情報処理学会論文誌 Vol. 49 No. 9 3001–3015 (Sep. 2008)

状態遷移秘匿プロトコルの提案とその応用

平原耕一[†1,*1] 渡邉昭宏[†1] 武田正之[†2]

本稿では，状態遷移アルゴリズム（決定性有限オートマトン，チューリングマシン）について，その遷移関数と入力値となる文字列の双方を秘匿したまま受理判定や計算などの処理を行う，Oblivious Transfer という暗号プロトコルを使用したプロトコル（OSTP，OTMP）を提案する．また，OSTP，OTMP が有用性を持つ具体的な例として，安全なメールフィルタリングサービス，電子透かし埋め込みサービス，グループ共有鍵作成サービスの構成をそれぞれ示す．メールフィルタリングサービスでは，メールサーバからフィルタリングアルゴリズムとフィルタリング条件を秘匿し，フィルタリングサーバからメールを秘匿することが両立可能であるという点で安全である．電子透かし埋め込みサービスでは，ユーザから埋め込みアルゴリズムを秘匿し，サービス提供者から埋め込みに必要なデータを秘匿することが両立可能であるという点で安全である．また，グループ共有鍵作成サービスでは，ユーザから作成アルゴリズムを秘匿し，サービス提供者から固有情報を秘匿することが両立可能であるという点で安全である．

Oblivious State Transition Protocol and Its Applications

Koichi Hirahara,[†1,*1] Akihiro Watanabe[†1] and Masayuki Takeda[†2]

In this paper, we propose two protocols by using Oblivious Transfer for the state transition algorithms (Deterministic Finite Automaton and Turing Machine). These protocols fill following requirements. Server that has the state transition algorithm cannot obtain the client's input. Client that has the input string can only obtain processing result. We show examples of OSTP and OTMP for e-mail filtering service, digital watermarking service and group shared key generating service. In the mail filtering service, the mail server has mails and the filtering server has filtering conditions. In the digital watermarking service, the user has a image and data for embedding and the server has the embedding algorithm. In the group shared key generating service, each user has his secret share which is used key generation and the server has the key generating algorithm.These services are secure since they can mutually hide the algorithm and data secretly.

1. はじめに

近年，個人情報保護への関心が高まっている．その理由としては，メーカやインターネットプロバイダなどから個人情報の漏洩事件が多発していることや，個人情報保護法の施行により個人情報の取扱いがこれまで以上に厳重になったことがあげられる．現在，個人情報保護に関連して様々なセキュリティ技術の研究が行われている．

本稿では，状態遷移アルゴリズム（決定性有限オートマトン，チューリングマシン）について，その遷移関数と入力値となる文字列の双方を秘匿したまま受理判定や計算などの処理を行う，Oblivious Transfer（紛失通信，OT）[1] という暗号プロトコルをサブプロトコルとして使用したプロトコル（OSTP，OTMP）を提案する．

また，OSTP，OTMP が有用性を持つ具体的な例として，安全なメールフィルタリングサービス，電子透かし埋め込みサービス，グループ共有鍵作成サービスの構成をそれぞれ示す．メールフィルタリングサービスではメールサーバとフィルタリングサーバが異なる組織により運営されていて，メールサーバがメールを，フィルタリングサーバがフィルタリングアルゴリズムとフィルタリング条件を持つとき，メールサーバからフィルタリングアルゴリズムとフィルタリング条件を秘匿し，フィルタリングサーバからメールを秘匿することが両立可能であるという点で安全である．電子透かし埋め込みサービスでは，ユーザが画像と埋め込み用データを，サービス提供者が埋め込みアルゴリズムを持つとき，ユーザから埋め込みアルゴリズムを秘匿し，サービス提供者から埋め込みに必要なデータを秘匿することが両立可能であるという点で安全である．また，グループ共有鍵作成サービスでは，ユーザが共有鍵を作成するための各ユーザ固有の情報を，サービス提供者が作成アルゴリズムを持つとき，ユーザから作成アルゴリズムを秘匿し，サービス提供者から固有情報を秘匿することが両立可能であるという点で安全である．

本稿の以降の構成は次のとおりである．2 章で，既存研究の動向について示す．3 章で，本提案で使用する既存暗号プロトコルである OT について示す．4 章で，状態遷移アルゴリ

†1 東京理科大学大学院理工学研究科情報科学専攻
Graduate School of Science and Technology, Tokyo University of Science
†2 東京理科大学理工学部情報科学科
Department of Information Sciences, Tokyo University of Science
∗1 現在，株式会社富士通エフサス
Presently with FUJITSU FSAS INC.

ズムと入力値の双方を秘匿して処理を行うプロトコルをそれぞれ示す．5 章で，4 章で述べた手法について安全性の考察を示す．6 章で，OSTP，OTMP のネットワークサービスへの応用を示す．7 章で本稿をまとめる．

2. 関 連 研 究

ネットワーク上においてプログラム提供者の処理アルゴリズムと利用者の入力値の双方を秘匿する手法の研究としては，主に次のものがあげられる．

文献 2) や 3) では OT を利用してプライバシ要件を満たしたままチューリングマシンの実行を行う手法が提案されており，本研究の目的に近い研究といえる．

秘密回路計算という技術は，任意の論理回路について入力データや回路のロジックを秘匿したまま処理を実行するものである．文献 4) では，秘密回路計算技術を実装し，応用例として共通鍵暗号を用いて暗号化されたデータを復元することなく文字検索する手法を提案している．

また別のアプローチとして，文献 5) では検索サービスにおいて，検索サービスの提供者が何も得られずに，利用者が検索結果を得られる手法を提案している．提案手法では Oblivious Polynomial Evaluation（紛失多項式評価，OPE）[6] という暗号プロトコルがサブプロトコルとして使用されている．ここで，OPE とは 1999 年に Naor らによって提案された，OT を基礎プロトコルとする 2 者間の暗号プロトコルで，入力者の持つ入力値と式所有者の持つ一変数多項式を互いに相手に漏らすことなく多項式計算を行い，入力者だけが計算結果を得られるという特徴を持つ．

上述の OPE を拡張したプロトコルとしては文献 7) があげられる．文献 7) では，OPE で式所有者が秘匿できる式の種類を任意の n 変数多項式へと拡張し，応用例として安全な情報埋め込みサービスの構成例を示している．

3. Oblivious Transfer

3.1 1-out-of-2 Oblivious Transfer

Oblivious Transfer を利用すると，汎用的なマルチパーティプロトコルを実現できることが知られている．最も基本的な OT の形式は 1-out-of-2 OT であり，以下のような機能を実現するプロトコルである．受信者 A は 1 ビットの秘密 a を持っており，送信者 B は，2 つの 1 ビットの秘密 m_0,m_1 を持っている．プロトコル終了後，A は m_a を得る．その際，次の 2 つの要件を満たす．

(1) A は，m_{1-a} についてまったく分からない．

(2) B は，a についてまったく分からない．

3.2 k-out-of-n Oblivious Transfer

1-out-of-2 OT の自然な拡張として定義される k-out-of-n OT は次のような機能を実現するプロトコルである．受信者 A は k（$\leq n$）個の秘密 $a_1, a_2, ..., a_k$（$a_i \in \{1, ..., n\}, i = 1, ..., k$）を持っており，送信者 B は n 個の秘密 $m_1, m_2, ..., m_n$ を持っている．プロトコル終了後，A は $m_{a_1}, m_{a_2}, ..., m_{a_k}$ を得る．

その際，次の 2 つの要件を満たす．

(1) A は，$m_{a_1}, m_{a_2}, ..., m_{a_k}$ 以外についてまったく分からない．

(2) B は，$a_1, a_2, ..., a_k$ についてまったく分からない．

4. 提案プロトコル

本章では状態遷移の情報を秘匿する対象として決定性有限オートマトン（Deterministic Finite Automaton．以下，DFA）とチューリングマシン（Turing Machine．以下，TM）の場合を考え，次の 2 種類のプロトコルを提案する．

- Oblivious State Transition Protocol（OSTP）
- Oblivious Turing Machine Protocol（OTMP）

4.1 Oblivious State Transition Protocol

4.1.1 要　　件

Oblivious State Transition Protocol（以下，OSTP）では，入力者 A と DFA 所有者 B が存在し，入力者 A は DFA への入力文字列 w を持っている．プロトコルの実行後，入力者 A は w が受理されたかどうかの結果を得る．その際，次の要件を満たす．

(1) 入力者 A は DFA の状態数以外の遷移関数の情報に関してまったく分からない．

(2) DFA 所有者 B は，入力者 A の入力文字列 w（文字数を除く）と受理されたかどうかに関してまったく分からない．

4.1.2 プロトコル

このプロトコルで対象となる DFA $M =< Q, \Sigma, q_0, \delta, F >$ を次のように定義する．状態数 n，状態の集合 $Q = \{q_0, q_1, ..., q_{n-1}\}$，入力文字の集合 $\Sigma = \{0, 1\}$，遷移関数 $\delta : Q \times \Sigma \to Q$，開始状態 q_0，終了状態の集合 $F \subset Q$ である．

［初期化］入力者 A は，入力文字列 $w \in \Sigma^*$ を用意し，初期状態を表す変数 $r_0 = 0$ を定義する．なお r_0 という表記は (Step5) で述べる r_p と状態番号を表すという点が共通なの

で，便宜上表記を合わせたものである．ここで，w の文字列長を $|w|$ と表し，文字列 w の p 番目の文字を w_p と表す（w_1 は w の先頭文字を表す）．なお，$|w| < p$ のとき $w_p = null$ であるとする．

DFA 所有者 B は DFA M を用意し，状態数 n を A に知らせる．また各状態 q_j $(j = 0, 1, ..., n-1)$ に対して，$N_0(q_j) = j$ を定義し，$N_0^{-1}(j) = q_j$ とする．

［プロトコル本体］

(Step 1) p の初期値を 1 として，(Step 2) から (Step 6) の処理で状態遷移を行う．

(Step 2) $w_p = null$，すなわち文字入力がすべて完了していれば (Step 7) へ進む．そうでなければ (Step 3) へ進む．

(Step 3) B は値 $\{0, 1, ..., n-1\}$ を各状態 $\{q_0, q_1, ..., q_{n-1}\}$ に重複しないようランダムに割り当てる．ここで，各状態 q_j に割り当てた値を $N_p(q_j)$ で表す．また，$N_p^{-1}(\bullet)$ を $N_p(\bullet)$ の逆写像とする．

(Step 4) 各状態 q_j に対して

$(N_{p-1}(q_j), 0) : N_p(\delta(q_j, 0))$

$(N_{p-1}(q_j), 1) : N_p(\delta(q_j, 1))$

という 2 個の値をそれぞれ用意する．よって用意する値は全部で $2n$ 個となる．ここで，コロンの右側の値はデータ，コロンの左側の値は，(Step 5) で OT を行ううえでのデータのインデックスを表す．

(Step 5) A は B と 1-out-of-$2n$ OT を実行する．A は w_p を DFA に入力した結果に相当する情報を得るため，インデックスが (r_{p-1}, w_p) である値を選択する．

つまり，p 文字目の入力文字が 0 ならば

$N_p(\delta(N_{p-1}^{-1}(r_{p-1}), 0))$

1 ならば

$N_p(\delta(N_{p-1}^{-1}(r_{p-1}), 1))$

を得る．ここで，A の得た値を r_p とおく．r_p は現在の DFA の状態番号を A から秘匿するための値である．

(Step 6) p の値を 1 増やし，(Step 2) へ戻る．

(Step 7) B は，各状態 q_j に対して $N_{p-1}(q_j) : F(q_j)$ を用意する．ここで，$F(q_j)$ は状態 q_j が終了状態なら 0，そうでなければ 1 を表す．また (Step 4) と同様にコロンの左側の値は，(Step 8) で OT を行ううえでのインデックスである．

(Step 8) A と B は 1-out-of-n OT を実行する．A は w が受理されたかどうかを知るため，インデックスが r_{p-1} である値を選択する．ここで得た値から A は入力文字列 w が受理されたかを知る．すなわち，1 が得られたら入力文字列は受理され，そうでなければ受理されなかったことになる．

以上が OSTP のプロトコルである．

4.1.3　OSTP の実行例

図 1 の DFA を用いて，OSTP の実行例を示す．

［初期化］入力者 A の持つ入力文字列を $w = 1101$ とし，$r_0 (= N_0(q_0)) = 0$ とする．

DFA 所有者 B の持つ DFA は，状態の集合が $Q = \{q_0, q_1, q_2, q_3\}$，開始状態が q_0，終了状態の集合が $F = \{q_3\}$ である．また，遷移関数 δ は**表 1** で定義される．

そして，B が各状態 q_j に割り当てる値 $N_0(q_j)$ は以下のとおりである．

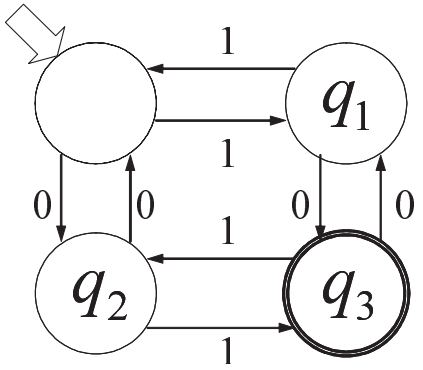

図 1　DFA の状態遷移図
Fig. 1　State transition diagram of DFA.

表 1　DFA の状態遷移表
Table 1　State transition table of DFA.

	0	1
q_0	q_2	q_1
q_1	q_3	q_0
q_2	q_0	q_3
q_3	q_1	q_2

	q_0	q_1	q_2	q_3
$N_0(\bullet)$	0	1	2	3

［プロトコルの実行］

(1 文字目の入力) $p=1$ とする．$w_p=w_1=1$，すなわち A の 1 文字目の入力文字は 1 である．B は各状態 q_j にランダムに値を割り当て，それを $N_1(q_j)$ とする．割り当てた値を以下に示す（$N_0(q_j)$ の値も併記している）．

	q_0	q_1	q_2	q_3
$N_0(\bullet)$	0	1	2	3
$N_1(\bullet)$	2	3	0	1

また，B は各状態 q_j に対して，

$(N_0(q_j),0):N_1(\delta(q_j,0))$

$(N_0(q_j),1):N_1(\delta(q_j,1))$

という値を用意する．よって，用意する値は以下のとおりである．

$(0,0):0$　$(0,1):3$

$(1,0):1$　$(1,1):2$

$(2,0):2$　$(2,1):1$

$(3,0):3$　$(3,1):0$

A は B と 1-out-of-8 OT を実行する．ここで，A はインデックスが $(r_0,w_1)=(0,1)$ である値を選択する．よって，A が得る値は 3 である．$r_1=3$ とおく．

ここまでで，1 文字目を入力したときの状態遷移を実現している．A の得た値 3 は，B が q_1 に割り当てた値であり，実際に状態 q_0 から 1 を入力したときの遷移先は q_1 である．ただし，A は値 3 と q_1 との対応を知ることはできない．

(2 文字目の入力) $p=2$ とする．$w_p=w_2=1$，すなわち A の 2 文字目の入力文字は 1 である．B は各状態 q_j にランダムに値を割り当て，それを $N_2(q_j)$ とする．割り当てた値を以下に示す（$N_1(q_j)$ の値も併記している）．

	q_0	q_1	q_2	q_3
$N_1(\bullet)$	2	3	0	1
$N_2(\bullet)$	1	2	3	0

また，B は各状態 q_j に対して，

$(N_1(q_j),0):N_2(\delta(q_j,0))$

$(N_1(q_j),1):N_2(\delta(q_j,1))$

という値を用意する．よって，用意する値は以下のとおりである．

$(0,0):1$　$(0,1):0$

$(1,0):2$　$(1,1):3$

$(2,0):3$　$(2,1):2$

$(3,0):0$　$(3,1):1$

A は B と 1-out-of-8 OT を実行する．ここで，A はインデックスが $(r_1,w_2)=(3,1)$ である値を選択する．よって，A が得る値は 1 である．$r_2=1$ とおく．

(3 文字目の入力) $p=3$ とする．$w_p=w_3=0$，すなわち A の 3 文字目の入力文字は 0 である．B は各状態 q_j にランダムに値を割り当て，それを $N_3(q_j)$ とする．割り当てた値を以下に示す（$N_2(q_j)$ の値も併記している）．

	q_0	q_1	q_2	q_3
$N_2(\bullet)$	1	2	3	0
$N_3(\bullet)$	3	2	0	1

また，B は各状態 q_j に対して，

$(N_2(q_j),0):N_3(\delta(q_j,0))$

$(N_2(q_j),1):N_3(\delta(q_j,1))$

という値を用意する．よって，用意する値は以下のとおりである．

$(0,0):2$　$(0,1):0$

$(1,0):0$　$(1,1):2$

$(2,0):1$　$(2,1):3$

$(3,0):3$　$(3,1):1$

A は B と 1-out-of-8 OT を実行する．ここで，A はインデックスが $(r_2, w_3) = (1, 0)$ である値を選択する．よって，A が得る値は 0 である．$r_3 = 0$ とおく．

(4 文字目の入力) $p = 4$ とする．$w_p = w_4 = 1$，すなわち A の 4 文字目の入力文字は 1 である．B は各状態 q_j にランダムに値を割り当て，それを $N_4(q_j)$ とする．割り当てた値を以下に示す（$N_3(q_j)$ の値も併記している）．

	q_0	q_1	q_2	q_3
$N_3(\bullet)$	3	2	0	1
$N_4(\bullet)$	2	1	3	0

また，B は各状態 q_j に対して，

$(N_3(q_j), 0) : N_4(\delta(q_j, 0))$

$(N_3(q_j), 1) : N_4(\delta(q_j, 1))$

という値を用意する．よって，用意する値は以下のとおりである．

$(0, 0) : 2$ $quad(0, 1) : 0$

$(1, 0) : 1$ $quad(1, 1) : 3$

$(2, 0) : 0$ $quad(2, 1) : 2$

$(3, 0) : 3$ $quad(3, 1) : 1$

A は B と 1-out-of-8 OT を実行する．ここで，A はインデックスが $(r_3, w_4) = (0, 1)$ である値を選択する．よって，A が得る値は 0 である．$r_4 = 0$ とおく．

(受理判定) $p = 5$ とする．$w_p = w_5 = null$ であるため，プロトコルの (Step 7) へ進む．B は各状態 q_j に対して，

$N_4(q_j) : F(q_j)$

という値を用意する．よって，用意する値は以下のとおりである．

$0 : 1$

$1 : 0$

$2 : 0$

$3 : 0$

A は B と 1-out-of-4 OT を実行する．ここで，A はインデックスが $r_4 = 0$ である値を選択する．よって，A が得る値は 1 である．したがって，A は入力文字列 $w = 1101$ が受理されたことが分かる．

4.2 Oblivious Turing Machine Protocol

4.2.1 要　　件

Oblivious Turing Machine Protocol（以下，OTMP）では，入力者 A と TM 所有者 B とが存在し，入力者 A は TM への入力値を持っている．プロトコルの実行後，入力者 A は TM 所有者 B の持つ TM で計算した結果を得る．その際，次の要件を満たす．

(1) 入力者 A は TM の状態数以外の遷移関数の情報に関してまったく分からない．

(2) TM 所有者 B は，入力者 A の入力値（文字数を除く）と計算結果に関してまったく分からない．

4.2.2 プロトコル

このプロトコルで対象となる TM $M =< Q, \Sigma, \Gamma, \delta, q_0, X_b, F >$ を次のように定義する．状態数 n，$Q = \{q_0, q_1, ..., q_{n-1}\}$，$\Sigma = \{X_0, X_1, ..., X_{m-1}\}$，テープ記号の有限集合 $\Gamma = \Sigma \cup \{X_b\}$（$X_b$ は空白記号，$X_m = X_b$ とおく），$F = \phi$ である．遷移関数 $\delta(q_i, X_j)$ $(i = 0, ..., n-1; j = 0, ..., m)$ の値は，値が定義されている引数の場合，[次の状態，書き込む記号，ヘッドの移動する方向] を $(q_{i,j}, X_{i,j}, D_{i,j})$ と定義する．ヘッドの移動する方向は R（右），L（左），H（移動せず）の 3 種類である．

［初期化］

A は各入力値 $s \in \Sigma^*$ をテープの左端から記入し，初期状態を表す変数 $r_0 = 0$ を定義する．なお r_0 という表記は (Step4) で述べる r_p と便宜上表記を合わせたものである．ここで，s の文字数を $|s|$，s の p 番目の文字を s_p $(0 < p \leq |s|)$ とする．また，A は B から 1 回目の入力で得る情報の復号に使用する鍵（(Step 4) で後述）$K(0, q_0, X_0), K(0, q_0, X_1), ..., K(0, q_0, X_m)$ を受信する．

B は TM M を用意し，状態数 n を A に知らせる．また $N_0(q_0) = 0$ を定義し，q_0 以外の状態 q_i（$i = 1, ..., n-1$）には，値 $\{1, ..., n-1\}$ を重複しないようランダムに割り当て，その値を $N_0(q_i)$ で表す．

［プロトコル本体］

(Step 1) $p = 1$ とする．

(Step 2) B は，状態 q_i（$i = 0, 1, ..., n-1$）に対して，値 $\{0, 1, ..., n-1\}$ を重複しないようランダムに割り当てその値を $N_p(q_i)$ で表す．

注：以下，2 回目以降の (Step 3)，(Step 4) においては，1 回前の実行において A から B に伝えられた p の増減が-1（ヘッドが左に移動）の場合は各添え字 $p-1$ を $p+1$ に，p の増減が 0（ヘッドは動かず）の場合は $p-1$ を p に置き換える．ただし，p の増減が 0 の

場合は表記上 (Step 3) の鍵の添え字が前回のものと同じになるが，各鍵は新しく作り直すものとする．

(Step 3) B は各状態 q_i（$i = 0, 1, ..., n-1$）に対して，$\delta(q_i, X_j)$ が定義されている場合は，

$(N_{p-1}(q_i), X_0)$:
$E_{K(p-1,q_i,X_0)}[(N_p(q_{i,0}), X_{i,0}, D_{i,0}, K(p, q_{i,0}, X_0), ..., K(p, q_{i,0}, X_m))]$
$(N_{p-1}(q_i), X_1)$:
$E_{K(p-1,q_i,X_1)}[(N_p(q_{i,1}), X_{i,1}, D_{i,1}, K(p, q_{i,1}, X_0), ..., K(p, q_{i,1}, X_m))]$
...
$(N_{p-1}(q_i), X_m)$:
$E_{K(p-1,q_i,X_m)}[(N_p(q_{i,m}), X_{i,m}, D_{i,m}, K(p, q_{i,m}, X_0), ..., K(p, q_{i,m}, X_m))]$

という暗号文の組をそれぞれ用意する（暗号文は計 $(m+1)n$ 個）．ここで $q_{i,j}$，$X_{i,j}$，$D_{i,j}$ はそれぞれ前述した，次に遷移する状態，テープに書き込む記号，ヘッドの移動方向であり，各行のコロンの右側の値はデータ，コロンの左側の値 $(N_{p-1}(q_i), X_j)$ は (Step 4) で OT を行ううえでのデータのインデックスを表す．また，定義されていない遷移については，以下の形の暗号文を用意する．n_{undef} は A と B との間で決めた未定義を表す値とする．

$(N_{p-1}(q_i), X_j) : E_{K(p-1,q_i,X_j)}[n_{undef}]$ $(j \in \{0, 1, ..., m\})$

ここで，$E_{K(x,y,z)}$ は鍵 $K(x, y, z)$ によって復号可能な暗号文を意味する．各鍵についている添え字は便宜上のものであり，A は鍵の添え字に関する情報を得ることができない．

(Step 4) A は B と 1-out-of-$(m+1)n$ OT を実行して，上記 (Step 3) の暗号文の 1 つを得る．A は s_p を TM に入力した結果に相当する情報を得るため，インデックスが (r_{p-1}, s_p) である値を選択しうる．取得後，A は持っている鍵のうち，$K(p-1, N_{p-1}^{-1}(r_{p-1}), s_p)$ を使用して暗号文を復号し各々の値を得る．ここで $N_p(q_{i,j})$ を新たに r_p とおく．r_p は現在の TM の状態番号を A から秘匿するための値である．そして A はテープの該当位置の値を $X_{i,j}$ で書き換える．また復号した値が n_{undef} であった場合には，遷移関数 δ の値が定義されていないのでプロトコルを終了し，A は計算結果を得る．

(Step 5) A は (Step 4) で得た $D_{i,j}$ の値が R（右へ移動）ならば $p \to p+1$，L（左へ移動）ならば $p \to p-1$，H（動かない）ならば p のままとして B に p の値の増減を伝える．その後 (Step 2) 以降の処理を繰り返す．

以上が，OTMP のプロトコルである．

4.2.3　OTMP の実行例

図 2 の TM を用いて，OTMP の実行例を示す．

B の持つ TM M は，状態の集合が $Q = \{q_0, q_1, q_2, q_3, q_4\}$，開始状態が q_0，記号の集合 $\Gamma = \{1, X_b\}$ である．また，遷移関数 δ は**表 2** で定義される．M は入力した 2 つの自然数を加算する TM である．入力値は自然数であり，値に 1 を加えた数だけ 1 をテープ上に記述する．また結果も同様に値から 1 つだけ多い数の 1 がテープに記入される．

［初期化］入力者 A の入力値を $s = 11X_b11$（$1+1$ に相当）とし，$r_0 = 0$ とする．B が各状態 q_j に割り当てる値 $N_0(q_j)$ は以下のとおりである．また，A は B から鍵 $K(0, q_0, 1)$，$K(0, q_0, X_b)$ を受信する．

	q_0	q_1	q_2	q_3	q_4
$N_0(\bullet)$	0	1	2	3	4

［プロトコルの実行］

(1 回目の入力) $p = 1$ とする．$s_p = s_1 = 1$ である．B は各状態 q_j にランダムに値を割

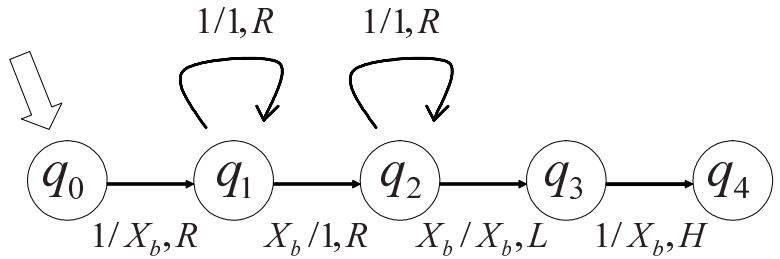

図 2　TM の状態遷移図
Fig. 2　State transition diagram of TM.

表 2　TM の状態遷移表
Table 2　State transition table of TM.

	1	X_b
q_0	(q_1, X_b, R)	-
q_1	$(q_1, 1, R)$	$(q_2, 1, R)$
q_2	$(q_2, 1, R)$	(q_3, X_b, L)
q_3	(q_4, X_b, H)	-
q_4	-	-

り当て，それを $N_1(q_j)$ とする．割り当てた値を以下に示す（$N_0(q_j)$ の値も併記している）．

	q_0	q_1	q_2	q_3	q_4
$N_0(\bullet)$	0	1	2	3	4
$N_1(\bullet)$	2	3	4	0	1

また，B は各状態 q_j に対して下記の暗号文を用意する（未定義用の値 $n_{undef} = 10000$ とする）．

$(0,1): E_{K(0,q_0,1)}[(3, X_b, R, K(1,q_1,1), K(1,q_1,X_b))]$
$(0,X_b): E_{K(0,q_0,X_b)}[10000]$
$(1,1): E_{K(0,q_1,1)}[(3,1,R,K(1,q_1,1),K(1,q_1,X_b))]$
$(1,X_b): E_{K(0,q_1,X_b)}[(4,1,R,K(1,q_2,1),K(1,q_2,X_b))]$
$(2,1): E_{K(0,q_2,1)}[(4,1,R,K(1,q_2,1),K(1,q_2,X_b))]$
$(2,X_b): E_{K(0,q_2,X_b)}[(0,X_b,L,K(1,q_3,1),K(1,q_3,X_b))]$
$(3,1): E_{K(0,q_3,1)}[(1,X_b,H,K(1,q_4,1),K(1,q_4,X_b))]$
$(3,X_b): E_{K(0,q_3,X_b)}[10000]$
$(4,1): E_{K(0,q_4,1)}[10000]$
$(4,X_b): E_{K(0,q_4,X_b)}[10000]$

A は B と 1-out-of-10 OT を実行する．ここで，A はインデックスが $(r_0, s_1) = (0,1)$ である値を選択し復号する．よって，A が得る値は

$(3, X_b, R, K(1,q_1,1), K(1,q_1,X_b))$

である．復号後，A は s_1 を X_b で書き換えヘッドを右に移動させ，$r_1 = 3$ とおく．また A は B に p の値を 1 増やすように伝える．なお，A の得た値 3 は B が q_1 に割り当てた値であり，実際には状態 q_0 から 1 を入力したときの遷移先は q_1 である．ただし，A は値 3 と q_1 との対応を知ることはできない．

(2 回目の入力) $p = 2$ であり，$s_p = s_2 = 1$ である．B は各状態 q_j にランダムに値を割り当て，それを $N_2(q_j)$ とする．割り当てた値を以下に示す（$N_1(q_j)$ の値も併記している）．

	q_0	q_1	q_2	q_3	q_4
$N_1(\bullet)$	2	3	4	0	1
$N_2(\bullet)$	4	0	1	2	3

また，B は各状態 q_j に対して下記の暗号文を用意する．

$(2,1): E_{K(1,q_0,1)}[(0, X_b, R, K(2,q_1,1), K(2,q_1,X_b))]$
$(2,X_b): E_{K(1,q_0,X_b)}[10000]$
$(3,1): E_{K(1,q_1,1)}[(0,1,R,K(2,q_1,1),K(2,q_1,X_b))]$
$(3,X_b): E_{K(1,q_1,X_b)}[(1,1,R,K(2,q_2,1),K(2,q_2,X_b))]$
$(4,1): E_{K(1,q_2,1)}[(1,1,R,K(2,q_2,1),K(2,q_2,X_b))]$
$(4,X_b): E_{K(1,q_2,X_b)}[(2,X_b,L,K(2,q_3,1),K(2,q_3,X_b))]$
$(0,1): E_{K(1,q_3,1)}[(3,X_b,H,K(2,q_4,1),K(2,q_4,X_b))]$
$(0,X_b): E_{K(1,q_3,X_b)}[10000]$
$(1,1): E_{K(1,q_4,1)}[10000]$
$(1,X_b): E_{K(1,q_4,X_b)}[10000]$

A は B と 1-out-of-10 OT を実行する．ここで，A はインデックスが $(r_1, s_2) = (3,1)$ である値を選択し復号する．よって，A が得る値は

$(0, 1, R, K(2,q_1,1), K(2,q_1,X_b))$

である．復号後，A は s_2 を 1 で書き換えヘッドを右に移動させ，$r_2 = 0$ とおく．また A は B に p の値を 1 増やすように伝える．

(3 回目の入力) $p = 3$ であり，$s_p = s_3 = X_b$ である．B は各状態 q_j にランダムに値を割り当て，それを $N_3(q_j)$ とする．割り当てた値を以下に示す（$N_2(q_j)$ の値も併記している）．

	q_0	q_1	q_2	q_3	q_4
$N_2(\bullet)$	4	0	1	2	3
$N_3(\bullet)$	1	2	3	4	0

また，B は各状態 q_j に対して下記の暗号文を用意する．

$(4,1): E_{K(2,q_0,1)}[(2, X_b, R, K(3,q_1,1), K(3,q_1,X_b))]$
$(4,X_b): E_{K(2,q_0,X_b)}[10000]$

$(0,1):E_{K(2,q_1,1)}[(2,1,R,K(3,q_1,1),K(3,q_1,X_b))]$
$(0,X_b):E_{K(2,q_1,X_b)}[(3,1,R,K(3,q_2,1),K(3,q_2,X_b))]$
$(1,1):E_{K(2,q_2,1)}[(3,1,R,K(3,q_2,1),K(3,q_2,X_b))]$
$(1,X_b):E_{K(2,q_2,X_b)}[(4,X_b,L,K(3,q_3,1),K(3,q_3,X_b))]$
$(2,1):E_{K(2,q_3,1)}[(0,X_b,H,K(3,q_4,1),K(3,q_4,X_b))]$
$(2,X_b):E_{K(2,q_3,X_b)}[10000]$
$(3,1):E_{K(2,q_4,1)}[10000]$
$(3,X_b):E_{K(2,q_4,X_b)}[10000]$

A は B と 1-out-of-10 OT を実行する．ここで，A はインデックスが $(r_2,s_3)=(0,X_b)$ である値を選択し復号する．よって，A が得る値は

$(3,1,R,K(3,q_2,1),K(3,q_2,X_b))$

である．復号後，A は s_3 を 1 で書き換えヘッドを右に移動させ，$r_3=3$ とおく．また A は B に p の値を 1 増やすように伝える．

(4 回目の入力) $p=4$ であり，$s_p=s_4=1$ である．B は各状態 q_j にランダムに値を割り当て，それを $N_4(q_j)$ とする．割り当てた値を以下に示す（$N_3(q_j)$ の値も併記している）．

	q_0	q_1	q_2	q_3	q_4
$N_3(\bullet)$	1	2	3	4	0
$N_4(\bullet)$	3	4	0	1	2

また，B は各状態 q_j に対して下記の暗号文を用意する．

$(1,1):E_{K(3,q_0,1)}[(4,X_b,R,K(4,q_1,1),K(4,q_1,X_b))]$
$(1,X_b):E_{K(3,q_0,X_b)}[10000]$
$(2,1):E_{K(3,q_1,1)}[(4,1,R,K(4,q_1,1),K(4,q_1,X_b))]$
$(2,X_b):E_{K(3,q_1,X_b)}[(0,1,R,K(4,q_2,1),K(4,q_2,X_b))]$
$(3,1):E_{K(3,q_2,1)}[(0,1,R,K(4,q_2,1),K(4,q_2,X_b))]$
$(3,X_b):E_{K(3,q_2,X_b)}[(1,X_b,L,K(4,q_3,1),K(4,q_3,X_b))]$
$(4,1):E_{K(3,q_3,1)}[(2,X_b,H,K(4,q_4,1),K(4,q_4,X_b))]$
$(4,X_b):E_{K(3,q_3,X_b)}[10000]$
$(0,1):E_{K(3,q_4,1)}[10000]$
$(0,X_b):E_{K(3,q_4,X_b)}[10000]$

A は B と 1-out-of-10 OT を実行する．ここで，A はインデックスが $(r_3,s_4)=(3,1)$ である値を選択し復号する．よって，A が得る値は

$(0,1,R,K(4,q_2,1),K(4,q_2,X_b))$

である．復号後，A は s_4 を 1 で書き換えヘッドを右に移動させ，$r_4=0$ とおく．また A は B に p の値を 1 増やすように伝える．

(5 回目の入力) $p=5$ であり，$s_p=s_5=1$ である．B は各状態 q_j にランダムに値を割り当て，それを $N_5(q_j)$ とする．割り当てた値を以下に示す（$N_4(q_j)$ の値も併記している）．

	q_0	q_1	q_2	q_3	q_4
$N_4(\bullet)$	3	4	0	1	2
$N_5(\bullet)$	0	1	2	3	4

また，B は各状態 q_j に対して下記の暗号文を用意する．

$(3,1):E_{K(4,q_0,1)}[(1,X_b,R,K(5,q_1,1),K(5,q_1,X_b))]$
$(3,X_b):E_{K(4,q_0,X_b)}[10000]$
$(4,1):E_{K(4,q_1,1)}[(1,1,R,K(5,q_1,1),K(5,q_1,X_b))]$
$(4,X_b):E_{K(4,q_1,X_b)}[(2,1,R,K(5,q_2,1),K(5,q_2,X_b))]$
$(0,1):E_{K(4,q_2,1)}[(2,1,R,K(5,q_2,1),K(5,q_2,X_b))]$
$(0,X_b):E_{K(4,q_2,X_b)}[(3,X_b,L,K(5,q_3,1),K(5,q_3,X_b))]$
$(1,1):E_{K(4,q_3,1)}[(4,X_b,H,K(5,q_4,1),K(5,q_4,X_b))]$
$(1,X_b):E_{K(4,q_3,X_b)}[10000]$
$(2,1):E_{K(4,q_4,1)}[10000]$
$(2,X_b):E_{K(4,q_4,X_b)}[10000]$

A は B と 1-out-of-10 OT を実行する．ここで，A はインデックスが $(r_4,s_5)=(0,1)$ である値を選択し復号する．よって，A が得る値は

$(2,1,R,K(5,q_2,1),K(5,q_2,X_b))$

である．復号後，A は s_5 を 1 で書き換えヘッドを右に移動させ，$r_5=2$ とおく．また A は B に p の値を 1 増やすように伝える．

(6 回目の入力) $p=6$ であり，$s_p=s_6=X_b$ である．B は各状態 q_j にランダムに値を割り当て，それを $N_6(q_j)$ とする．割り当てた値を以下に示す（$N_5(q_j)$ の値も併記している）．

	q_0	q_1	q_2	q_3	q_4
$N_5(\bullet)$	0	1	2	3	4
$N_6(\bullet)$	2	3	4	0	1

また，B は各状態 q_j に対して下記の暗号文を用意する．

$(0,1):E_{K(5,q_0,1)}[(3,X_b,R,K(6,q_1,1),K(6,q_1,X_b))]$
$(0,X_b):E_{K(5,q_0,X_b)}[10000]$
$(1,1):E_{K(5,q_1,1)}[(3,1,R,K(6,q_1,1),K(6,q_1,X_b))]$
$(1,X_b):E_{K(5,q_1,X_b)}[(4,1,R,K(6,q_2,1),K(6,q_2,X_b))]$
$(2,1):E_{K(5,q_2,1)}[(4,1,R,K(6,q_2,1),K(6,q_2,X_b))]$
$(2,X_b):E_{K(5,q_2,X_b)}[(0,X_b,L,K(6,q_3,1),K(6,q_3,X_b))]$
$(3,1):E_{K(5,q_3,1)}[(1,X_b,H,K(6,q_4,1),K(6,q_4,X_b))]$
$(3,X_b):E_{K(5,q_3,X_b)}[10000]$
$(4,1):E_{K(5,q_4,1)}[10000]$
$(4,X_b):E_{K(5,q_4,X_b)}[10000]$

A は B と 1-out-of-10 OT を実行する．ここで，A はインデックスが $(r_5,s_6)=(2,X_b)$ である値を選択し復号する．よって，A が得る値は

$(0,X_b,L,K(6,q_3,1),K(6,q_3,X_b))$

である．復号後，A は s_6 を X_b で書き換えヘッドを左に移動させ，$r_6=0$ とおく．また A は B に p の値を 1 減らすように伝える．

(7 回目の入力) $p=5$ であり，$s_p=s_5=1$ である．B は各状態 q_j にランダムに値を割り当て，それを $N_5(q_j)$ とする．割り当てた値を以下に示す（$N_6(q_j)$ の値も併記している）．

	q_0	q_1	q_2	q_3	q_4
$N_6(\bullet)$	2	3	4	0	1
$N_5(\bullet)$	4	0	1	2	3

また，B は各状態 q_j に対して下記の暗号文を用意する．

$(2,1):E_{K(6,q_0,1)}[(0,X_b,R,K(5,q_1,1),K(5,q_1,X_b))]$
$(2,X_b):E_{K(6,q_0,X_b)}[10000]$
$(3,1):E_{K(6,q_1,1)}[(0,1,R,K(5,q_1,1),K(5,q_1,X_b))]$
$(3,X_b):E_{K(6,q_1,X_b)}[(1,1,R,K(5,q_2,1),K(5,q_2,X_b))]$
$(4,1):E_{K(6,q_2,1)}[(1,1,R,K(5,q_2,1),K(5,q_2,X_b))]$
$(4,X_b):E_{K(6,q_2,X_b)}[(2,X_b,L,K(5,q_3,1),K(5,q_3,X_b))]$
$(0,1):E_{K(6,q_3,1)}[(3,X_b,H,K(5,q_4,1),K(5,q_4,X_b))]$
$(0,X_b):E_{K(6,q_3,X_b)}[10000]$
$(1,1):E_{K(6,q_4,1)}[10000]$
$(1,X_b):E_{K(6,q_4,X_b)}[10000]$

A は B と 1-out-of-10 OT を実行する．ここで，A はインデックスが $(r_6,s_5)=(0,1)$ である値を選択し復号する．よって，A が得る値は

$(3,X_b,H,K(5,q_4,1),K(5,q_4,X_b))$

である．復号後，A は s_5 を X_b で書き換えヘッドを移動させず，$r_5=3$ とおく．また A は B に p の値に増減がないことを伝える．

(8 回目の入力) 前回と同じく $p=5$ であり，$s_p=s_5=X_b$ である．区別のため前回のヘッダの位置を 5 番目，今回の位置を $5'$ 番目と呼ぶ．B は各状態 q_j にランダムに値を割り当て，それを $N_{5'}(q_j)$ とする．割り当てた値を以下に示す（$N_5(q_j)$ の値も併記している）．

	q_0	q_1	q_2	q_3	q_4
$N_5(\bullet)$	4	0	1	2	3
$N_{5'}(\bullet)$	1	2	3	4	0

また，B は各状態 q_j に対して下記の暗号文を用意する．

$(4,1):E_{K(5,q_0,1)}[(2,X_b,R,K(5',q_1,1),K(5',q_1,X_b))]$
$(4,X_b):E_{K(5,q_0,X_b)}[10000]$
$(0,1):E_{K(5,q_1,1)}[(2,1,R,K(5',q_1,1),K(5',q_1,X_b))]$
$(0,X_b):E_{K(5,q_1,X_b)}[(3,1,R,K(5',q_2,1),K(5',q_2,X_b))]$
$(1,1):E_{K(5,q_2,1)}[(3,1,R,K(5',q_2,1),K(5',q_2,X_b))]$
$(1,X_b):E_{K(5,q_2,X_b)}[(4,X_b,L,K(5',q_3,1),K(5',q_3,X_b))]$
$(2,1):E_{K(5,q_3,1)}[(0,X_b,H,K(5',q_4,1),K(5',q_4,X_b))]$
$(2,X_b):E_{K(5,q_3,X_b)}[10000]$
$(3,1):E_{K(5,q_4,1)}[10000]$

$(3, X_b) : E_{K(5,q_4,X_b)}[10000]$

A は B と 1-out-of-10 OT を実行する．ここで，A はインデックスが $(r_5, s_5) = (3, X_b)$ である値を選択し復号する．よって A が得る値は 10000 となるので，A はプロトコルを終了し，計算結果 $X_b111X_b\cdots$（$= 2$）を得る．

なお，各入力時での状態と A のテープの状態は次のようになる．

n 回目	入力後の状態	テープ（[] はヘッド）
0（初期状態）	q_0	$[1]1X_b11X_b\cdots$
1	q_1	$X_b[1]X_b11X_b\cdots$
2	q_1	$X_b1[X_b]11X_b\cdots$
3	q_2	$X_b11[1]1X_b\cdots$
4	q_2	$X_b111[1]X_b\cdots$
5	q_2	$X_b1111[X_b]\cdots$
6	q_3	$X_b111[1]X_b\cdots$
7	q_4	$X_b111[X_b]X_b\cdots$
8	-	$X_b111[X_b]X_b\cdots$

5. 考　　察

5.1 OSTP の安全性の検討

4.1 節の OSTP において，入力者 A と DFA 所有者 B の双方のプライバシが守られるかについて考察する．

5.1.1 DFA 所有者のオートマトンの秘匿性

A が B から受け取るデータは，各回の OT の実行で得る $N_p(\delta(N_{p-1}^{-1}(r_{p-1}), 0/1))$ および，最後の OT の実行で得る $F(\delta(N_{p-1}^{-1}(r_{p-1})))$ だけである．OT の性質によって，A は B の持つ選択されなかった他の値をいっさい復元することができない．また各回ごとに B は各状態に値 $\{0, 1, ..., n-1\}$ をランダムに割り当てているので，A はどの回とどの回に通った状態が同一の遷移を持つ状態であるかを判別することができない．よって A は自分の通った状態だけから B の持つ DFA の状態遷移図を構築することができないので，B の持つ DFA の秘匿性は保障される．

5.1.2 入力者の入力文字列の秘匿性

OT の性質により，B には A がどの遷移情報を選択したかを知ることができないので，A がインデックスとして使用する文字 w_p を知ることはできない．よって，A の入力文字列に関する文字数 $|w|$ を除く秘匿性は保障される．

5.2 OTMP の安全性の検討

4.2 節の OTMP において，入力者 A と TM 所有者 B の双方のプライバシが守られるかについて考察する．

5.2.1 TM 所有者の秘匿性

A が B から受け取る値は，各回の OT の実行で得る

$E_{K(p-1,q_i,X_j)}[(N_p(q_{i,j}), X_{i,j}, D_{i,j}, K(p, q_{i,j}, X_0), ..., K(p, q_{i,j}, X_m))]$

だけである．1-out-of-$(m+1)n$ OT の性質により，A は B の持つ他の値をいっさい復元することができない．また OSTP と同様に，各回ごとに B は各状態に値 $\{0, 1, ..., n-1\}$ をランダムに割り当てているので，A はどの回とどの回に通った状態が同一の遷移を持つ状態であるかを判別することができない．よって A は自身の得た情報だけから B の持つ TM の遷移関数 δ を決定することができないので，B の持つ TM の秘匿性は保障される．

5.2.2 入力者の入力文字列の秘匿性

OT の性質により，B には A がどの遷移情報を選択したかを知ることができないので，A がインデックスとして使用する文字 s_p を知ることはできない．ただし，TM 中にヘッドの移動方向のいずれかが 1 遷移にしか対応していない場合には，当該遷移を選んだという情報が B に知られてしまう[⋆1]．しかし A の全入力時に対して同様の現象が起こる可能性は少なく，また B がその現象を可能とするような TM を作成することは困難と考えられるので，A の文字列のすべてが B に知られる可能性は少ない．よって A の入力文字列に関する文字数 $|s|$ を除く秘匿性は保障される．

5.3 第三者からの攻撃

第三者 *Eve* が入力者 A の処理結果を得ようとする攻撃を考える．*Eve* が処理結果を得るには，(1) $A - B$ 間での各回の OT で A が使用するインデックスのうち，p 文字目の入力文字，または (2) 各回の OT で送受信される B の暗号文の内容，のいずれかを知る必要がある．しかし OT の性質により，A が使用する p 文字目の入力文字の値（各回の OT での

⋆1 p を共有せず，すべての場合で A が選択した遷移を秘匿可能なプロトコルについては，今後の課題として検討中である．

鍵の添え字に相当）を Eve は得られない．また B の暗号文についても OT の性質により，暗号文を得るための鍵を Eve は得られず，暗号文を得ることはできない．よって，Eve は A の処理結果について知ることができない．次に，Eve が DFA，TM の遷移情報を得ようとする攻撃を考える．Eve が遷移情報を得るには，$A-B$ 間での各回の OT で送受信される B の暗号文を復号する必要がある．しかし前述したように，Eve は暗号文を復号できないので，DFA の遷移情報は得られない．

5.4　OSTP と OTMP の能力の比較

OSTP で扱える処理の範囲は DFA で表現可能な言語の受理判定であり，OTMP で扱える処理の範囲は TM で表現可能な計算，言語受理判定である．すなわち計算能力では OSTP は OTMP に完全に包含される．しかし実行時における OT の回数を考えると，OSTP では入力文字列の文字数分だけ OT が行われるのに対し，OTMP では TM が停止するまでの入力回数分であり状態数と入力データ量の増加に従って増加する．よって OSTP で処理可能な範囲の受理判定であれば，OSTP を使用することで OTMP 使用時よりも少ない OT の実行回数で処理を実行できる．

5.5　関連研究との比較

文献 2) およびそれを発展させた文献 3) で対象としている Turing Machine Game の計算能力には背景理論からの制限[⋆1]がかかっており，本研究は TM で表現可能な処理であれば制限なく適用できるという点でより適用範囲が広いといえる．文献 4) で扱っている計算は任意の論理回路計算であるが，論理回路を TM 上で表現することで同様の論理回路計算を OTMP プロトコルを適用して実行することが可能となる．つまり 1 個の合成論理回路で表現される処理の範囲は TM で表現される処理（アルゴリズム）の範囲に包括されるので，本研究はより適用範囲が広いといえる．文献 5) で示されている Oblivious Keyword Search（OKS）プロトコルでは，検索語と辞書データとを秘匿したまま検索サービスを行っている．辞書データ中に存在するインデックスだけを受理する DFA を作成することが可能な程度のデータ量であれば，インデックスが辞書データ中に存在するかを判定する処理に関しては OSTP プロトコルを適用して実行できる．また文献 7) で示されているプロトコルでは，多次多変数多項式の計算を扱うことしかできない．多次多変数多項式の計算は TM 上で表現できるので，文献 7) のプロトコルで実行可能な計算については OTMP プロトコルを適用して実行することが可能である．

⋆1 入力やパーティ数に対して多項式時間の TM を仮定している．

6. 応　用　例

6.1　Oblivious State Transition Protocol の応用例

6.1.1　メールフィルタリングサービス

メールサーバに届いたメールをユーザのフィルタリング条件に基づいてフィルタリングするサービスを考える．ユーザはフィルタリング条件をフィルタリングサーバ FS に登録し，メールサーバ MS から FS にアクセスすることでフィルタリングを行う．ここでは FS を，MS を運営するプロバイダとは異なる組織が管理していると仮定する．この場合，FS 側では各ユーザが設定するフィルタリング条件は個人情報なので安易な提供は避けるべきであり，フィルタリングアルゴリズムが MS 側に漏洩する事態も避けなければならない．MS 側としてもユーザのメールの内容が FS 側に知られてしまうことは個人情報の漏洩となるため，防ぐ必要が生じる．またフィルタリング条件を正規表現とすると，フィルタリングアルゴリズムが正規表現を受理する DFA のアルゴリズムに帰着でき，上記のセキュリティ要件を満たすため OSTP を適用して実行することが可能になる．

6.1.2　フィルタリングサービスへの適用

本項では，FS と MS 間でのフィルタリングサービスを，「DFA 化したフィルタリング条件の受理アルゴリズムを OSTP を使用して実行するプログラムを実装する」ことによって秘匿したまま実行する手順（**図 3**）を述べる．

［要件］フィルタリングサーバ FS と，受信メールサーバ MS の 2 者が存在し，MS はユーザのメール w を持つ．プロトコルの実行後，MS はフィルタリング結果（フィルタリング条件を満たしたかどうか）を得る．その際，次の要件を満たす．

(1) MS は DFA 化したフィルタリングアルゴリズムの状態数以外の遷移関数のすべての情報を得ることができない．

(2) FS は MS が持つユーザのメールの内容 w とフィルタリング結果を得ることができない．

［適用手順］

(Step 1) OSTP を使用したプログラムの作成

FS はフィルタリングアルゴリズムに対応する DFA を作成し，その DFA を OSTP を使用して実行するプログラムを実装する．またユーザは各自のフィルタリング条件を正規表現に変換して FS に登録する．

(Step 2) プログラムの実行

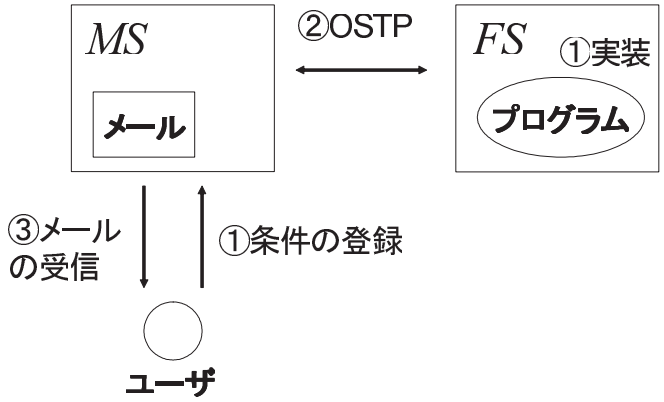

図 3　OSTP の応用例
Fig. 3　An application of OSTP.

MS は FS の所有する OSTP 適用プログラムをネットワーク上で実行する．なお，実行のタイミングはユーザがメール受信のため MUA（Mail User Agent）から MS にアクセスしたときとする．

(Step 3) フィルタリング結果の取得

(Step 2) の実行終了後，MS はフィルタリング結果を得て，ユーザにフィルタリング後のメールへのアクセスを許可する．

以上が適用例の手順である．

6.2　Oblivious Turing Machine Protocol の応用例

6.2.1　電子透かし埋め込み処理

画像に画素置換型の電子透かしを埋め込むサービスを考える．ユーザは埋め込み処理をしたい画像と文字列などの埋め込みデータを持ち，埋め込みプログラムを持つサーバに処理を依頼する．このとき，サーバ側では埋め込みアルゴリズムをユーザから秘匿する必要があり，ユーザには入力する画像と埋め込み用データをサーバから秘匿したいという要求が存在する．ここで，画素置換型の電子透かしでは個々の画素に直接ビット値を埋め込む．よって TM のテープに画像の各ビット値と埋め込みデータとを記入することで，OTMP を適用して上記のセキュリティ要件を満たすため実行することが可能になる．

6.2.2　電子透かし埋め込み処理への適用

本項では，画素置換型電子透かしの埋め込みサービスを，「TM 化した埋め込みアルゴリズムを OTMP を使用して実行するプログラムを実装する」ことによって秘匿したまま実行する手順（**図 4**）を述べる．

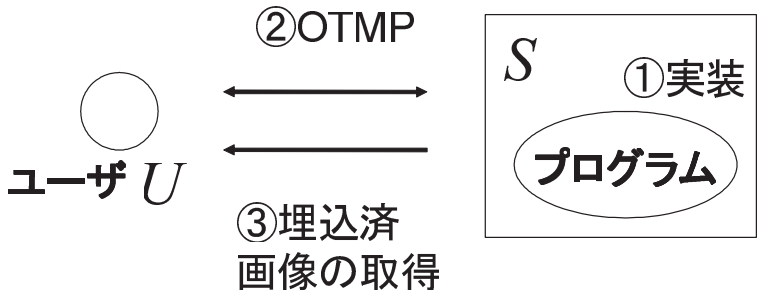

図 4　OTMP の応用例
Fig. 4　An application of OTMP.

［要件］埋め込みサービス提供者 S とユーザ U が存在し，U は画像と埋め込み用データを持つ．プロトコルの実行後，U は電子透かしが埋め込まれた画像を得る．その際，次の要件を満たす．

(1) U は TM 化した埋め込みアルゴリズムの状態数以外の遷移関数のすべての情報を得ることができない．

(2) S は U が持つ画像，埋め込み用データと埋め込み済画像を得ることができない．

［適用手順］

(Step 1) OTMP を使用したプログラムの作成

S は埋め込みアルゴリズムに対応する TM を作成し，その TM を OTMP を使用して実行するプログラムを実装する．

(Step 2) プログラムの実行

U は S の所有する OTMP 適用プログラムをネットワーク上で実行する．

(Step 3) 透かしが埋め込まれた画像の取得

(Step 2) の実行終了後，U は埋め込み済み画像を得る．

以上が適用例の手順である．

6.3　入力者が複数の場合の Oblivious Turing Machine Protocol の応用例

6.3.1　グループ共有鍵

グループ G は n 名のユーザからなり，各ユーザ U_i はユーザに固有の情報 N_i を持っている．暗号鍵 E_G はグループ内で単一で，$N_1, ..., N_n$ の値から作成される．復号鍵はユーザ

ごとに異なるものが存在し，U_i 用の復号鍵 D_i は E_G に N_i の値を反映させたものである．復号鍵を複数持つグループ共有鍵方式では，グループに参加していない第三者による暗号内容の盗聴，改竄が困難になる．また，復号鍵の漏洩時に誰が漏洩させたかの追跡が容易にもなるため，有用な方式と考えられる．このグループ暗号鍵作成アルゴリズムをサービスとしてグループ外の提供者が提供することにより，グループを離脱した元ユーザが作成アルゴリズムを使用して暗号内容の盗聴，改竄を行うことを防止することが可能となる．

6.3.2　OTMP プロトコルの拡張

前項のグループ暗号鍵作成アルゴリズムに OTMP プロトコルを適用させることで，ユーザから作成アルゴリズムを秘匿し，サービス提供者から固有情報を秘匿させることが両立可能となる．しかし，前述の作成アルゴリズムでは各ユーザが別々の入力値を持っているので，OTMP を単純に適用することはできない．よって本項では，OTMP プロトコルの自然な拡張として入力者が複数存在する場合のプロトコルについて述べる．

6.3.2.1　要件

拡張された OTMP プロトコルでは，入力者 $A_1, ..., A_t$ と TM 所有者 B が存在し，$A_1, ..., A_t$ は TM への各入力値を持っている．プロトコルの実行後，$A_1, ..., A_t$ は B の持つ TM で計算した結果を得る．その際，次の要件を満たす．

(1) 入力者 $A_1, ..., A_t$ は TM の状態数以外の遷移関数の情報に関してまったく分からない．

(2) TM 所有者 B は，入力者 $A_1, ..., A_t$ の入力値と計算結果に関してまったく分からない．

6.3.2.2　プロトコル

このプロトコルでは，一般的な TM の代わりに多テープの TM を使用する．多テープ TM には複数のテープが存在し，これによって複数の入力値を 1 度に TM で扱うことが可能となる．TM M の定義に関しては 4.2 節のものと同様である．初期化，プロトコルについてもほとんど 2 者間 OTMP と同様であるので，以下では相違点について述べる．

［初期化］$A_1, ..., A_t$ は各入力値 $s_1, ..., s_t \in \Sigma^*$ を各テープの左端からプロトコル実行に必要な数の空白も含めて記入する．s_k（$k = 1, ..., t$）の文字数を $|s_k|$，s_k の p 番目の文字を s_{kp}（$0 < p \leq |s|$）とする．また M の状態数 n については事前に $A_1, ..., A_t$ に知らせておく．なお，B の作業は 2 者間 OTMP と同様である．

［プロトコル本体］

(Step 1) から (Step 3) は 2 者間 OTMP と同様に行う．ただし，(Step 4) で行う各 A_k と B との OT は互いに独立に行われるので，B は (Step 2) での $N_p(q_i)$，(Step 3) での暗号文を各 A_k に対して別々に作成する．

(Step 4) 各 A_k（$k = 1, ..., t$）は B とそれぞれ 1-out-of-$(m+1)n$ OT を実行して，上記 (Step 3) の暗号文の 1 つを得る．各 A_k はインデックスが (r_{p-1}, s_p) である値を選択し得る．取得後，各 A_k は持っている鍵のうち，$K(p-1, N_{p-1}^{-1}(r_{p-1}), s_p)$ を使用して暗号文を復号し各々の値を得る．ここで各 A_k は得た $N_p(q_{i,j})$ をそれぞれ新たに r_p とおく．そして各テープの該当位置の値をそれぞれ得た $X_{i,j}$ で書き換える．また復号した値が n_{undef} であった場合には，遷移関数 δ の値が定義されていないのでその入力者はプロトコルを終了する．すべての入力者のプロトコルが終了した後，入力者は計算結果を得る．なお，計算結果の書かれているテープがどの入力者が所有しているかについては，事前に B から知らされているものとする．

(Step 5) 各 A_k は (Step 4) で得た $D_{i,j}$ の値から B に p の値の増減を伝え，(Step 2) 以降の処理を繰り返す．

以上が，拡張された OTMP プロトコルである．

6.3.3　グループ暗号鍵作成サービスへの適用

本項ではグループ暗号鍵作成サービス提供者とグループ内ユーザ間で，グループ暗号鍵の作成を，「TM 化した作成アルゴリズムを拡張 OTMP を使用して実行するプログラムを実装する」ことによって秘匿したまま実行する手順（**図 5**）を述べる．

［要件］グループ内のユーザ $U_1, ..., U_n$ と，作成サービス提供者 S が存在し，$U_1, ..., U_n$ は固有の情報 $N_1, ..., N_n$ を持つ．プロトコルの実行後，$U_1, ..., U_n$ はグループ暗号鍵を得る．その際，次の要件を満たす．

(1) ユーザ $U_1, ..., U_n$ は TM 化した作成アルゴリズムの状態数以外の遷移関数のすべて

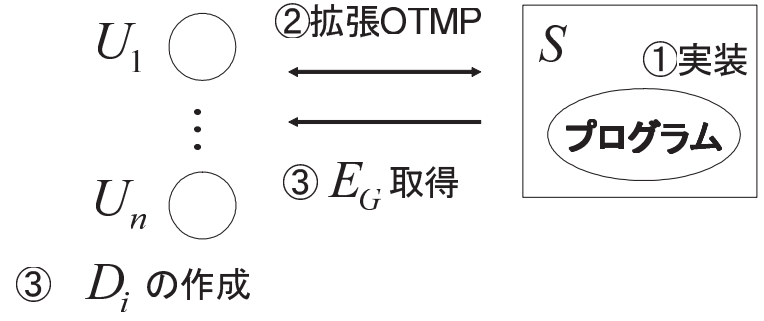

図 5　拡張 OTMP の応用例
Fig. 5　An application of extended OTMP.

の情報を得ることができない．

(2) サービス提供者 S はユーザ $U_1, ..., U_n$ の固有情報 $N_1, ..., N_n$ とグループ暗号鍵についてのすべての情報を得ることができない．

［適用手順］

(Step 1) 拡張 OTMP を使用したプログラムの作成

サービス提供者はグループ暗号鍵作成アルゴリズムに対応するチューリングマシンを作成し，そのチューリングマシンを拡張 OTMP を使用して実行するプログラムを実装する．

(Step 2) プログラムの実行

ユーザはサービス提供者の所有する拡張 OTMP 適用プログラムをネットワーク上で実行する．

(Step 3) グループ暗号鍵の共有

(Step 2) の実行終了後，ユーザはグループ暗号鍵を得る．また，各ユーザはグループ暗号鍵と固有情報からユーザごとの復号鍵を作成する．

以上が適用例の手順である．

6.4 応用例に関する考察

本章であげた応用例の適用において既存研究[4] との比較を行うと，文献 4) では処理を 1 個の合成論理回路に変換する必要がある．メールフィルタリング処理，画素置換型電子透かしの埋め込み処理，グループ暗号鍵作成処理はいずれも 1 個の合成論理回路として構成することは困難であり，提案手法の OSTP，OTMP では，そのような複雑な回路への変換が必要なく，簡潔で効率の良い実装が可能である．

7. ま と め

本稿では，状態遷移アルゴリズムについて，その遷移関数と入力値となる文字列の双方を秘匿したまま受理判定や計算などの処理を行う OSTP，OTMP プロトコルを提案した．このプロトコルにより状態遷移アルゴリズムを所有する側は実行にあたって何も情報が得られず，入力値を所有する側は処理結果以外のいっさいの情報が得られないという要件を満たすことが可能になった．

また，OSTP，OTMP が有用性を持つ具体的な例として，サービス提供者側のサービスアルゴリズムと利用者側の入力情報との双方を保護可能な安全なメールフィルタリングサービス，電子透かし埋め込みサービス，グループ共有鍵作成サービスの構成をそれぞれ示した．

今後の課題として，今回提案した OSTP，OTMP の通信コストの詳細な評価，実装を通してのさらなる応用例の検討などがある．

参 考 文 献

1) Naor, M. and Pinkas, B.: Oblivious transfer and polynomial evaluation, *Proc. 31st Symp. on Theory of Computer Science*, pp.245–254 (1999).
2) Yao, A.: How to generate and exchange secrets, *Proc. FOCS*, pp.162–167 (1986).
3) Goldreich, O., Micali, S. and Wigderson, A.: How to play any mental game, or a completeness theorem for protocols with honest majority, *Proc. STOC*, pp.218–229 (1987).
4) 千田浩司，谷口展郎，山本　剛，岡崎聖人，塩野入理，金井　敦：エルガマル暗号に基づく秘匿回路計算の実装と応用，コンピュータセキュリティシンポジウム 2005（CSS 2005），pp.475–480 (2005).
5) Ogata, W. and Kurosawa, K.: Oblivious Keyword Search, *Journal of Complexity*, Vol.20, pp.356–371 (2004).
6) Naor, M. and Pinkas, B.: Oblivious Polynomial Evaluation, *SIAM* (*Society for Industrial and Applied Mathematics*) *J. Comput.*, Vol.35, No.5, pp.1254–1281 (2006).
7) 小瀬木浩昭，平原耕一，大矢健太，折笠大典，武田正之：Multi-Variable Oblivious Polynomial Evaluation，第 5 回情報科学技術フォーラム（FIT2006），pp.266–268 (2006).

(平成 19 年 11 月 29 日受付)
(平成 20 年 6 月 3 日採録)

平原　耕一（正会員）

2006 年東京理科大学理工学部情報科学科卒業．2008 年同大学大学院修士課程修了．同年株式会社富士通エフサス入社．

渡邉　昭宏（学生会員）

1984 年生．2007 年東京理科大学理工学部情報科学科卒業．現在，同大学大学院修士課程 2 年生．情報セキュリティの暗号プロトコル分野の研究に従事．

武田　正之（正会員）

1977 年東京理科大学理工学部電気工学科卒業．1982 年東京工業大学大学院博士課程（電子物理工学専攻）修了．同年東京理科大学理工学部情報科学科助手となり，現在，同大学教授．工学博士．著書（共著）に『Prolog とその応用 2』，総研出版（1985 年）等がある．プログラミング言語の意味論，並列・分散システム，知識情報処理に興味を持つ．昭和 57 年度情報処理学会論文賞受賞．電子情報通信学会，日本ソフトウェア科学会，人工知能学会，ACM 各会員．