mathrm{BB}}$ へ書き込んだら，$\mathcal{F}_{\mathrm{ZK}}^{R_{\mathrm{DS}}}$ を用いて正しくシャッフルされたものか検証する．検証が正しく行われない場合，$\mathcal{F}_{\mathrm{KG}}$ に復号してもらう．
5. 最後の Mix-server が $\mathcal{F}_{\mathrm{BB}}$ に書いたリストを辞書順に並べて出力する．

図 7: Protocol 1 の概要

プロトコル π_{AMN}

送り手 $P_1, P_2, \cdots, P_n$, Mix-server $M_1, M_2, \cdots, M_k$, 検証者 A と動作する．
送り手 P_i

1. 入力 (`Send`,m_i) を受け取ると，Protocol1 に従って暗号化メッセージ (u_i, v_i) を作成する．
2. (`Sign`,$(P_i,\ sid)$,(u_i, v_i)) を $\mathcal{F}_{\mathrm{CERT}}$ へ送る．
3. (`Signature`,$(P_i$,$sid)$,(u_i, v_i),σ_i) を $\mathcal{F}_{\mathrm{CERT}}$ から受け取ると，(`Write`,$((u_i, v_i)$,sid,$\sigma_i)$) を $\mathcal{F}_{\mathrm{BB}}$ へ送る．

検証者 A

1. $((u_i, v_i)$,sid,$\sigma_i))$ が $\mathcal{F}_{\mathrm{BB}}$ 上に現れると，(`Verify`,sid,$(u_i, v_i), \sigma_i$) を $\mathcal{F}_{\mathrm{CERT}}$ へ送る．
2. (`Verified`, $(u_i, v_i), f_i$) を $\mathcal{F}_{\mathrm{CERT}}$ から受け取ると，
   - $f_i = 1$，かつ，P_i が $\mathcal{F}_{\mathrm{BB}}$ 上の有権者リストに含まれているならば (`Write`,`Authenticated`,$P_i, (u_i, v_i)$) を $\mathcal{F}_{\mathrm{BB}}$ へ送る．

以降，(`Authenticated`,$P_i, (u_i, v_i)$) と $\mathcal{F}_{\mathrm{BB}}$ 上に現れたメッセージを新たに (u_i, v_i) として，Wikström の提案した Protocol1 を実行する．

図 8: プロトコル π_{AMN}