Vol.2023-DPS-194 No.66
Vol.2023-CSEC-100 No.66
2023/3/7

トークン型電子現金方式の二重使用検知およびプライバシーに関する形式検証の考察

奥田 哲矢[1,a)]　荒井 研一[2]　齋藤 恆和[1]　千田 浩司[3]　中林 美郷[1]　山村 和輝[1]　宮澤 俊之[1]
阿部 正幸[1]

概要：本研究は，先行研究「トークン型電子現金方式の形式検証手法に関する初期検討」（第 99 回 CSEC 研究会）の続編である．近年，中央銀行デジタル通貨への期待が集まる中で，トークン型電子現金方式は，災害時や通信障害時のオフライン利用が可能となり得る点で注目されている．ただし，既知の事実としてトークン型電子現金方式では通貨が発行銀行に還収されるまで二重使用の検知が（一義的には）出来ないことが知られている．本論文では，まず，今回対象プロトコルについて，二重使用対策の仕組みを組み込む対象とすべきプロトコルのフェーズを考察するため，先行研究で定式化した通貨発行／引出／支払フェーズに加えて，両替／与信などの一時的なオンライン化のフェーズや，市中銀行への預入および発行銀行への還収のフェーズについて定式化を行った．さらに，先行研究で述べた通貨発行／引出／支払フェーズを含む，すべてのフェーズについて代表的な形式検証ツールである ProVerif による形式的な実装を行い，到達可能性 (reachability) を確認した．その上で，クライアント端末上のセキュアハードウェアによる二重使用対策について ProVerif による形式的な実装を行い，端末上の支払処理が正常終了した時に端末上では二重使用が発生しないことを確認した．さらに，二重使用検知などのコンプライアンスを重視した際にトレードオフとなるプライバシー要件として，仮名性（特定／追跡不可能性）について，ProVerif で観測等価性を表現する choice 文で記述する場合の方針について考察を行い，今回対象プロトコルにおける仮名性の検証結果が True（仮名／識別不可能）となることを確認した．

Formal verification of double spend identification and privacy for transferable electronic cash system

1. はじめに

近年, 中央銀行デジタル通貨への期待が集まる中で，トークン型電子現金方式は，災害時や通信障害時のオフライン利用が可能となり得る点で注目されている．ただし，既知の事実として，トークン型電子現金方式では通貨が発行銀行に還収されるまで二重使用の検知が（一義的には）出来ないことが知られている．本論文では，まず，今回対象プロトコルについて，二重使用対策の仕組みを組み込む対象とすべきプロトコルのフェーズを考察するため，先行研究で定式化した通貨発行／引出／支払フェーズに加えて，両替／与信などの一時的なオンライン化のフェーズや，市中銀行への預入および発行銀行への還収のフェーズについて，定式化を行った．さらに，先行研究で述べた通貨発行／引出／支払フェーズを含む，すべてのフェーズについて代表的な形式検証ツールである ProVerif による形式的な実装を行い，到達可能性を確認した．その上で，クライアント端末上のセキュアハードウェアによる二重使用対策について ProVerif による形式的な実装を行い，端末上の支払処理の正常終了時に端末上では二重使用が発生しないことを確認した．さらに，二重使用検知などのコンプライアンスを重視した際にトレードオフとなるプライバシー要件として，仮名性（特定／追跡不可能性）について ProVerif で観測等価性を表現する choice 文で記述する場合の方針について考察を行い，今回対象プロトコルにおける仮名性の検証結果が True（仮名／識別不可能）となることを確認した．

[1] NTT 社会情報研究所
180-8585 東京都武蔵野市緑町 3-9-11
[2] 長崎大学
[3] 群馬大学
[a)] tetsuya.okuda.uy@hco.ntt.co.jp

2. 先行研究

2.1 電子現金プロトコル

電子現金 (e-cash) プロトコルの研究は, David Chaum [5] に遡る．口座型の電子現金とは異なるトークン型の電子現金プロトコルであり，ブラインド署名により，ユーザは発行銀行に身元を明かさずに通貨引き出しが可能であり，“untraceable” すなわち発行銀行がユーザを特定／追跡できず，かつ，署名および公開鍵が特定ユーザに紐付けられないため，通貨の支払いを受けた店舗なども署名および公開鍵から個人を辿ることはできないことを特徴とする．

David Chaum の最初の提案は，オンライン型と呼ばれており，通貨を銀行に問い合わせて検証 (“与信”) されるまで支払いは受理されない．本特徴から，銀行側は事前に預金／還流した通貨でないかを確認することで，二重使用を防ぐことが可能である一方，“non-transferable” すなわち支払いで受け取ったコインを店舗はそのままでは再利用は出来ず，一度は銀行に問い合わせる必要が有る．実際には小切手に近い利用イメージの提案であったと言える．

その後，トークン型電子現金プロトコルを活かすために有効な特徴である “transferable” な電子現金プロトコルの提案が多くなされてきた [1], [4], [6], [9]．これらはオフライン型と呼ばれており，支払で受け取った通貨を銀行に問い合わせて検証 (“与信”) を行う必要は無い．そのため，受け取った通貨をそのまま他の取引の支払いに利用できる転々流通可能 “transferable” という特徴を有しており，災害時や通信障害時に利用可能であるとして，近年は注目されている．一方，二重使用を一義的には防げない点は課題として認識されている．また，プライバシーに関する機能についてもそれぞれ特徴があり，通貨の二重使用をしなければ匿名性は保たれるなど，プライバシーとコンプライアンスのバランスについても，近年は注目されている [7], [14].

2.2 電子現金と形式検証

電子現金プロトコルの安全性モデルと形式検証に関する先行研究 [8] では，電子現金プロトコルの安全性モデルとして，3 点の Security Properties と 2 点の Privacy Properties を定義している．さらに，左記の安全性モデルに従って，[5] のオンライン型プロトコルや，[6] のオフライン型プロトコルについて，セキュリティプロトコルの代表的な形式検証ツールである ProVerif [3] による形式的な実装を行い，各安全性を検証して提案手法を評価している．また，電子現金に限らず，セキュリティプロトコルの形式検証の研究としては，近年はプライバシー要件の形式検証についても注目されている [2].

3. 安全性モデル

本研究では，電子現金プロトコルの安全性モデルとして，先行研究 [7], [8] を参考として，以下の Security Properties, Financial Properties，Privacy Properties を想定する．

- Security Properties
  - Secrecy（秘匿性）
  - Authentication（認証性）
  - Unforgeability（偽造不可能性）
  - Double Spend Identication（二重使用特定性）
  - Exclupability（免責性）
  - Anti-Money Laundering（ロンダリング耐性）
- Financial Properties
  - Fairness（取引公平性）
  - Refreshability（通貨／鍵更新可能性）
  - Monetary Stability（通貨価値安定性）
- Privacy Properties
  - Pseudonymity（仮名性）
    すなわち Unidentifiability(特定不可能性)
    すなわち Untraceability(追跡不可能性)
  - Anonymity（匿名性）
    すなわち Unlinkability（結合不可能性）

Security Properties について，詳細は [11] で説明済みであるが，一部，より直感的に二重使用対策を論じるために，本稿では，二重使用検知のイベントを判定対象通貨と使用済み通貨 DB の一致により定義して，当該イベントへの到達可能性 (reachability) により，二重使用の存在有無を判定することとした．本稿の還収フェーズにおける二重使用検知については，本方針で以下の判定クエリを記述している．

$$
\begin{aligned}
&\mathrm{event}(\mathrm{currency\_collect})(B_0, (T[n], \ldots T[0])) \\
&\forall k(B_0(\mathit{collected}), T[n], \ldots, T[k], \ldots, T[0]) : \\
&\quad \forall i(B_0(\mathit{spended}), T[m], \ldots, T[i], \ldots, T[0]) : \\
&\qquad \mathit{if}\ T[k] == T[i]\ \mathit{then} \\
&\qquad \mathrm{event}(\mathrm{double\_spend\_detected})(B_0, T[k])
\end{aligned}
$$

Double Spend Identication は，上記の二重使用検知の後，二重使用を行ったユーザを特定可能とすること，仮名性（や匿名性）を剥奪して特定／追跡可能とすることである．Exclupability（免責性）は，被害者となるユーザが二重使用したように攻撃者が偽装することで，被害者の仮名性（や匿名性）を誤って剥奪して特定／追跡可能されないことであり，プライバシー脅威への対策とも言える．

Privacy Properties について, 本研究では, Pseudonymity（仮名性），すなわち Unidentifiability（特定不可能性）あるいは Untraceability(追跡不可能性) と，Anonymity（匿名性）すなわち Unlinkability（結合不可能性）を挙げている．

前者の仮名性とは，本論文では，仮名（公開鍵）から実名を辿れないこと／実名を特定できないことと定義する．また，後者の匿名性との差異として，同じユーザは複数取引間で同じ仮名（公開鍵）を使用することとする．これは

翻って，仮名により同じユーザを複数取引間で関連付けが可能（結合可能）であることを意味する．一方，後者の匿名性とは，本論文では，同じユーザを複数取引間で関連付けが不可能（結合不可能）であることと定義する．今回の検証対象プロトコルにおいては，[5], [6] のようなブラインド署名による匿名性を提供せず，通貨署名に用いる公開鍵による仮名性のみを提供する方針としている．本選択の理由としては，効率の観点として，[5], [6] の方式では，通貨発行／引出時のブラインド署名の blind 処理，通貨預金／還収時の unblind 処理の発行銀行サーバの処理負荷が大きくなること，後述する公開鍵を取引毎に一回限りの使用とする運用的な対応方法では，認証局（市中銀行）による都度の証明書発行の処理負荷が大きくなること，コンプライアンスの観点として，二重使用者の特定に公開鍵（仮名）が有効であること，が挙げられる．また，仮名性についてはエンティティ毎に充足されるべき要件は異なる．例えば，ユーザのメインバンクを担当する市中銀行に対してはユーザの個人情報を預託する想定であり，個人情報と公開鍵との対応表を有するために，市中銀行との通貨引出プロトコルにおいては仮名性は充足しない一方，他ユーザ／店舗との通貨支払プロトコルにおいては仮名性が充足される．

プライバシー要件の ProVerif による記述方針について，先行研究 [8] においては，観測等価性 (observational equivalence) により，追跡不可能性 (Untraceability) および結合不可能性 (Unlinkability) の判定が可能としている．本稿では，仮名性すなわち追跡不可能性について，下記のように定式化を行う．「受信者 U_k が送信者 U_j より通貨トークン $(T[n], T[n-1], T[n-2], \ldots, T[0])$ で支払を受領した」イベントを $\mathrm{spend}(U_j, U_k, (T[n], T[n-1], T[n-2], \ldots, T[0]))$，「受信者 U_k が送信者 U_h より通貨トークン $(T[n]', T[n-1]', T[n-2], \ldots, T[0])$ で支払を受領した」イベントを $\mathrm{spend}(U_h, U_k, (T[n]', T[n-1]', T[n-2], \ldots, T[0]))$ として，異なる送信者からの通貨の受取を表現すると，

$$\mathrm{event}(\mathrm{spend}(U_j, U_k, (T[n], T[n-1], T[n-2] \ldots T[0])) \;\&\&$$
$$\mathrm{event}(\mathrm{spend}(U_h, U_k, (T[n]', T[n-1]', T[n-2] \ldots, T[0]))$$
$$\rightarrow \mathit{observational\ equivalent\ for}\ U_k$$
$$((T[n], T[n-1]), (T[n]', T[n-1]'))$$

なお，上記定式化においては，$n \geqq 2$ として，送信者 U_j に関連する情報については，$T[n]$ に署名 $S[n]$ およびハッシュ値 $H(T[n-1])$ が，$T[n-1]$ に公開鍵 pk_U およびノンス値 $nonce_U$ が含まれるが，送信者 U_j, U_h に通貨が届く以前の $T[n-2] \ldots, T[0]$ の通貨使用履歴からの特定／追跡や通貨使用履歴間の結合については想定しない設定とした．

4. 検証対象プロトコル

今回の検証対象プロトコルとして，匿名公開鍵によるデジタル署名を用いた単純な（固定金額）トークン型電子現金を転々流通可能となるよう署名連鎖によって自然に拡張したサンプル方式を下記に示す．なお，当該プロトコルを構成するフェーズの内，(0) 通貨発行鍵配布，(1) ルート認証局および中間認証局の証明書配布，(2) 本人確認（利用者），(2)' 機関確認（銀行），(3) 通貨発行，(4) 通貨引出，(5) 通貨支払については，先行研究 [11] を参照されたい．

(6) 両替

本プロトコルにおいて，両替フェーズは，利用者から市中銀行への支払処理（預入処理）と市中銀行から利用者への支払処理（引出処理）のセットで表現される．利用者 U_k は両替したい通貨の額面 v と自身の公開鍵 pk_{U_j} を，市中銀行 B_i に送付する．市中銀行 B_i は利用者 U_k の公開鍵証明書を検証して，nonce 値と自身の公開鍵 pk_{B_i} を，利用者 U_k に送付する．利用者 U_k は市中銀行 B_i の公開鍵証明書を検証して，自身の未使用通貨 $(T[n-1], \ldots, T[0])$ を選択して，$T[n]$ を作成し，$(T[n], \ldots, T[0])$ を市中銀行 B_i に送付する．市中銀行 B_i は受け取った両替通貨の署名連鎖 $(T[n], \ldots, T[0])$ の署名検証を行い，署名検証の後，市中銀行 B_i は使用済み通貨 DB_i に両替通貨 $(T[n], \ldots, T[0])$ を格納するとともに，署名検証の通知を受けた利用者 U_k は両替通貨 $(T[n-1], \ldots, T[0])$ をユーザ端末から削除する．

より詳細には，(6-0-1) で利用者 U_k は両替額（両替通貨内訳 v 円）を確定，(6-0-2) で利用者 U_k は市中銀行 B_i に，利用者 U_k の公開鍵証明書を送付，(6-1-1) で市中銀行 B_i は pk_{A_0} 自己署名証明書を検証，(6-1-2) で市中銀行 B_i は中間認証局 A_1 の公開鍵証明書を検証，(6-1-3) で市中銀行 B_i は利用者 U_k の公開鍵証明書を検証する．(6-2-1) で市中銀行 B_i は $nonce$（市中銀行）を生成，(6-2-2) で市中銀行 B_i は利用者 U_k に市中銀行 B_i の公開鍵証明書，$nonce$（市中銀行）を送付，(6-3-1) で利用者 U_k は自己署名証明書を検証，(6-3-2) で利用者 U_k は中間認証局 A_1 の公開鍵証明書を検証，(6-3-3) で利用者 U_k は市中銀行 B_i の公開鍵証明書を検証，(6-4-1) で利用者 U_k は市中銀行 B_i の公開鍵と $nonce$（市中銀行）と $T[n-1]$ のハッシュ値のセットに対して署名 S_n を生成，(6-4-2) で利用者 U_k は署名 $T[n]$ の生成，両替通貨の署名連鎖 $(T[n-1], \ldots, T[0])$ に $T[n]$ を追加，(6-4-3) で利用者 U_k は市中銀行 B_i に両替通貨 $(T[n], \ldots, T[0])$ を送付，(6-5-0)(6-5-1)(6-5-2) で市中銀行 B_i は受け取った通貨一式の署名検証，(6-5-3)(6-5-4) で市中銀行 B_i は通貨 $T[1]$ の署名検証，(6-5-5)(6-5-6) で市中銀行 B_i は通貨 $T[0]$ の署名検証を行う．(6-5-7)(6-5-8) で市中銀行 B_i は利用者 U_k に通貨一式の署名検証結果を送付，(6-5-9) で利用者 U_k は両替に使用した通貨 $(T[n-1], \ldots, T[0])$ を端末より削除，(6-5-10) で市中銀行 B_i は両替通貨 $(T[n], \ldots, T[0])$ を使用済み通貨 DB_i に格納する．

(6-0-1) $U_k : v$

(6-0-2) $U_k \rightarrow B_i : (cert_{pk_{U_k}})$

(6-1-1) B_i : $(pk_{A_0}, sign_{pk_{A_0}}) = cert_{pk_{A_0}}$ if Verif $(pk_{A_0}, sign_{pk_{A_0}})_{pk_{A_0}}$ = True then

(6-1-2) B_i : $(pk_{A_1}, sign_{pk_{A_1}}) = cert_{pk_{A_1}}$ if Verif $(pk_{A_1}, sign_{pk_{A_1}})_{pk_{A_0}}$ = True then

(6-1-3) B_i : $(pk_{U_k}, sign_{pk_{U_k}}) = cert_{pk_{U_k}}$ if Verif $(pk_{U_k}, sign_{pk_{U_k}})_{pk_{A_1}}$ = True then

(6-2-1) B_i : $nonce_{B_i}$

(6-2-2) $B_i \rightarrow U_k$: $(cert_{pk_{B_i}}, v, nonce_{B_i})$

(6-3-1) U_k : $(pk_{A_0}, sign_{pk_{A_0}}) = cert_{pk_{A_0}}$ if Verif $(pk_{A_0}, sign_{pk_{A_0}})_{pk_{A_0}}$ = True then

(6-3-2) U_k : $cert_{pk_{A_1}} = (pk_{A_1}, sign_{pk_{A_1}})$ if Verif $(pk_{A_1}, sign_{pk_{A_1}})_{pk_{A_0}}$ = True then

(6-3-3) U_k : $(pk_{B_i}, sign_{pk_{B_i}}) = cert_{pk_{B_i}}$ if Verif $(pk_{B_i}, sign_{pk_{B_i}})_{pk_{A_0}}$ = True then

(6-4-0) U_k : Select$(T[n-1], \ldots, T[0])$

(6-4-1) U_k : S_n = Sign$((pk_{B_i}, nonce_{B_i}, H(T[n-1])), sk_{U_k})$

(6-4-2) U_k : $T[n] = ((pk_{B_i}, nonce_{B_i}, H(T[n-1])), S_n)$

(6-4-3) $U_k \rightarrow B_i$: $(T[n] \ldots, T[0])$

(6-5-0) B_i : for i = $[n, \ldots, 1]$

(6-5-1) B_i : $(pk_{U_i}, nonce_{U_i}, H(T[i-1], \ldots, T[0]), S_i) = T[i]$

(6-5-2) B_i : if Verif $((S_i, (pk_{U_i}, nonce_{U_i}, H(T[i-1], \ldots, T[0])))_{pk_{U_i}}$ = True then

(6-5-3) B_i : $(pk_{B_i}, nonce_{B_i}, H(T[0]), S_1) = T[1]$

(6-5-4) B_i : if Verif $((S_1, (pk_{B_i}, nonce_{B_i}, H(T[0])))_{pk_{B_i}}$ = True then

(6-5-5) B_i : $(v, y, inf_{B_i}, pk_{B_i}, nonce_{B_i}, S_0) = T[0]$

(6-5-6) B_i : if Verif $((v, y, inf_{B_i}, pk_{B_i}, nonce_{B_i}), S_0)_{pk_{B_0,v,y}}$ = True then

(6-5-7) $B_i \rightarrow U_k$: signature_true

(6-5-8) U_k : if signature_true = true then

(6-5-9) U_k : Delete $(T[n-1], \ldots, T[0])$

(6-5-10) B_i : Insert $(T[n], \ldots, T[0])$

その後，利用者 U_k は両替通貨内訳 v と自身の公開鍵証明書および nonce 値を市中銀行 B_i に送付．市中銀行 B_i は利用者 U_k の公開鍵証明書を検証して，未使用通貨 DB_i の v に一致する通貨 $T[0]$ を選択，U_k の公開鍵を保有者情報として $T[1]$ を作成し，$T[0]$ を未使用通貨 DB_i から削除．利用者 U_k は市中銀行 B_i の公開鍵証明書と通貨 $(T[1], T[0])$ を検証して，通貨 $(T[1], T[0])$ をユーザ端末に格納する．

より詳細には，(6-6-0) で利用者 U_k は両替額（両替通貨内訳 v 円）を確定，(6-6-1) で利用者 U_k は *nonce*（利用者）を生成，(6-6-2) で利用者 U_k は市中銀行 B_i に利用者 U_k の公開鍵証明書，両替通貨内訳 v，*nonce*（利用者）を送付，(6-7-1)(6-7-2) で市中銀行 B_i は未使用通貨 DB_i から引出通貨 T[0] を選択，(6-7-3)(6-7-4) で市中銀行 B_i は利用者公開鍵と *nonce*（利用者）に対して署名 (T[1]) を生成し，引出通貨に追加，(6-7-5) で市中銀行 B_i は引出に使用した通貨 $T[0]$ を未使用通貨 DB_i から削除，(6-8-0) で市中銀行 B_i は利用者 U_k に市中銀行 B_i の公開鍵証明書，引出通貨 $(T[1], T[0])$ を送付，(6-9-1)(6-9-2) で利用者 U_k は市中銀行 B_i の公開鍵証明書を用いて通貨 $T[1]$ の通貨署名 S_1 を検証，(6-9-3)(6-9-4) で利用者 U_k は発行銀行 B_0 の通貨発行公開鍵の公開鍵証明書を用いて通貨 $T[0]$ の通貨署名 S_0 を検証，(6-10-0) で利用者 U_k は引出通貨 $(T[1], T[0])$ をユーザ端末に保存，(6-11-0) で市中銀行 B_i は未使用通貨 DB_i 内の引出に使用した通貨 $(T[0])$ を削除する．

(6-6-0) U_k : v'

(6-6-1) U_k : $nonce_{U_k}$

(6-6-2) $U_k \rightarrow B_i$: $(cert_{pk_{U_k}}, v, nonce_{U_k})$

(6-7-1) B_i : Select$(num, (v, y, inf_{B_i}, pk_{B_i}, nonce_{B_i}, S_0))$

(6-7-2) B_i : $T[0] = (v, y, inf_{B_i}, pk_{B_i}, nonce_{B_i}, S_0)$

(6-7-3) B_i : S_1 = Sign$((pk_{U_k}, nonce_{U_k}, H(T[0])), sk_{B_i})$

(6-7-4) B_i : $T[1] = ((pk_{U_k}, nonce_{U_k}, H(T[0])), S_1)$

(6-7-5) B_i : Delete $(T[0])$

(6-8-0) $B_i \rightarrow U_k$: $(cert_{pk_{B_i}}, (T[1], T[0]))$

(6-9-1) U_k : $((pk_{U_k}, nonce_{U_k}, H(T[0])), S_1) = T[1]$

(6-9-2) if Verif $((pk_{U_k}, nonce_{U_k}, H(T[0])), S_1)_{pk_{B_i}}$=True then

(6-9-3) U_k : $((v, y, inf_{B_i}, pk_{B_i}, nonce_{B_i}), S_0) = T[0]$

(6-9-4) U_k : if Verif $((v, y, inf_{B_i}, pk_{B_i}, nonce_{B_i}), S_0)_{pk_{B_0,v,y}}$ = True then

(6-10-0) U_k : Insert$(T[1], T[0])$

(6-11-0) B_i : Delete$(T[0])$

(7) 与信

利用者が通貨の利用可否を調べる際には通貨を市中銀行に送付する．市中銀行は使用中通貨 DB に対象通貨を含むレコードがあれば利用可と応答する．本フェーズの詳細は次回作で述べたい．

(8) 預入

本プロトコルにおいて，預入処理は，利用者から市中銀行への支払処理に相当として表現される．利用者 U_k は預入したい通貨の額面 v と自身の公開鍵 pk_{U_k} を，市中銀行 B_i に送付する．市中銀行 B_i は利用者 U_k の公開鍵証明書を検証して，市中銀行 B_i の nonce 値と公開鍵証明書を利用者 U_k に送付する．利用者 U_k は市中銀行 B_i の公開鍵証明書を検証して，未使用通貨 $(T[n-1], ..., T[0])$ を選択，通貨 $T[n]$ を作成して通貨 $(T[n], ..., T[0])$ を市中銀行 B_i に送付．市中銀行 B_i は受け取った預入通貨の署名連鎖 $(T[n], \ldots, T[0])$ の署名検証，署名検証の後，市中銀行 B_i は使用済み通貨 DB_i に預入通貨 $(T[n], \ldots, T[0])$ を格納するとともに，署名検証の通知を受け取った利用者 U_k は預入した通貨 $(T[n-1], \ldots, T[0])$ をユーザ端末から削除．

より詳細には，(8-0-1) で利用者 U_k は預入額（預入通貨内訳）を確定，(8-1-1) で利用者 U_k は市中銀行 B_i に利用者 U_k の公開鍵証明書，預入額を送付，(8-1-2) で市中銀行 B_i は認証局 A_0 の pk_{A_0} の自己署名証明書を検証，(8-1-3) で市中銀行 B_i は中間認証局 A_1 の pk_{A_1} の公開鍵証明書を検証，(8-1-4) で市中銀行 B_i は利用者 U_k の pk_{U_k} の公開鍵証明書を検証，(8-2-1) で市中銀行 B_i は $nonce$（市中銀行）を生成，(8-2-2) で市中銀行 B_i は利用者 U_k に市中銀行 B_i の pk_{B_i} の公開鍵証明書，$nonce$（市中銀行）を送付，(8-2-3) で利用者 U_k は認証局 A_0 の pk_{A_0} の自己署名証明書を検証，(8-2-4) で利用者 U_k は市中銀行 B_i の pk_{B_i} の公開鍵証明書を検証，(8-3-1) で利用者 U_k は市中銀行 B_i の公開鍵と $nonce$（市中銀行）と $T[n-1]$ のハッシュ値のセットに対して署名 S_n を生成，(8-3-2) で利用者 U_k は署名 $T[n]$ の生成，預入通貨の署名連鎖 $(T[n-1],\ldots,T[0])$ に $T[n]$ を追加，(8-3-3) で利用者 U_k は市中銀行 B_i に預入通貨 $(T[n],\ldots,T[0])$ を送付，(8-4-0)(8-4-1)(8-4-2) で市中銀行 B_i は受け取った通貨一式の署名検証，(8-4-3)(8-4-4) で市中銀行 B_i は通貨 $T[1]$ の署名検証，(8-4-5)(8-4-6) で市中銀行 B_i は通貨 $T[0]$ の署名検証を行う．(8-5-1)(8-5-2) で市中銀行 B_i は利用者 U_k に通貨一式の署名検証結果を送付，(8-5-3) で市中銀行 B_i は預入通貨 $(T[n],\ldots,T[0])$ を使用済み通貨 DB_i に格納，(8-5-4) で利用者 U_k は預入した通貨 $(T[n-1],\ldots,T[0])$ を端末より削除する．

(8-0-1) U_k : v

(8-1-1) $U_k \rightarrow B_i$: $(cert_{pk_{U_k}}, v)$

(8-1-2) B_i : $(pk_{A_0}, sign_{pk_{A_0}}) = cert_{pk_{A_0}}$ if Verif $(pk_{A_0}, sign_{pk_{A_0}})_{pk_{A_0}}$ = True then

(8-1-3) B_i : $(pk_{A_1}, sign_{pk_{A_1}}) = cert_{pk_{A_1}}$ if Verif $(pk_{A_1}, sign_{pk_{A_1}})_{pk_{A_0}}$ = True then

(8-1-4) B_i : $(pk_{U_k}, sign_{pk_{U_k}}) = cert_{pk_{U_k}}$ if Verif $(pk_{U_k}, sign_{pk_{U_k}})_{pk_{A_1}}$ = True then

(8-2-1) B_i : $nonce_{B_i}$

(8-2-2) $B_i \rightarrow U_k$: $(cert_{pk_{B_i}}, nonce_{B_i})$

(8-2-3) U_k : $(pk_{A_0}, sign_{pk_{A_0}}) = cert_{pk_{A_0}}$ if Verif $(pk_{A_0}, sign_{pk_{A_0}})_{pk_{A_0}}$ = True then

(8-2-4) U_k : $(pk_{B_i}, sign_{pk_{B_i}}) = cert_{pk_{B_i}}$ if Verif $(pk_{B_i}, sign_{pk_{B_i}})_{pk_{A_0}}$ = True then

(8-3-0) U_k : Select$(T[n-1],\ldots,T[0])$

(8-3-1) U_k : S_n = Sign$((pk_{B_i}, nonce_{B_i}, H(T[n-1])), sk_{U_k})$

(8-3-2) U_k : $T[n] = ((pk_{B_i}, nonce_{B_i}, H(T[n-1])), S_n)$

(8-3-3) $U_k \rightarrow B_i$: $(T[n]\ldots,T[0])$

(8-4-0) B_i : for i = $[n,\ldots,1]$

(8-4-1) B_i : $(pk_{U_i}, nonce_{U_i}, H(T[i-1],\ldots,T[0]), S_i) = T[i]$

(8-4-2) B_i : if Verif $((S_i, (pk_{U_i}, nonce_{U_i}, H(T[i-1],\ldots,T[0])))_{pk_{U_i}}$ = True then

(8-4-3) B_i : $(pk_{B_i}, nonce_{B_i}, H(T[0]), S_1) = T[1]$

(8-4-4) B_i : if Verif $((S_1, (pk_{B_i}, nonce_{B_i}, H(T[0])))_{pk_{B_i}}$ = True then

(8-4-5) B_i : $(v, y, inf_{B_i}, pk_{B_i}, nonce_{B_i}, S_0) = T[0]$

(8-4-6) B_i : if Verif $((v, y, inf_{B_i}, pk_{B_i}, nonce_{B_i}), S_0)_{pk_{B_0}, v, y}$ = True then

(8-5-1) $B_i \rightarrow U_k$: signature_true

(8-5-2) U_k : if signature_true = true then

(8-5-3) B_i : Insert $(T[n],\ldots,T[0])$

(8-5-4) U_k : Delete $(T[n-1],\ldots,T[0])$

(9) 還収

本稿で着目する二重使用については，還収フェーズにおいて二重使用通貨を含む全ての通貨が発行銀行に還ってくることから，一般に還収フェーズにおいて検知可能となる想定である．市中銀行 B_i は発行銀行 B_0 に自身の公開鍵証明書と還収通貨を送付．発行銀行 B_0 は市中銀行 B_i の公開鍵証明書を検証，還収通貨の署名連鎖 $(T[n],\ldots,T[0])$ の署名検証を行う，または，発行銀行のサーバ負荷低減のため，市中銀行の業務フローを信頼して，還収通貨の署名 $T[0]$ のみ署名検証を行い，発行銀行 B_0 の管理する発行済み通貨 DB を還収済みに更新，還収通貨の署名検証結果を受け取った市中銀行 B_i は，自身の管理する使用済み通貨 DB_i から還収通貨 $(T[n],\ldots,T[0])$ を削除する．

より詳細には，(9-1-1) で市中銀行 B_i は発行銀行 B_0 に，市中銀行 B_i の公開鍵証明書，還収通貨 $(T[n],\ldots,T[0])$ を送付，(9-1-2) で発行銀行 B_0 は認証局 A_0 の pk_{A_0} の自己署名証明書を検証，(9-1-3) で発行銀行 B_0 は中間認証局 A_1 の pk_{A_1} の公開鍵証明書を検証，(9-1-4) で発行銀行 B_0 は市中銀行 B_i の公開鍵証明書を検証，(9-1-5)(9-1-6) で発行銀行 B_0 は還収通貨 $T[0]$ の署名検証を行う．(9-2-1) で発行銀行 B_0 は還収通貨 $(T[n],\ldots,T[0])$ の発行済み通貨 DB の状態を更新，(9-2-2) で発行銀行 B_0 は還収通貨 $(T[n],\ldots,T[0])$ を還収済み通貨 DB に格納．(9-2-3)(9-2-4) で発行銀行 B_0 は市中銀行 B_i に通貨一式の署名検証結果を送付，(9-2-5) で市中銀行 B_i は還収済み通貨 $(T[n],\ldots,T[0])$ を削除する．

(9-1-1) $B_i \rightarrow B_0$: $(cert_{pk_{B_i}}, (T[n],\ldots,T[0]))$

(9-1-2) B_0 : $(pk_{A_0}, sign_{pk_{A_0}}) = cert_{pk_{A_0}}$ if Verif $(pk_{A_0}, sign_{pk_{A_0}})_{pk_{A_0}}$ = True then

(9-1-3) B_0 : $(pk_{A_1}, sign_{pk_{A_1}}) = cert_{pk_{A_1}}$ if Verif $(pk_{A_1}, sign_{pk_{A_1}})_{pk_{A_0}}$ = True then

(9-1-4) B_0 : $(pk_{B_i}, sign_{pk_{B_i}}) = cert_{pk_{B_i}}$ if Verif $(pk_{B_i}, sign_{pk_{B_i}})_{pk_{A_0}}$ = True then

(9-1-5) B_0 : $(v, y, inf_{B_i}, pk_{B_i}, nonce_{B_i}, S_0) = T[0]$

(9-1-6) B_0 : if Verif $((v, y, inf_{B_i}, pk_{B_i}, nonce_{B_i}), S_0)_{pk_{B_0}, v, y}$ = True then

(9-2-1) B_0 : Update Currency_DB(B_0) $(T[n],\ldots,T[0])$

(9-2-2) B_0 : Insert Currency_DB(B_i) $(T[n],\ldots,T[0])$

(9-2-3) $B_0 \rightarrow B_i$: signature_true
(9-2-4) B_i : if signature_true = true then
(9-2-5) B_i : Delete $(T[n], \ldots, T[0])$

(10) 鍵追加

仮名性や匿名性の担保に必要な通貨署名鍵の都度追加／生成機能である．詳細は次回作で述べることとしたい．

(11) 更新

古い（署名数の多い）通貨の更新／リフレッシュ機能である．市中銀行に通貨が戻ったタイミングで，通貨が市中で利用された後の $T[0]$ への付加情報 $(T[n], \ldots, T[1])$ の更新を行い，更新通貨を未使用通貨 DB に保存する．更新後の通貨 $T[0]$ を利用者の次の引出や両替の要求に対して利用する．市中銀行 B_i の署名鍵 sk_{B_i} で署名を行う．

$S[n+1] = ((pk_{U_j}, nonce_{U_j}, H(T[0])), sk_{B_i})$
$T[n+1] = ((pk_{U_j}, nonce_{U_j}, H(T[0])), S[n+1])$
$(T[n+1], T[0])$

再更新においては，一度更新済みの通貨 $(T[n+k], \ldots, T[n+1], T[0])$ について, 再び通貨付加情報 $(T[n+k], \ldots, T[n+1])$ の更新を行い，再利用する．更新済み通貨を管理する通貨 DB において，更新時に削除した付加情報を結合する $(T[n+k], \ldots, T[1]) = (T[n+k], \ldots, T[n+1]) \mid T[n], \ldots, T[1])$ ．更新済み通貨について, 再び還収を行う場合には，通貨付加情報を結合して，還収通貨 $(T[n+k], \ldots, T[0])$ を発行銀行に送付する。本フェーズのより詳細は次回作で述べたい．

5. 実装方針と検証結果および考察

前章および [11] で述べた検証対象プロトコルについて，ProVerif による形式的な実装を行い，すべてのフェーズについて到達可能性 (reachability) を確認できた．以降，各安全性要件検証のためのプロトコル実装やクエリ設計に関する方針および工夫点を述べる．

5.1 還収時における二重使用検知

一般に，トークン型電子現金方式では，市中で二重使用が行われた場合に，二重使用が行われた時点で検知することは出来ないものの，発行銀行に全ての通貨が還ってくる還収の時点で，通貨の使用履歴から二重使用を検知することが出来る．特に，今回の検証対象プロトコルにおいては，使用履歴が通貨付加情報 $(T[n], \ldots, T[1])$ として署名連鎖の形で通貨トークン $T[0]$ に付属しており，署名連鎖の分岐を検知することで，当該通貨トークンの二重使用を検知することが出来る．あるいは，同じ $T[0]$ を有する通貨トークンが複数回還収された場合も，当該通貨トークンの二重使用検知と捉えることが出来る．本稿の検証においては，還収時に上記いずれかの事象が発生した場合に相当する ProVerif のイベントを配置して，当該イベントへの到達可能性 (reachability) により二重使用検知を試みた．より詳細の実装としては，還収されてきた判定対象通貨の署名連鎖を構成する通貨付加情報および通貨 $T[0]$ を還収済み通貨 DB と比較する実装とした（以下，署名検証処理およびテーブル探索処理の詳細を省略した疑似コードである）．

```
query event(double_spend_detected);

let BO_CHECK_PROCESS =
in(BO_spent_DB_CALL,Tkto0:bitstring);
(* 以下を還収通貨の署名連鎖(T[k,...,0])について繰り返す *)
let (Tk:bitstring,Tkm1to0:bitstring) = Tkto0
    in
(
 get BO_spent_DB(=Tk) in
 (
  (* 対象通貨が還収済み通貨リストに有る場合*)
  out(BO_spent_DB_RES,(double_spend_detected))
      ;
 )
 else
 (
  (* 対象通貨が還収済み通貨リストに無い場合*)
  out(BO_spent_DB_RES,(not_double_spend));
  (* 二重使用検知がなければ新たに還収済み通貨に追加*)
  insert BO_spent_DB(Tk);
 )
 (* 還収通貨(T[k-1,...,0])を再帰的に検証 *)
 out(BO_spent_DB_CALL,(Tkm1to0))
).
```

上記の実行結果として，還収時の二重使用検知イベントへの到達可能性は cannot be proved，さらにトレースを確認することで reached（到達可能）であることが分かった．すなわち，攻撃者は二重使用検知イベントに到達することが出来る，二重使用を起こすことが可能ということであり，これは既存研究の知見と整合する．

- 二重使用検知イベントへの到達可能性 (reachability) : cannot be proved (トレースの確認結果は reached)

ただし，二重使用検知された通貨が市中を流通した通貨であることは，今回の実行結果のトレースからは読み取れず，実装の改善が必要である．市中を流通する過程で二重使用が起きた通貨を還収時に検知するための，システム全体としての検知機構の設計については今後の課題である．

5.2 端末上のセキュアハードウェアによる対策

二重使用が行われた時点で検知および防止する対策としては, 端末上のセキュアハードウェアを用いた対策が有効と考えられる．例えば, 多くのスマートフォンに搭載されていることが期待されるセキュアエレメント (Secure Element)

や ARM TrustZone などの TEE(Trusted Execution Environment) が本対策の実装においては有効である [10], [12]. これらを本稿ではセキュアハードウェアと呼称する.

上記のセキュアハードウェアは，同じ端末上の一般のアプリとは隔離された実行環境が提供されており，既に多くのスマートフォン上で認証系アプリや決済系アプリを通信キャリアや端末ベンダの特権的な管理下で動作させることを実現している．これにより正規のアプリが改ざんされること無く実行されることが保証される．また，署名鍵や認証情報などの秘匿情報を一般アプリ領域から隔離して格納することも可能であり，これにより正規アプリが利用する秘匿情報が一般アプリから利用できないことが保証される.

本稿では，セキュアハードウェアに通貨署名鍵を格納して，一般アプリ領域で動作する電子現金クライアントアプリが通貨署名を行いたい場合に，必ずセキュアハードウェアに署名依頼する構成を想定する．さらに，セキュアハードウェアには，当該端末の通貨使用履歴 DB を管理することで，署名対象通貨について既に署名依頼が届いていないか，すなわち二重使用を試みる署名依頼でないかを監視することが出来る．より詳細の実装としては，署名依頼された判定対象通貨の署名連鎖を構成する通貨付加情報 $T[n-1],\ldots,T[1]$ が当該端末の通貨使用履歴 DB と一致するかを比較する実装とした（以下，通貨の分割処理およびテーブル探索処理の詳細を省略した疑似コードである）.

```
let HARDWARE_PROCESS_Uj(skUj:sskey) =
(* T[n-1] のハッシュ値 H_Tp_nm1 に署名する*)
in(SE_API_call_Uj,(pkUk:spkey, Uk_nonce:
    bitstring, H_Tp_nm1:bitstring));
(
 get UserDevice_HARDWARE_DB_Uj(=H_Tp_nm1) in
 (
  (* 二重使用検知有り*)
  event Uj_HW_double_spend_detected(H_Tp_nm1);
 )
 else
 (
  (* 二重使用検知無し*)
  event Uj_HW_NON_double_spend(H_Tp_nm1);
  let Spn = Sign((pkUk, Uk_nonce, H_Tp_nm1),
      skUj) in
  let Tpn = (pkUk, Uk_nonce, H_Tp_nm1, Spn) in
  (* 使用した通貨 T[n-1] のハッシュ値
      H_Tp_nm1 を通貨使用履歴に追加する*)
  insert UserDevice_HARDWARE_DB_Uj(H_Tp_nm1);
  (* 新規に署名した通貨をユーザ端末に返す*)
  out(SE_API_call_Uj,(Tpn,Spn))
 )
).
```

上記のチェック処理を利用者の端末上に追加した場合に，端末上において二重使用が発生するか否かを検証するクエリは下記を設定した．本クエリの意味としては，支払ユーザ U_j の支払処理が正常に完了した場合に，U_j の端末上では二重使用は発生していないということである.

```
query event(Uj_spend_end) ==>
event(Uj_HW_NON_double_spend).
```

上記クエリの実行結果としては True（支払処理が正常終了した時に二重使用は発生していない）が出力された．すなわち，上記の対策を行った端末上では，攻撃者が二重使用を行った場合，支払処理を正常終了することが出来ないということである.

- 対策を行った端末上において支払処理が正常終了した時の二重使用の有無 : True（二重使用は発生しない）

ただし，還収時における二重使用検知イベントは到達可能 (reachable) のままであった．本結果は，チェック処理を追加していない端末や他エンティティが存在すれば，システム全体としては二重使用を防止できていないと解釈できる.

- 還収時における二重使用検知イベントの到達可能性 : cannot be proved (トレースの確認結果は reached)

そこで，市中銀行，利用者（個人），利用者（店舗），等の ProVerif の全てのプロセスに本節のチェック処理を追加する方針とした．本チェック処理を全てのエンティティ上に追加した後の還収時における二重使用検知イベントへの到達可能性は True が出力されることが期待されるが，本稿執筆時点では実行結果の解析の途上である．本対策の社会実装を想定する上では，電子現金サービスに参画する各エンティティへの要件として，上述のセキュアハードウェアを用いた対策を課す必要がある．現在のスマートフォン端末へのセキュアハードウェアの普及状況を鑑みると，実現できないセッティングでは無いことが期待される.

5.3 利用者間の支払フェーズにおける仮名性の担保

利用者間の支払フェーズにおける仮名性の担保，言い換えると，支払に使われた通貨付加情報 $T[n],\ldots,T[1]$ から支払処理を行った利用者の個人情報に辿り着けないこと，利用者を特定／追跡できないことを，ProVerif により検証する方針について検討を行った.

ProVerif ガイド [3] および e-cash プロトコルの形式検証の先行研究 [8] によれば，e-voting や e-cash の署名からの署名者（署名鍵）の特定が出来ないことを，観測等価性 (observational equivalence) で表現することとしている．これは暗号理論における識別不可能性 (indistinguishability) に相当する．具体的な ProVerif における実装としては，choice 文および nointerf 文が候補として挙げられている.

前述の仮名性を choice 文を用いて ProVerif で実装するための表現例は下記である．ここでは，ProVerif ガイドにおける e-voting の記述例に代わり，より最新の関連研究 [2] を参考としてクエリ設計の改良を行った.

```
let U_jk_sA =
choice[(skUj,Auth_pkUj), (skUh,Auth_pkUh)] in
(* ・・・(中略)・・・*)
(!U_jk_process(U_jk_sA,Auth_pkA1,Auth_pkA0))
```

上記の仮名性を検証するクエリの実行結果は，True（識別不可能）となった．すなわち，攻撃者は通貨署名鍵の異なる複数ユーザプロセス間を識別不可能ということである．

- 仮名性 (Pseudonymity)：True（識別不可能）

今回は，プライバシー要件の内，仮名性のみを検証対象として，結合不可能性 (unlinkability) については検証対象外とした．結合可能であることは，ある通貨支払時に仮名（すなわち公開鍵）に対応する実名が特定されると，それまでに同じ鍵で行ったすべての支払／取引が紐付いて特定されるということであり，プライバシー脅威としてはもちろん，さらなる被害に繋がる可能性があり得るため，同等の効率であれば，結合不可能性を提供できることが望ましい．ただし，今回対象プロトコルではユーザの公開鍵に認証局の証明書を発行するため，結合不可能性を提供するためには，公開鍵を一回限りの使用とすること，すなわち支払毎に認証局がユーザの公開鍵の証明書を発行する必要が有り，非効率的となる．また，公開鍵を一回限りの使用とした場合，二重使用検知が困難になる可能性もある．そのため，今回対象プロトコルでは結合不可能性を提供対象外として，仮名性を提供対象とした．今後の課題としては，上記の相反する要件に対応した署名技術およびプロトコルを検討するとともに，結合不可能性を ProVerif で記述するためのクエリ設計の改良を行う予定である．

6. おわりに

本研究では，トークン型電子現金プロトコルの安全性として，特にセキュリティ要件（二重使用検知）とプライバシー要件（仮名性）について，形式検証ツール ProVerif を用いた手法により検証を試みた．上記目的のために，今回対象プロトコルの通貨発行／引出／支払／預入／還収フェーズについて ProVerif による形式化を行い，すべてのフェーズが ProVerif により到達可能であることを確認した．その上で，二重使用への対策となることを目指して，セキュアハードウェアを用いた端末上の二重使用チェック処理を ProVerif により形式的に実装した．本対策の実装時の検証結果として True（端末上の正常終了時に二重使用が発生しない）を確認した．また，仮名性を検証するためのクエリ設計として，観測等価性を表現する ProVerif の choice 文を用いた実装を検討して，仮名性の検証結果として True（仮名／識別不可能）であることを確認した．本稿で活用した形式検証におけるプロトコル表現の工夫点については，2023 年 3 月の応用数理学会でご紹介する予定である [13].

謝辞 暗号プロトコルに関するご指導を頂きました，北陸先端科学技術大学院大学の藤崎英一郎先生に感謝を申し上げます．また，本研究プロジェクトに多くのご協力をいただいた，京都大学の荒川さん，長崎大学の山本さんと江島さんに，感謝を申し上げます．

参考文献

[1] Masayuki Abe and Eiichiro Fujisaki. How to date blind signatures. In *ASIACRYPT*, volume 1163 of *Lecture Notes in Computer Science*, pages 244–251. Springer, 1996.
[2] Karthikeyan Bhargavan, Vincent Cheval, and Christopher Wood. A symbolic analysis of privacy for TLS 1.3 with encrypted client hello. CCS '22, page 365–379. Association for Computing Machinery, 2022.
[3] Bruno Blanchet. Automatic verification of security protocols in the symbolic model: The verifier proverif. In *Foundations of security analysis and design VII*, pages 54–87. Springer, 2013.
[4] Sébastien Canard, Aline Gouget, and Jacques Traoré. Improvement of efficiency in (unconditional) anonymous transferable e-cash. In *Financial Cryptography*, volume 5143 of *Lecture Notes in Computer Science*, pages 202–214. Springer, 2008.
[5] David Chaum. Blind signatures for untraceable payments. In *CRYPTO*, pages 199–203, 1982.
[6] David Chaum, Amos Fiat, and Moni Naor. Untraceable electronic cash. In *CRYPTO*, volume 403 of *Lecture Notes in Computer Science*, pages 319–327. Springer, 1988.
[7] David Chaum, Christian Grothoff, and Thomas Moser. How to issue a central bank digital currency, 2021.
[8] Jannik Dreier, Ali Kassem, and Pascal Lafourcade. Formal analysis of e-cash protocols. In *SECRYPT*, pages 65–75. SciTePress, 2015.
[9] Tatsuaki Okamoto and Kazuo Ohta. Disposable zero-knowledge authentications and their applications to untraceable electronic cash. In *CRYPTO*, volume 435 of *Lecture Notes in Computer Science*, pages 481–496. Springer, 1989.
[10] 磯部 光平 and 宇根 正志. スマートフォン等のスマート・デバイスにおけるセキュリティ：プラットフォーム化によるリスクの現状と展望. In **日本銀行金融研究所ディスカッション・ペーパー・シリーズ** *(2020-J-17/***情報技術***)*.
[11] 奥田 哲矢, 荒井 研一, 齋藤 恆和, 千田 浩司, 中林 美郷, 山村 和輝, 宮澤 俊之, and 阿部 正幸. トークン型電子現金方式の形式検証手法に関する初期検討. In *CSEC99*, 2022.
[12] 須崎 有康. Trusted execution environment の実装とそれを支える技術. **電子情報通信学会 基礎・境界ソサイエティ** *Fundamentals Review*, 14(2):107–117, 2020.
[13] 山本 輪, 江島 奨悟, 奥田 哲矢, and 荒井 研一. Proverif によるトークン型電子現金プロトコルの形式検証. In **日本応用数理学会 研究部会連合発表会（***FAIS* **オーガナイズド・セッション）**, 2023.
[14] 大塚 玲. 耐タンパー性に基づくデジタル通貨ウォレットの研究動向ー匿名性と透明性の両立に向けてー. In **日本銀行金融研究所ディスカッション・ペーパー・シリーズ** *(2022-J-9/***情報技術***)*.