準同型性暗号を用いた拡張文字列の秘匿パターン照合

原田 弘毅 †　　笹川 裕人 ‡　　有村 博紀 ‡　　佐久間 淳 †

† 筑波大学 大学院システム情報工学研究科
305-8573 茨城県つくば市天王台 1-1-1
h.hiroki@mdl.cs.tsukuba.ac.jp, jun@cs.tsukuba.ac.jp

‡ 北海道大学 大学院情報科学研究科
060-0814 北海道札幌市北区北 9 条西 1 4
{sasakawa, arim}@ist.hokudai.ac.jp

あらまし 本研究では、パターンと文字列の両方が秘密情報であるときに、互いの情報を秘匿したまま、文字列上でパターンの検索を行うプロトコルを提案する。本方式の用途として、個人ゲノムにおける遺伝子検索や、顧客の購買パターン検出などが挙げられる。従来の秘匿有限オートマトン評価では、本稿で扱う拡張文字列パターンのクラスに対して、一般に決定性有限オートマトンの状態数が指数的に爆発する問題がある。提案方式では、非決定性有限オートマトンの模倣に基づくパターン照合手法の一つであるビット並列照合手法に基づき、準同型性暗号を用いた二者間の秘密計算によって、状態数の爆発を回避しつつ、照合を効率良く実行する。

Secure Pattern Matching Using Homomorphic Encryption for Extended String Patterns

Hiroki Harada†　　Hirohito Sasakawa‡　　Hiroki Arimura ‡　　Jun Sakuma †

†Collage of Information Science, University of Tsukuba
Tennodai, Tsukuba, Ibaraki 1-1-1, JAPAN
h.hiroki@mdl.cs.tsukuba.ac.jp, jun@cs.tsukuba.ac.jp

‡Graduate School of Information Science and Technology, Hokkaido University
kita 14 nishi 9 Kita, Sapporo, Hokkaido 060-0814, Japan
{sasakawa, arim}@ist.hokudai.ac.jp

Abstract In this paper, we propose a protocol for secure pattern matching between two parties which have a string and pattern, respectively. This protocol searches the pattern on the string without revealing their information each other. It is applicable to DNA search in personal genomes and detection of purchase patterns of customers. The existing approach with oblivious automata evaluation has a problem of exponential state explosion for the class of extended string patterns in general. To avoid this problem, our method extends bit-parallel pattern matching, which simulates a non-deterministic finite automaton, to secure multiparty computation using homomorphic encryption between two parties.

1 はじめに

近年の個人情報保護の必要性の高まりにより, 保護が必要なデータを開示せずに処理を行う秘密計算が注目されている. この秘密計算を利用して, 文字列から特定のパターンを検出する秘密パターン照合は, 広範な応用が期待される重要なタスクである. 秘密パターン照合の 2 つのシナリオを紹介する.

例 1. 特定の疾患と関連のある塩基配列パターンを保持するサービス提供者が, 個人が持つ DNA を用いて疾患の事前診断を行う例を考える. この場合, サービス提供者にとって塩基配列パターンを公開することはサービスの運営上好ましくなく, また個人が持つ DNA をサービス提供者に公開することはプライバシ保護の観点で問題がある. 互いの持つデータを互いに秘匿しつつ, DNA 所有者は自身の DNA に, サービス提供者が持つパターンが含まれるかを照合したい.

例 2. 小売業者は個人の商品の購買履歴を保持している. Web 広告会社は, ある特定の購買パターンを示した個人についてある種の広告を Web に表示させるターゲティング広告を考える. この場合, 小売業者にとって購買履歴を Web 広告会社に公開することはプライバシ保護の観点で問題があり, また Web 広告会社にとって, 広告出稿のトリガとなる購買パターンを公開することはサービスの運営上好ましくない. 互いの持つデータを互いに秘匿しつつ, Web 広告会社は自身が設計した購買パターンに, 小売業者が持つどの顧客の購買履歴にいつマッチするかを求めたい.

上記の二つの例は, 秘密の文字列と秘密のパターンの二種類の入力をとる秘密計算として定式化される. 以降, パターンを保持するパーティを**パターン保有者**, 文字列を保持するパーティを**文字列保有者**と呼ぶ. これらは以下の 2 つの点で異なる. まず, 秘密計算の結果, 例 1 では文字列保有者が照合結果を受け取り, 例 2 ではパターン保有者が照合結果を受け取っている. また例 1 では評価時点で文字列 (=DNA) は完結しており, パターン照合の結果はクエリに対する応答として受け取ることができるが, 例 2 では文字列 (=購買履歴) は評価時点では完結しておらず, パターン照合の結果はストリームに対するイベント検出として受け取る必要がある. 本稿では前者を**オフライン秘匿パターン照合**, 後者を**オンライン秘匿パターン照合**と呼ぶ. またいずれの例も, 単なるパターン照合では対応できず, 文字列クラスや部分繰り返しを含むパターンへの照合が必要である.

本稿では特にプライバシ保護が必要な情報を含む文字列のストリームに対し, パターン保有者が照合したいパターンを秘匿しつつ, 合致するパターンの検出を行うオンライン秘匿パターン照合のアルゴリズムを提案する.

関連研究 秘匿パターン照合の既存の解法として, 紛失オートマトン計算 (Oblivious Automata Evaluation, OAE) の解法が複数提案されている (表 1). OAE ではパターン保有者は秘密のパターンをオートマトンに変換し, オートマトンと入力文字列を互いに秘密にしたまま照合を行い, 文字列保有者が照合結果を受け取ることができる.

Troncoso ら [7] は加法準同型性暗号と紛失通信 (Oblivious Transfer) を利用し, 決定性有限オートマトン (Deterministic Finite Automaton, DFA) の遷移状態をランダマイズした状態で更新する OAE 手法を提案した. この手法は更新処理に, 文字列の文字数 n と文字列が含みうるアルファベット数 $|\Sigma|$ の積に比例するコストを必要とする. このため, $|\Sigma|$ の小さい DNA の計算等には適するが, n,$|\Sigma|$ が共に大きい日本語テキストや購買ログ等の計算には適さない (表 1, 2 列目).

Frikken [2] は Troncoso らの手法に Yao の秘匿関数計算 [9] を組み込んだ OAE 手法を提案している. 渡辺ら [10] は Troncoso らの手法を基に, 暗号文領域検査が可能な暗号方式 [3] を用いて定数回の通信で紛失オートマトン計算を行う手法を提案している. これらの手法は定数ラウンドで計算可能であるが, 事前処理のコストが $n \times |\Sigma|$ に比例する (表 1, 3,4 列目).

また, いずれの提案手法もパターン照合に DFA の評価を想定しており, 正規表現を含むパターンの照合を行う場合, まず非決定性有限オートマトン (Non-determinisitic Finite Automaton, NFA) を構築し, これを DFA に変換し処理を行っている. 一部の正規表現クラスを表す NFA は, DFA に変換すると状態数が指数的に増加する場合があり, 既存手法は計算が多項式時間で

表 1: 既存手法と提案手法の性能比較. n は文字列長, Σ はアルファベット集合, 既存手法の m は DFA の状態数 (対象オートマトンが NFA の場合は DFA に変換する), 提案手法の m は NFA の状態数, PH はパターン保有者, SH は文字列保有者, CM は完全合致, CC は文字種, ILG は無限長ギャップを示す.

		Troncoso+ [7]	Frikken [2]	渡邊ら [10]	SPM 1	SPM 2
照合結果の取得者		PH	PH	PH	SH or PH	SH or PH
事前処理	時間計算量	$O(m)$	$O(nm\|\Sigma\|)$	$O(n\|\Sigma\|)$	$O(m\|\Sigma\|)$	$O(m\|\Sigma\|)$
	通信計算量	$O(\|\Sigma\|)$	$O(nm\|\Sigma\|)$	$O(n(m+\|\Sigma\|))$	$O(m\|\Sigma\|)$	$O(m\|\Sigma\|)$
更新処理	時間計算量	$O(m\|\Sigma\|)$	$O(mn)$	$O(n)$	$O(m)$	$O(m)$
	通信計算量	$O(m+\|\Sigma\|)$	$O(1)$	$O(1)$	$O(1)$	$O(m)$
ラウンド計算量		$O(n)$	$O(1)$	$O(1)$	$O(1)$	$O(n)$
オンライン対応パターンクラス		非対応 DFA	非対応 DFA	非対応 DFA	対応 CM + CC	対応 CM + CC + ILG

終了する保証がない. またいずれの既存手法もオンラインパターン照合には対応していない. 後に詳述するが, オンラインパターン照合はオフラインパターン照合を含むより一般的な問題である.

本研究の貢献 本研究の貢献は, 定数文字, 文字種, 無限長ギャップを含む NFA について、オンライン秘匿パターン照合を多項式時間で扱うことができる 2 つのアルゴリズムの提案である.

- SPM 1: 定数文字, 文字種を含む NFA に対し, $O(1)$ ラウンドで照合を行うオンライン秘匿パターン照合プロトコル
- SPM 2: 定数文字, 文字種, 無限長ギャップによる非限定繰り返しを含む NFA に対し, $O(n)$ ラウンドで照合を行うオンライン秘匿パターン照合プロトコル

提案手法は NFA を直接評価するため, 文字種や無限長ギャップを含むオートマトンに対しても常に多項式時間で処理可能である. また時間計算量・通信計算量が, 事前処理では n, 更新処理では $|\Sigma|$ にそれぞれ依存しないため, n,$|\Sigma|$ が共に大きい日本語テキストや購買ログ等に対しても効率的なパターン照合が可能である. またストリームデータなど無限長の文字列に対する秘匿パターン照合にも対応可能なオンラインアルゴリズムを達成している.

本稿の構成を示す. 第二章では本稿が扱う問題の定義を行う. 第三章では提案手法の基礎となる Shift-OR アルゴリズムを紹介する. 第四章では Shift-OR アルゴリズムを用いた提案手法である秘匿パターン照合プロトコルを提案する. 第五章では結論を示す.

2 問題定義

2.1 パターンクラスとパターン照合問題

パターン照合問題を, 文献 [4] に従い導入する. Σ を文字集合とする. A^* で, 集合 $A \subseteq \Sigma$ の要素の有限列全体を表す. パターン P は正規表現の部分クラスである. **拡張文字列パターン**は, 以下に定義する直線状の正規表現 $P = \beta[1]\cdots\beta[m]$ $(m \geq 0)$ であり, その言語は各要素の言語の連接 $L(P) = L(\beta[1])\cdots L(\beta[m])$ である [4]. ここに, 各 $1 \leq i \leq m$ に対して, 要素 $\beta[i]$ とその言語 $L(\beta[i])$ は次の (i)–(iii) のどれかである：(i) 完全合致 (CM) $\beta[i] = a \in \Sigma$, (ii) 文字種 (CC) $\beta[i] = A = [abc\cdots] \subseteq \Sigma$, または, (iii) 無限回繰り返し $\beta[i] = A^*$ である. ここに,$a \in \Sigma$ かつ $A \subseteq \Sigma$ とする. P が (i) のみで構成される場合,P を**文字列パターン**（完全合致）と呼ぶ. 上記の (ii) と (iii) で $A = \Sigma$ のときは, それぞれ，ワイルドカードおよび無限長ギャップ (ILG) とよばれる [4]. 本稿では, 定数文字と文字種を含むパターン (CM+CC) と, 定数文字と文字種と無限長ギャップ (CM+CC+ILG) の両方を含むパターンの二つの部分クラスを考察する.

完全合致と文字クラスをもつパターン $P = a[ab]^m$ に対して，その照合を表す NFA は $\Sigma =$

$\{a, b\}$ のとき，次節より $m+2$ 個の状態で実現可能だが，等価な DFA は 2^{m+1} 個の状態を必要とすることが知られている [6].

次にパターンの照合を定義する．入力文字列 $T = T[1] \cdots T[n] \in \Sigma^n$ $(n \geq 0)$ において，P が末尾出現位置（または単に位置）$1 \leq p \leq n$ に**出現する**とは，T の部分文字列 $u, v, w \in \Sigma^*$ が存在して，(i) $T = uvw$，かつ (ii) $p = |uv|$ を (iii) $v \in L(P)$ が成立することをいう．

まず，**オフラインのパターン照合タスク**を次のように定義する．入力としてサイズ m のパターン P と長さ n の文字列 T を仮定する．

- OFFLINE MATCH: テキスト T 中のパターン P の出現の有無を,YES/NO で答える．
- COUNT: テキスト T 中のパターン P の出現位置の個数 $count(P, T) \geq 0$ を返す．
- LOCATE: テキスト T 中のパターン P のすべての出現位置を返す．

続いて，**オンラインのパターン照合タスク**を次のように定義する．入力として，パターン P と無限長の文字ストリーム $T = T[1]\,T[2] \cdots \in \Sigma^\infty$ を仮定し，時点 $i \geq 1$ において各文字 $T[i] \in \Sigma$ が順に与えられるとする．このとき, ONLINE MATCH タスクとは，各時点 $i = 1, 2, \ldots$ において，パターン P が現在の入力文字列 $T[1, i] = T[1] \cdots T[i]$ において，末尾出現位置 i をもつかを YES/NO で返す問題である．

2.2 オンライン秘密パターン照合問題

本稿で扱う問題は，ONLINE MATCH タスクの秘密計算である．この問題では 2 つのパーティが存在する．1 つは文字列を保持する**文字列保有者**である．文字列保有者は有限文字集合 Σ から生成した長さ n の文字列 $T \in \Sigma^n$ を持つ．もう 1 つはパターンを持つ**パターン保有者**である．パターン保有者は同じ有限文字集合 Σ から生成した長さ m のパターン $P \in \Sigma^m$ を持つ．このとき，本稿で対象とする問題は以下のように定義される．

Definition 1. *(パターン保有者 (文字列保有者) によるオンライン秘密パターン照合)* 文字列保有者が文字列 $T[1, i-1] \in \Sigma^n$ を，パターン保有者がパターン $P \in \Sigma^m$ を持つ．文字列保有者が時点 i において文字 $T[i]$ を受け取り，プロトコル実行後，パターン保有者 *(文字列保有者)* は，パターン P が文字列 $T[1, i]$ の末尾に出現したか否かを得るがそれ以外は何も得ない．文字列保有者 *(パターン保有者)* は何も得ない．

3 パターン照合アルゴリズム

本節では，ストリームデータに対するパターン照合アルゴリズムとして，ビット並列手法を用いた照合アルゴリズム Shift-OR 法を紹介し，加法準同型暗号を用いた秘密計算へ拡張する前段階として,Shift-OR の加法のみを用いた変更についても説明する．

3.1 ビット並列パターン照合手法

ビット並列パターン照合手法とは，ビット演算と算術演算のレジスタ内並列性を利用し，パターン照合を高速化する手法であり，最も基本的なものに Shift-OR 法 [1,8] がある．Shift-OR 法は完全一致と文字種を含んだパターンに対する照合アルゴリズムであり，入力されたパターン P からそれを受理する NFA を構築し，文字列 T 上でその遷移を模倣することで照合を行う．

ビット並列パターン照合手法を準同型暗号上の秘密計算へ拡張するため，配列と加法のみを用いた手法を与える．なお，本稿では Shift-OR 法を配列上で行うアルゴリズムを配列 Shift-OR 法と呼び，3.1.3 節の無限長ギャップを含むパターンへの拡張を行ったアルゴリズムを拡張配列 Shift-OR 法と呼ぶ．

3.1.1 文字列パターンの照合

ビット並列手法では二進数の値に対しビット演算を用いて処理を行うのに対し，配列 Shift-OR では，二進数の各桁を配列とみなし，算術加算を用いて状態集合の更新を行う．以下に照合の手順を与える．ここで,$1 \leq j \leq m$ とする．

1. Σ の各要素 σ について，P から式 1 によってマスク配列 $M_\sigma \in \{0, 1\}^m$ を生成する．

$$M_\sigma[j] = \begin{cases} 0 & (P[j] = \sigma) \\ 1 & (\text{otherwise}) \end{cases} \quad (1)$$

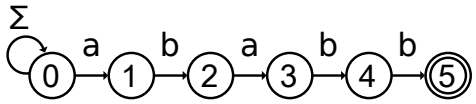

図 1: $P = ababb$ を受理する NFA.

表 2: $P = ababb$ のとき配列 Shift-OR 法で前処理時に生成する文字マスク.

状態番号	1	2	3	4	5
M_a	0	1	0	1	1
M_b	1	0	1	0	0

2. 配列 S_0 を $S_0[0] = 0$, $S_0[j] = 1, j \in \{1, ..., m\}$ に初期化する. ここで, 配列 $S_i[j]$ は文字列の i 文字目を読み込んだ際の j 番目の遷移状態を指す. j 番目の遷移状態は $S_i[j] = 0$ ならば active, $S_i[j] \neq 0$ ならば inactive である.
3. 文字列の各文字を読み込み, 式 2 により遷移状態を更新する. なお, 配列 S_i について, $S_i[0] = 0$ とする.

$$S_i[j] = S_{i-1}[j-1] + M_{T[i]}[j] \qquad (2)$$

4. 各 i について, $S_i[m]$ が active ならば i 文字目がマッチしたパターンの終端であるとみなし, これを出力する.

配列 Shift-OR 法による $P = ababb, T = abababb$ での照合の具体例を表 3 に示す. 図 1 は, 拡張 Shift-OR 法が遷移を模倣する NFA である.

1. P と Σ から, 文字マスク $M_a = (0, 1, 0, 1, 1)$, $M_b = (1, 0, 1, 0, 0)$(表 2) を生成する.
2. $S_0[0] = 0$, $S_0[j] = 1 (j \in \{1, ..., 5\})$ に初期化し, 照合を始める.
3. 入力 $T[1] = a$ に対し, 式 2 により S_1 を生成する. 以降各 i に対し S_i を生成する.
4. $i = 7$ において, $S_i[5] = 0$(active) を検出し, 7 文字目がマッチしたパターンの終端であることがわかる.

3.1.2 文字種を含むパターンへの拡張

定数文字 a による状態遷移では, 特定の文字 a の入力によりある状態に遷移する. 一方, 文字種 $A \subseteq \Sigma$ による状態遷移では, 文字種に含まれる任意の文字 $a \in A$ の入力により, 次の状態に遷移する. この拡張は, P の ℓ 番目の要素に文字種 A を持つとき, 各文字 $\sigma \in \Sigma$ に対して, マスク配列を式 3 により生成する.

$$M_\sigma[\ell] = \begin{cases} 0 & (P[\ell] = \sigma \in A) \\ 1 & (\text{otherwise}) \end{cases} \qquad (3)$$

表 3: $P = ababb$ と, $T = abababb$ のときの配列 Shift-OR の動作. $i = 5$ では $S_i[5] \neq 0$(inactive), $i = 7$ では $S_7[5] = 0$(active) である.

Cycle i	Input $T[i]$	状態集合					
		0	1	2	3	4	5
0	/	**0**	1	1	1	1	1
1	a	0	**0**	2	1	2	2
2	b	**0**	1	**0**	3	1	2
3	a	0	**0**	2	**0**	4	2
4	b	0	1	**0**	3	**0**	4
5	a	0	0	2	**0**	4	1
6	b	0	1	0	3	**0**	4
7	b	0	1	1	1	3	**0**

表 4: NFA の各状態 j におけるセルフループ遷移時の出力.

		$S_{i-1}[j]$	
		inactive	active
$S_i[j]$	inactive	inactive	if $j \in L$: active if $j \notin L$: inactive
	active	active	active

3.1.3 無限長ギャップを含むパターンへの拡張

無限長ギャップは, パターンから生成される NFA において, 現在の状態への遷移（セルフループ）によって実現される. アルゴリズムは, セルフループが存在する状態番号を要素に持つ集合 L を生成する. セルフループによる遷移は, 状態配列 S_i を表 4 に示す関係に基づいて更新することで実現できる. 表 4 の出力は, 式 2 により文字遷移を実行した後, 式 4 の処理によって得られる.

$$S_i[j] = \begin{cases} S_i[j] \cdot S_{i-1}[j] & (j \in L) \\ S_i[j] & (\text{otherwise}) \end{cases} \qquad (4)$$

式 4 は, ループのある状態については, 文字遷移以前の状態との積をとることでセルフループ遷移を実現し, ループのない状態に対しては, 何も行わない.

4 秘密計算による Shift-OR 法

本章では配列 Shift-OR 法, 拡張配列 Shift-OR 法の秘密計算への拡張である, 秘密配列 Shift-OR 法 (4.2 節), 秘密拡張配列 Shift-OR 法 (4.3 節) の 2 つの秘密パターン照合プロトコルを提案する. 以降では平文 A に対し, $\bar{A} = \mathrm{Enc}_{\mathrm{pk}}(A)$ と表記する. また, パターン保有者をパターン保有者, 文字列保有者を SH と表記する. $\in_r$ は対象の集合から要素をランダムに取得する演算子である.

4.1 加法準同型性公開鍵暗号

加法準同型性公開鍵暗号は, 同一の公開鍵で暗号化された暗号文同士の積が, 対応する平文同士の和の暗号文となる性質を持つ. この性質を持つ暗号系は, Paillier 暗号 [5] などが知られている. 以降では, 秘密鍵を sk, 公開鍵を pk と表記する. また, 暗号処理の記法として, $c = \mathrm{Enc}_{\mathrm{pk}}(t; r)$ を暗号化, $t = \mathrm{Dec}_{\mathrm{sk}}(c)$ を復号の処理とする.

加法準同型性公開鍵暗号は次の性質を満たす.

$$\mathrm{Enc}_{\mathrm{pk}}(t_1; r_1) \circ \mathrm{Enc}_{\mathrm{pk}}(t_2; r_2) = \mathrm{Enc}_{\mathrm{pk}}(t_1 + t_2; r_1 + r_2),$$
$$\mathrm{Enc}_{\mathrm{pk}}(t_1; r_1)^{t_2} = \mathrm{Enc}_{\mathrm{pk}}(t_1 t_2; r_1 r_2)$$

$\circ$ を暗号同士の積の演算子とし, t_1, t_2 は平文, r_1, r_2 は乱数を表す. r_1, r_2 は暗号の安全性を保持するために, 暗号化ごとに変更する. 以降では簡単のため, 乱数 r_1, r_2 の表記を省略し, 暗号化, 復号の処理を $\mathrm{Enc}_{\mathrm{pk}}(t)$,$\mathrm{Dec}_{\mathrm{sk}}(c)$ と表す. また, $c = \mathrm{Enc}_{\mathrm{pk}}(t) \circ \mathrm{Enc}_{\mathrm{pk}}(0)$ により, 同一の平文を持つ異なる暗号文を生成することができる (これを再暗号化と呼ぶ).

4.2 秘密配列 Shift-OR 法

まず, 配列 Shift-OR 法を秘密計算に拡張したプロトコルを Algorithm 1 に示す. このプロトコルでは加法準同型性を利用し, Shift-OR における $S_i[j]$ の更新を暗号化した状態で処理する. このプロトコルは定数文字, 文字種で構成されたパターンを照合可能である.

Algorithm 1 では, 1-3 行目で鍵交換とパターン保有者, 文字列保有者の事前処理を行う. 5 行目では, Shift-OR の更新処理 (式 2) を加法準同型性を利用して暗号上で行う. この処理を Algorithm 2 に示す. 6 行目では照合結果をパターン保有者, または文字列保有者に出力する処理を行う. この処理を Algorithm3 に示す.

Algorithm 1 秘密配列 Shift-OR 法

-PH's input: $P \in \Sigma$, $\Sigma = \{w_1, ..., w_{|\Sigma|}\}$
-PH's output: Γ^{PH}
-SH's input: $T \in \Sigma$, $\Sigma = \{w_1, ..., w_{|\Sigma|}\}$
-SH's output: Γ^{SH}
1: PH は秘密鍵 sk, 公開鍵 pk を生成し, サーバに pk を送信する.
2: PH は各 $w_i \in \Sigma$ について, マスクビット配列 $M_{w_i} \in \{0,1\}^m$ を生成し, 各要素を暗号化し SH に送信する.
3: SH は $\bar{S}_0[0] = \mathrm{Enc}_{\mathrm{pk}}(0)$, $\bar{S}_0[j] = \mathrm{Enc}_{\mathrm{pk}}(1), j \in \{1, ..., m\}$ により配列 $\bar{S}_0$ を生成し, 次の演算を行う.
4: **for** $i = 1$ to n **do**
5: SH は Algorithm 2 により S_i を更新する.
6: Algorithm 3 により, PH は $\Gamma^{\mathrm{PH}}[i]$ を, SH は $\Gamma^{\mathrm{SH}}[i]$ を出力する.
7: **end for**

Algorithm 2 秘密配列 Shift-OR の状態更新処理

-input: $T[i]$, $\bar{M}_{T[i]}$, $\bar{S}_{i-1}$
-output: $\bar{S}_i$
1: $\bar{S}_i[0] = \mathrm{Enc}_{\mathrm{pk}}(0)$ とする.
2: **for** $j = 1$ to m **do**
3: SH は式 5 により S_i を更新する.

$$\bar{S}_i[j] = \bar{S}_{i-1}[j-1] \circ \bar{M}_{t_i}[j] \quad (5)$$

4: **end for**

Algorithm 2 に示す. 6 行目では照合結果をパターン保有者, または文字列保有者に出力する処理を行う. この処理を Algorithm3 に示す.

文字列の i 番目の文字に関して, 配列 Shift-OR の出力 $S_i[m]$ は 0 から m までの値をとり, $S_i[m] = 0$ の場合パターンの検出成功 (active), $S_i[m] \neq 0$ の場合パターンの検出失敗 (inactive) を示す. $S_i[m]$ の値の公開は, パターン保有者, 文字列保有者どちらに対しても, 他方の情報の推測を可能にしてしまうため, Algorithm 3 の式 6 では, 照合結果取得者が $S_i[m] = 0$ かのみを取得するよう式 6 の乱数 $V[i]$ でランダム化し, 照合結果のシェアを生成している.

Algorithm 3 の 3 行目以降は, シェアから照合結果を復号する処理である. この処理は最終結果をパターン保有者, 文字列保有者どちらに公開するかによって異なる. 4 行目はパターン保有者に照合結果を公開する処理である. $\Gamma^{\mathrm{PH}}[i] = S_i[m]^{V[i]}$ について, パターン保有者は $V[i]$ を持たないため, $S_i[m] = 0$ かの情報のみを得る. 5-8 行目は文字列保有者に照合結果を公開する処理である. 式 7 で生成される $\Gamma^{\mathrm{SH}}[i] = S_i[m]^{V[i] \cdot W'[i]}$ について, 文字列保有者は $W'[i]$

Algorithm 3 照合結果の出力処理

-PH's input: $\mathrm{sk}^{\mathrm{PH}}$
-SH's input: $\mathrm{pk}^{\mathrm{PH}}$, $\hat{S}_i[m]$
-PH's output: $\Gamma^{\mathrm{PH}}[i]$
-SH's output: $\Gamma^{\mathrm{SH}}[i]$

1: SH は乱数 $V[i], W[i] \in_r \mathbb{Z}_N$ を生成し, 式 6 より $\bar{Z}[i]$ を計算し PH に送信する.

$$\bar{Z}[i] = \bar{S}_i[m]^{V[i]} \circ \mathrm{Enc}_{\mathrm{pk}^{\mathrm{PH}}}(W[i])^{-1} \tag{6}$$

2: PH は $\bar{Z}[i]$ を復号し, $Z[i]$ を得る.
3: **if** PH に照合結果を公開 **then**
4: SH は $W[i]$ を PH に送信し, SH は $\Gamma^{\mathrm{SH}}[i] = \emptyset$, PH は $\Gamma^{\mathrm{PH}}[i] = Z[i] + W[i] = S_i[m]^{V[i]}$ をそれぞれ出力する.
5: **else if** SH に照合結果を公開 **then**
6: SH は $\mathrm{pk}^{\mathrm{SH}}$, $\mathrm{sk}^{\mathrm{SH}}$ を生成し, PH に $\mathrm{pk}^{\mathrm{SH}}$, $\bar{W}[i] = \mathrm{Enc}_{\mathrm{pk}^{\mathrm{SH}}}(W[i])$ を送信する.
7: PH は乱数 $W'[i]$ を生成し, 式 7 より $\bar{Z}'[i]$ を計算し SH に送信する.

$$\bar{Z}'[i] = (\bar{Z}[i] \circ \bar{W}[i])^{W'[i]} \tag{7}$$

8: SH は $\Gamma^{\mathrm{SH}}[i] = \mathrm{Dec}_{\mathrm{sk}^{\mathrm{SH}}}(\bar{Z}'[i]) = S_i[m]^{V[i] \cdot W'[i]}$, PH は $\Gamma^{\mathrm{PH}}[i] = \emptyset$ をそれぞれ出力する.
9: **end if**

を持たないため,$S_i[m] = 0$ かの情報のみを得る. Algorithm 1 について, 次の定理を与える.

Theorem 1. パターンが *CM+CC* クラスに属するならば, Algorithm *1* は *Definition 1* におけるパターン保有者および文字列保有者によるオンライン秘密パターン照合を正しく解く.

なお証明は省略するが, 式 5 の S_i の更新は準同型性の性質より自明である.

4.3 秘密拡張配列 Shift-OR

次に, 拡張配列 Shift-OR を秘密計算に拡張したプロトコルを Algorithm 4 に示す. このプロトコルでは定数文字, 文字種, 無限長ギャップで構成されたパターンを照合可能である. このプロトコルが解く問題では, パターン保有者は P に加えセルフループを行う状態番号集合 $L \subset \{1, ..., m\}$ を持つ. このプロトコルではセルフループの遷移処理 (式 4) を含む遷移状態の更新を秘密計算により行う. Algorithm 4 は遷移状態の更新処理以外は, Algorithm 1 と同様の処理を行う.

Algorithm 5 では, $\bar{S}_i$ を表 4 のように更新するため次の手順を行う. 3 行目では, 文字列保有者は $\bar{S}_i$, $\bar{S}_{i-1}$ に乱数を加えて生成した値 $\bar{S}'_i$, $\bar{S}'_{i-1}$, $j \notin L$ のときに用いる値 $\bar{\Delta}_i$ を送信する. パターン保有者は各 j について, $j \in L$ の場合,

Algorithm 4 秘密拡張配列 Shift-OR

-PH's input: $P \in \Sigma$, $L \subset \{1, ..., m\}$, $\Sigma = \{w_1, ..., w_{|\Sigma|}\}$
-SH's input: $T \in \Sigma$, $\Sigma = \{w_1, ..., w_{|\Sigma|}\}$
-PH's output: $\Gamma^{\mathrm{PH}}[i]$
-SH's output: $\Gamma^{\mathrm{SH}}[i]$

1: PH は秘密鍵 sk, 公開鍵 pk を生成し, サーバに pk を送信する.
2: PH は各 w_i についてマスクビット配列 M_{w_i} を生成し, 各要素を暗号化しサーバに送信する.
3: SH は $\bar{S}_0[0] = \mathrm{Enc}_{\mathrm{pk}}(0)$, $\bar{S}_0[j] = \mathrm{Enc}_{\mathrm{pk}}(1) (j \in \{1, ..., m\})$ により $\bar{S}_0$ を生成し, 次の演算を行う.
4: **for** $i = 1$ to n **do**
5: SH は Algorithm 5 により S_i を更新する.
6: Algorithm 3 により, PH は $\Gamma^{\mathrm{PH}}[i]$ を, SH は $\Gamma^{\mathrm{SH}}[i]$ を出力する.
7: **end for**

Algorithm 5 秘密拡張配列 Shift-OR 法の状態更新処理

-PH's input: L, $\mathrm{sk}_{\mathrm{PH}}$, $\mathrm{pk}_{\mathrm{PH}}$
-SH's input: $T[i]$, $M_{T[i]}$, $\bar{S}_{i-1}$, $\mathrm{pk}_{\mathrm{PH}}$
-PH's output: ϕ
-SH's output: $\bar{S}_i$

1: SH は Algorithm 2 により, $\bar{S}_i$ を生成する.
2: **for** $j = 1$ to m **do**
3: SH は乱数 $K_i[j]$, $R_i[j]$, $Q_i[j]$, $U_i[j] \in_r \mathbb{Z}_N$ を生成し, $\bar{S}'_i$, $\bar{S}'_{i-1}$, $\bar{\Delta}_i$ を PH に送信する.

$$\begin{aligned} \bar{S}'_i[j] &= \bar{S}_i[j] \circ \mathrm{Enc}_{\mathrm{pk}}(R_i[j]) \\ \bar{S}'_{i-1}[j] &= \bar{S}_{i-1}[j] \circ \mathrm{Enc}_{\mathrm{pk}}(Q_i[j]) \\ \bar{\Delta}_i[j] &= \bar{S}_{i-1}[j]^{R_i[j]} \circ \mathrm{Enc}_{\mathrm{pk}}(Q_i[j] \cdot R_i[j]) \\ &\quad \circ \bar{S}_i[j]^{K_i[j] \cdot Q_i[j]} \circ \mathrm{Enc}_{\mathrm{pk}}(U_i[j]) \end{aligned}$$

4: **end for**
5: **for** $j = 1$ to m **do**
6: PH は以下より配列 $\bar{S}^*_i$, $\bar{E}_i$ を生成し, SH に送信する.
7: **if** $j \in L$ **then**
8:

$$\begin{aligned} \bar{S}^*_i[j] &= \mathrm{Enc}_{\mathrm{pk}}(\mathrm{Dec}_{\mathrm{sk}}(\bar{S}_i[j]) \cdot \mathrm{Dec}_{\mathrm{sk}}(\bar{S}_{i-1}[j])) \\ &= \mathrm{Enc}_{\mathrm{pk}}((S_i[j] + R_i[j]) \cdot (S_{i-1}[j] + Q_i[j])) \\ &= \mathrm{Enc}_{\mathrm{pk}}(S_i[j] \cdot S_{i-1}[j] + Q_i[j] \cdot S_i[j] \\ &\quad + R_i[j] \cdot S_{i-1}[j] + R_i[j] \cdot Q_i[j]) \end{aligned} \tag{8}$$

9: $\bar{E}_i[j] = \mathrm{Enc}_{\mathrm{pk}}(0)$
10: **else**
11:

$$\bar{S}^*_i[j] = \bar{\Delta}_i[j] \circ \mathrm{Enc}_{\mathrm{pk}}(0) \tag{9}$$

12: $\bar{E}_i[j] = \mathrm{Enc}_{\mathrm{pk}}(1)$
13: **end if**
14: **end for**
15: **for** $j = 1$ to m **do**
16:

$$\begin{aligned} \bar{S}_i[j] &= \bar{S}^*_i[j] \circ \bar{S}_{i-1}[j]^{-R_i[j]} \circ \mathrm{Enc}_{\mathrm{pk}}(Q_i[j] \cdot R_i[j])^{-1} \\ &\quad \circ \bar{S}_i[j]^{-Q_i[j]} \circ \bar{E}_i[j]^{-U_i[j]} \end{aligned} \tag{10}$$

17: **end for**

$j \notin L$ の場合で異なる処理を行う. 7-9 行目の処理は $j \in L$ の場合の処理である. パターン保有者は式 8 において, $\bar{S}'_i$, $\bar{S}'_{i-1}$ を復号し, その

積の値を計算する．その値を暗号化し, $\bar{S}_i^*[j]$ の値とする．10-13 行目の処理は $j \notin L$ の場合の処理である．式 9 では, パターン保有者がある j に対し, どちらの処理を行ったかを文字列所有者に秘匿するため, $\bar{\Delta}_i[j]$ を再暗号化する．16 行目の処理では, 文字列保有者はパターン保有者から受け取った $\bar{S}_i^*$, $\bar{E}_i$ を用いて, $\bar{S}_i[j]$ を更新する．Algorithm 4 について, 次の定理を与える．

Theorem 2. パターンが *CM+CC+ILG* クラスに属するならば, Algorithm *4* は *Definition 1* におけるパターン保有者および文字列保有者によるオンライン秘密パターン照合を正しく解く．

なお, 証明は省略するが, これは Algorithm 5 が式 4 と同様の処理を行うことを示すことによって導かれる．

4.4 提案法の性質

計算量解析 秘密配列 Shift-OR 法は遷移状態の更新ごとに $\bar{S}_i[m]$ のみを出力するため, ラウンド計算量は $O(1)$, 更新に必要な通信量は $O(1)$, 計算量は $O(m)$ である． また, 秘密拡張配列 Shift-OR では更新ごとに遷移状態を通信/計算するため, ラウンド計算量は $O(n)$, 計算量/通信量は $O(m)$ である．詳細な計算量解析は表 1 を参照されたい．

安全性証明 提案する 2 つのプロトコルに関して, 次の定理を与える．

Theorem 3. 用いる公開鍵暗号が強秘匿性および加法準同型性を満たし, かつ, パターン保有者および文字列保有者が *semi-honest* に振る舞うならば, 秘密配列 *Shift-OR* および秘密拡張配列 *Shift-OR* は *Definition 1* の意味で安全である．

証明は view を模倣するシミュレーションが多項式アルゴリズムとして得られることを示すことで得られるが, 紙数の都合上省略する．

5 まとめと今後の課題

本稿では Shift-OR 法を利用して NFA を秘密計算上で評価する手法を提案した．NFA を直接評価することで, NFA から DFA への変換で状態数が爆発する既存手法の問題点を解決することができる．また, 提案手法は出現するアルファベットの種類数と文字列長の積に依存しないため, 自然言語からなるテキストなどより広範なデータに対応する．さらに, 提案手法は文字列中でのパターンの出現位置を出力することが可能であり, 事前処理, 更新処理を文字列長に依存しない通信量/計算量で行うことができる．そのため, ストリームデータのような無限の文字列長を持つ文字列に対しても一般性を失わず適用可能である．今後の課題は, 提案手法の枠組みでの, より多くの正規表現への対応である．

謝辞 本研究は, 最先端研究開発プログラム「超巨大データベース時代に向けた最高速データベースエンジンの開発と当該エンジンを核とする戦略的社会サービスの実証・評価」及び科学研究費 12913388 の助成を受けた．また, 北海道大学の吉田諭史氏には，パターン照合に関して多大な議論を頂いたことに感謝する．

参考文献

[1] R. Baeza-Yates and G. H. Gonnet. A new approach to text searching. *CACM*, 35(10):74–82, Oct. 1992.

[2] K. B. Frikken. Practical private dna string searching and matching through efficient oblivious automata evaluation. In *Data and Applications Security XXIII*, pages 81–94. Springer, 2009.

[3] Y. Lindell and B. Pinkas. A proof of security of yao ' s protocol for two-party computation. *Journal of Cryptology*, 22(2):161–188, 2009.

[4] G. Navarro and M. Raffinot. *Flexible pattern matching in strings — practical on-line search algorithms for texts and biological sequences.* Cambridge, 2002.

[5] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in cryptology EUROCRYPT' 99*, pages 223–238. Springer, 1999.

[6] D. Perrin. Finite automata. In J. van Leeuwen, editor, *Handbook of Theor. Comput. Sci, Vol.B, Chap. 1*, pages 1–57, 1990.

[7] J. R. Troncoso-Pastoriza, S. Katzenbeisser, and M. Celik. Privacy preserving error resilient dna searching through oblivious automata. In *Proc. Comput. Commun. Security (CCS'07)*, pages 519–528. ACM, 2007.

[8] S. Wu and U. Manber. Fast text searching: allowing errors. *CACM*, 35(10):83–91, Oct. 1992.

[9] A. C.-C. Yao. Protocols for secure computations. In *FOCS'82*, pages 160–164, 1982.

[10] 渡邊裕治，立石孝彰. 通信回数を低減した紛失オートマトン計算. In 暗号と情報セキュリティシンポジウム (SCIS 2012) 予稿集, 2012 年.