1X-01

DAWGに基づいた部分文字列検索可能暗号の改善

小田　亮輔†　山本　博章†　藤原　洋志†

信州大学工学部†

1. はじめに

近年, クラウドコンピューティングの普及により, リモートストレージサービスの需要が拡大している. このようなシステムでは、サーバー上のデータの機密性を保護するため, データを暗号化して保存し, 暗号化したままデータを検索する技術の開発が重要となる. このような検索手法は、検索可能暗号と呼ばれ, 特に, 対称鍵暗号化を使用する手法は, 検索可能対称暗号 (SSE) と呼ばれる. これまで, このような背景の下で安全かつ効率的な SSE に関する研究が積極的に行われてきた [1], [2].

山本ら [3] は, 平文の部分文字列検索問題を解決することで知られている DAWG を改良し, 空間効率が良い安全な部分文字列 SSE を提案した. しかし, [3] はドキュメントにクエリ (検索文字列) が含まれない誤った検索結果を返す場合がある. これは偽陽性 (false positive) と呼ばれる. そこで, 本論文では偽陽性を解消できるように [3] を改良した手法を提案し, その実装内容及び結果について報告する.

2. 準備

$D = \{d_0, \ldots, d_{N-1}\}$ を N 個のドキュメントが含まれる集合, $x \parallel y$ を文字列 x と y を連結した文字列とする. 各ドキュメント $d_i (0 \leq i \leq N-1)$ はドキュメント ID と呼ばれる識別番号 i が割り振られている. また, 任意のクエリ q に対して $D(q) = \{(i,j) \mid q$ は d_i の位置 j に出現する $\}$ と定義する. 本手法において, トラップドアと暗号化索引の作成に疑似ランダム関数 F_{sk}, データの暗号化に共通鍵暗号 Enc_{sk} を使用する. ここで, sk は秘密鍵である.

3. 暗号化索引

暗号化索引 Π = **(LMAP, NMAP, IND)** は, ADAWG[3] に対し, 状態遷移を暗号化したまま行う **LMAP**, 最後の遷移で到達可能な状態における情報が保存されている **NMAP**, 及びドキュメント ID と文字列の出現位置を保持するペアを取り出すための **IND** から構成される. **NMAP**, **IND** の構成は [3] と同様であるため省略する.

3.1 LMAP の構成

$\tilde{M}(D) = (Q, \tilde{\Sigma}, \tilde{\delta}, st_0)$ を D に対する ADAWG とする. ここで, Q は状態の集合, $\tilde{\Sigma}$ は遷移ラベル (メタ記号) の集合, $\tilde{\delta}$ は遷移関数, st_0 は初期状態とする. $\sigma \in \tilde{\Sigma}$ に対し, $\tilde{\delta}(out, \sigma) = in$ が成立するような $\tilde{M}(D)$ の遷移元の状態 out, 遷移先の状態 in は一意に決定できる. よって, $\mathbf{LMAP}[F_{sk_1}(\sigma \parallel 0) \oplus r_{out}]$ には $F_{sk_1}(\sigma \parallel 1) \oplus r_{in}$ が格納される. ここで, r_{st} は状態 st に対応する乱数である. r_{st} で暗号化を施すことにより, **LMAP** は $\tilde{M}(D)$ の構造を隠したまま状態遷移を実現し, 偽陽性を排除できる.

4. 検索

検索プロトコルは, 3 ラウンドの通信で実行される. ここで, **LMAP** におけるキーの集合を $\mathbf{LMAP}.keys$ とする. 1 ラウンド目にユーザーはクエリ q から **MakeTrap1** を用いて $TRAP(q)$ をサーバーへ送る. サーバーは **ExistTest** を用いて q がドキュメントに含まれているか確認する. 含まれている場合, サーバーから返された $EXIST$ と q からユーザーは **MakeTrap2** を用いて, 2 ラウンド目に (Y, Z) をサーバーへ送る. サーバーは **NMAP** と **IND** を用いて $D(q)$ をユーザーに返す. そして, 3 ラウンド目にユーザーはサーバーへドキュメント ID を送り, 目的のドキュメントを得る.

Algorithm 1 MakeTrap1(q)

Input: $q = a_1 \cdots a_m$
1: $P_1 \leftarrow (F_{sk_1}(a_1 \parallel 0), F_{sk_1}(a_1 \parallel 1))$
2: $X_1 \leftarrow \{P_1\}$
3: $SYM \leftarrow \emptyset$
4: **for** $j = 2$ to m **do**
5: $k \leftarrow 1$
6: **for** $i = 1$ to $j-1$ **do**
7: $\sigma \leftarrow a_i \cdots a_j$
8: **if** $\sigma \notin SYM$ **then**
9: $P_k \leftarrow (F_{sk_1}(\sigma \parallel 0), F_{sk_1}(\sigma \parallel 1))$
10: $X_j \leftarrow X_j \cup \{P_k\}$
11: $k++$
12: $SYM \leftarrow SYM \cup \{\sigma\}$
13: **end if**
14: **end for**
15: **end for**
16: **return** $TRAP(q) = (X_1, \ldots, X_m)$

Improving DAWG-based Substring Searchable Symmetric Encription

† Ryosuke Oda, Hiroaki Yamamoto, Hiroshi Fujiwara, Faculty of Engineering, Shinshu University

Algorithm 2 ExistTest($TRAP(q)$, LMAP)

Input: $TRAP(q) = (X_1, \ldots, X_m)$

$r \leftarrow 0$
for $j = 1$ to m **do**
  $Flag \leftarrow false$
  for $i = 1$ to $|X_j|$ **do**
    $(Y, Z) \leftarrow X_j$ から P_i を取り出す.
    $key \leftarrow Y \oplus r$
    if $key \in$ **LMAP**.$keys$ **then**
      if $j = m$ **then**
        return Y
      end if
      $Flag \leftarrow true$
      $r \leftarrow LMAP[key] \oplus Z$
      break
    end if
  end for
  if $Flag = false$ **then**
    return $\emptyset$
  end if
end for
return $TRAP(q) = (X_1, \ldots, X_m)$

Algorithm 3 MakeTrap2(q, $EXIST$)

Input: $q = a_1 \cdots a_m$

if $m = 1$ **then**
  $X \leftarrow F_{sk_1}(a_1 \parallel 0)$
  if $X = EXIST$ **then**
    $Y \leftarrow F_{sk_2}(a_1)$
    $Z \leftarrow F_{sk_3}(a_1)$
  end if
end if
if $m \geq 2$ **then**
  for $i = 1$ to $m - 1$ **do**
    $\sigma \leftarrow a_i \cdots a_m$
    $X \leftarrow F_{sk_1}(\sigma \parallel 0)$
    if $X = EXIST$ **then**
      $Y \leftarrow F_{sk_2}(\sigma)$
      $Z \leftarrow F_{sk_3}(\sigma)$
      break
    end if
  end for
end if
return (Y, Z)

5. 実験

提案手法について実験的に評価したため, その結果を示す. 実験では, ドキュメントとして, ランダムに並べた半角英数字から構成される 20MB のテキストファイルを使用した. 結果として, ADAWG におけるメタ記号の数は 48430517 個, ドキュメント ID と文字列の出現位置を保持するペアの数は 20971520 個であった. OS は Windows10, CPU は i7-9700, メモリは 32GB の環境において, Python3 を用いて提案手法の実装行った. 共通鍵暗号に AES, 疑似ランダム関数は HMAC-SHA256 を用いた.

今回, 暗号化索引の実装において, 辞書型を用いて F_{sk} における 128 ビット長の出力を int 型, Enc_{sk} の出力を byte 型として格納した. 暗号化索引のサイズを [3] と比較した結果, [3] は 5248MB, 提案手法は 5760MB であった. クエリがドキュメントに含まれるか確認する際, [3] はメタ記号の集合 (**LSET** と呼ばれる) を用いるのに対し, 提案手法はメタ記号と状態に対応する乱数を格納した連想配列を用いるため, サイズは大きくなった.

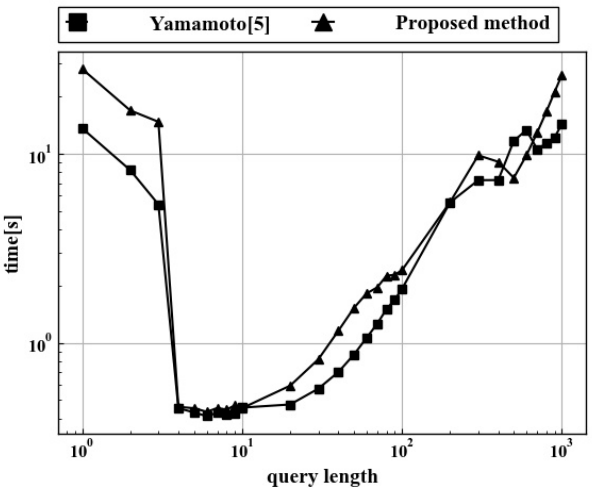

図 1 検索時間の比較

次に, 検索時間を [3] と比較した結果を図 1 に示す. 検索時間はクエリを与えてからドキュメントを得るまでの時間を計測し, クエリはドキュメントに含まれる部分文字列から無作為抽出したものを用いた. 提案手法では乱数を得るための計算が追加されたため, [3] よりも時間がかかった. また, クエリ長が短いため, 復号するペアの数が増加し, 時間がかかる場合があった.

6. まとめ

本論文では, [3] を偽陽性の解消に向けて改良した手法を提案した. 本手法は [3] と比較すると, 暗号化索引のサイズと検索時間をさらに要するが, ADAWG における状態遷移を実現できるため, より正確な検索が可能である. 今後の課題として, さらなる索引サイズ, 検索時間の削減に向けて, よりコンパクトな有限オートマトンの開発が挙げられる.

参考文献

[1] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, Searchable symmetric encryption: Improved definitions and efficient constructions, Journal of Computer Security, pp.895–934, 2011.
[2] 三好竜司, 山本博章, Backward 安全に向けた検索可能暗号の改良, SCIS2019, pp1-8, 2019.
[3] H. Yamamoto, Y. Wachi and H. Fujiwara, Space-Efficient and Secure Substring Searchable Symmetric Encryption Using an Improved DAWG, ProvSec2019, pp130-148, 2019.