ビット分解プロトコルに基づく文字列間の距離計算

Calculation of Distance between String based Bit-decomposition Protocol

中川竣太 [1*]　坂本時緒 [1]　申吉浩 [2]　坂本比呂志 [1]
Shunta Nakagawa[1]　Tokio Sakamoto[1]　Yoshihiro Shin　[2] Hiroshi Sakamoto[1]

[1] 九州工業大学情報工学府
[1] Graduate School of Computer Science and Systems Engineering, Kyushu Institute of Tecnology，Japan
[2] 兵庫県立大学応用情報科学研究科
[2] Graduate School of Applied Informatics University of Hyogo

Abstract: The purpose of this study to calculate degree of similarity based grammar compression between a character string S which a user has and W which a database stores on condition that those information is concealed. We suggest a protocol to realize concealment calculation using additively homomorphic encryption and bit-decomposition protocol. We implemented process secret retrieval of product rule to build derivation tree. We measured the processing time.

1　はじめに

マイナンバーやゲノムデータなどの個人情報や開発中の化合物などを用いてデータベース上の検索を行うとき，クライアントは検索する内容をデータベース側に秘匿したまま検索を行いたい．この時，データベース側にもクライアント側に必要以上にデータの情報を教えたくないようにする．これらの要求を満たすために，検索内容を暗号化したままデータベース上の検索を行う．これを秘匿検索という．秘匿検索は通常の計算より多くの処理が必要となる．軽減する方法として暗号化した数値を秘匿性を損なうことなくビット列への変換を行うビット分解プロトコル [1] というものがある．ビット分解プロトコルを用いることで，暗号文の論理回路演算を効率よく行うことができる．本研究では，加法準同型暗号のビット分解プロトコルと絶対値の近似計算 [2] によって，ユーザが保持している文字列とデータベース上の文字列の文法圧縮による類似度を秘匿計算を行う手法を提案する．

2　準備

文字の有限集合を Σ とおき，Σ の要素からなる文字列全体の集合を Σ^* と定義する．log の表記は $\lg = \log_2$，$\lg^{(1)} N$,$\lg^{(i+1)} N = \lg \lg^{(i)} N$ とする．$\lg^*$ は，$\lg^* = min\{i | lg^{(i)} N < 1\}$ であり，非常に大きい N に対して定数である．本研究では，サーバ（データベース）S と複数人のユーザ $U_1, \ldots, U_{h-1}$ によって協力して計算を行う．このとき，どの二者も共謀しないということを仮定する (semi-honest model)．

2.1　秘匿検索

秘匿検索を用いるために，加法準同型暗号である Paillier 暗号を用いる．Paillier 暗号は，ある 2 つの大きな素数 q, p をとり，その合成数 $n = pq$ を計算する．$k \in Z_n$ を選び，$g = (kn + 1) \bmod n^2$ を計算する．公開鍵は (g, n)、秘密鍵は (p, q) である．ある平文 m を Paillier 暗号で暗号化したものを $E[m]$ とすると，$E[m]$ は次のように計算する．

$$E[m] = g^m \cdot r^n \bmod N^2$$

ここで，r は $r \in Z^*$ で，暗号化ごとに選ばれる乱数である．$c = E[m]$ を復号するには次のように計算する．

$$m = \frac{L(c^\lambda \bmod N^2)}{g^\lambda \bmod N^2} \bmod n$$

ここで，$L(u) = \frac{u-1}{n}$ であり，$\lambda = \mathrm{lcm}(p-1, q-1)$ である．

Paillier 暗号は，ある平文 m_1, m_2 の暗号文 $E[m_1], E[m_2]$

*連絡先：九州工業大学大学院情報工学府先端情報工学専攻
〒 820-8502 福岡県飯塚市川津 680 番 4
E-mail:s_nakagawa@donald.ai.kyutech.ac.jp

があるとき，以下のような計算ができる

$$E[m_1 + m_2] = E[m_1]E[m_2]$$

$$E[m_1 - m_2] = E[m_1]E[m_2]^{-1}$$

このような性質をもつ暗号を加法準同型暗号という．また，平文 c と暗号文 $E[m]$ に対して以下のような計算ができる．

$$E[cm] = E[m]^c$$

この性質を利用することで，秘匿検索を行うことができる．

2.2 ビット分解プロトコル

秘匿検索を拘束で行うために Paillier 暗号のビット分解プロトコル [1] を用いる．ビット分解プロトコルとは，暗号化された整数を秘匿性を保ったままビットごとに分解した暗号文にする手法である，ビットまた特定の暗号文のみを複合できるように (n, n) 閾値 Paillier 暗号を用いる．(n, n) 閾値 paillier 暗号とは，秘密鍵を n 個に分割し，n 個の秘密鍵を使わなければ復号できない暗号である．分割した秘密鍵をユーザとサーバに分散することで，秘密情報を復号するためには全てのユーザとサーバが同意しなければならないようにすることができる．ここでは，ユーザ 2 人と，サーバ 1 つで計算を行うこととする (図 1).

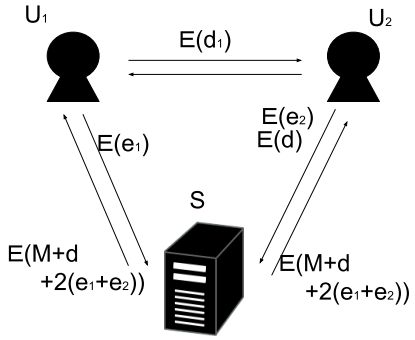

図 1: ビット分解プロトコルのマルチパーティー計算

以下がそのアルゴリズムである．ここで，$m(i)$ は自然数 m の 2 進数の下位 i 番目のビット $(0 \le i \le \ell - 1)$ である．
(1) ユーザ U_1 ,U_2 は，それぞれ乱数ビット d_1,d_2 と乱数 e_1,e_2 を生成して $E[d = d_1 \oplus d_2]$，$E[e_1], E[e_2]$ をサーバへ送る．
(2) サーバは $E[M + d + 2(e_1 + e_2)]$ を計算して，$\alpha = M + d + 2(e_1 + e_2)$ を全てのユーザとサーバの同意のもと復号して，最下位ビット $\alpha(0)$ を得る．
(3) サーバは $E[M(0) = d \oplus \alpha(0)] = E[d + \alpha(0) - 2d\alpha(0)]$ を計算して，$E(2^{-1}(M - M(0)))$ に対して以上の操作を繰り返すことで，再帰的に $E[M(0)], E[M(1)], \ldots, E[M(\ell - 1)]$ を得る．

2.3 文法圧縮

ある文字列 $w \in \Sigma^*$ のみを生成する文脈自由文法 (CFG) を文法圧縮という．文法圧縮 G は一般に $G(P, S)$ で表される．S は開始記号であり，P は $Z \to \alpha$ といった生成規則の集合である．$Z \in V$ は非終端記号と呼ばれ，$\alpha \in (\Sigma \cup V)^*$ は文字列を表す．V は任意の変数である．P にすべての非終端記号が無くなるまで生成規則を適用することで，元の文字列 w をえることができる．特に，任意の生成規則が $X_k \to X_i X_k (k \ge i, j)$ を満たす文法圧縮を Straight-Line Program(SLP) という．

2.4 移動付き編集距離

ある二つの文字列 S, W が与えられたとき，文字の挿入・削除・置換・および部分文字列移動の四つの編集操作としたとき，$S \to W$ の最小回数を移動付き編集距離 (EDM) $d(S, W)$ とする．EDM はこの計算は NP-困難であるが，近似解である L_1 距離を求めるアルゴリズム Edit Sensitive Parsing(ESP) が知られている．ESP はある文法圧縮を計算するアルゴリズムである．

定理 1:(cormode and Muthukrishnan [4])
ESP の出力 G から作られる構文木に出現する各文字 a_i の頻度 $f(a_i)$ で定義されるヒストグラム $v_g = (f(a_1), ..., f(a_n))$ とする．文字列 S,W が与えられたとき，

$$||v_s - v_w||_1 = \sum_{i=1}^{n} |f_s(a_i) - f_w(a_i)| = O(\lg N \lg^* N) d(S, W)$$

が成立する．ただし，$N = |SW|$

よって，v_s を用いることで S と W の距離の近似値を求めることができる．さらに，この近似率を保ったまま以下の性質を満たすように改良できる．

定理 2:(Maruyama et al. [5])
EDM の出力 $G = (P, S)$ を以下を満足するように限定できる．
(1) G は SLP である
(2) 任意の生成規則 $Z \to XY, Z' \to X'Y' \in P$ に対して，$XY \le_{lex} X'Y' \Rightarrow Z \le_{lex} Z'$．ここで，$\le_{lex}$ は文字の辞書順で定義される $(\Sigma \cup V)^*$ 上の全順序である．

3　文法圧縮による距離の秘匿計算

ユーザーが文字列 S をもっており，データベースには様々な文字列 W の v_w が格納されているとする．ここで，G の生成規則 $Z_k \to XY$ が公開されているのなら，S を ESP で圧縮するとき,XY に対して Z_k を割り当て構文木を作ることで，v_s を得ることができる．v_s と v_w を用いることで $d(S,W)$ の近似値を計算することができる．これを，ユーザー側の情報を秘匿した状態で行うために次のような手順で行う。

（1）ユーザーはある X,Y の組を暗号化した状態でサーバ（データベース）に送る．サーバ側は秘匿性を保持したまま，暗号化された k をユーザーに送信し，ユーザーはそれを復号して k を取得する．このとき，XY の組がデータベース上になければ $k=0$ を返す.

（2）k を得ることで，v_s が計算できる．その各成分を暗号化し，サーバに送る．サーバは d(S,W) の近似値として，$E[||v_s - s_w||] = E[\sum_{i=1}^{n} |f_s(a_i) - f_w(a_i)|]$ を秘匿したまま計算を行い，ユーザーに送信する．ここで秘匿計算を行うためベクトル x と y の L_1 距離 $||x-y||_1$ は，論文 [2] で示されている次元縮小関数 $g : x \to \hat{x}$ により，$||x-y||_1 \simeq ||\hat{x}-\hat{y}||_2^2$ と近似計算を用いる．これにより，暗号化された整数の絶対値を取る計算を行わずに，$d(S,W)$ の近似値を得ることができる．

3.1　$XY \to Z_k$ の秘匿計算

任意の文字 $X \in \Sigma \cup V$ は固定長のビット列で保持されているとする．このとき，XY はその 2 倍長のビット列で表現されるある整数と同一である．したがって，$XY \to Z_k$ の問い合わせは，ある自然数 a と $n \leq |\Sigma \cup V|^2$ なるある自然数の列 $(a_1, a_2, \ldots, a_n)$ に対して，$a_k = a$ を満たす k があればそれを出力するかさもなくば $k=0$ を出力する問題と等しい．ただし，定理 2 により，この数列は狭義の単調増加列であると限定できるため，この条件をみたす k は高々一つである．加法準同型暗号である，paillier 暗号を用いて秘匿計算を行う．以降では，任意の文字の組 X_iX_j の表現長を ℓ ビットに固定して，次のような記号を使用する．

$m^+ = \sum_{i=0}^{\ell-1} m(i)$
$m^* = ((m^+)\cdots)^+$: $\lceil \lg^* m \rceil$ 回の繰り返し．ただし，$\lg^* m$ は，$\lg^{(1)} m = \lg m$，$\lg^{(i+1)} m = \lg(\lg^{(i)} m)$ とするとき，$\lg^* m = \min\{i : \lg^{(i)} m \leq 1\}$ と定義される．

step1: ユーザは，検索文 m の 2 進数の下位 i 番目のビット $m(0), m(1), \ldots, m(\ell-1)$ をそれぞれ暗号化し $(E[m(0)], E[m(1)], \ldots, E[m(\ell-1)])$ をサーバへ送る．

step2: サーバは各 $i(1 \leq i \leq n)$ に対して $E[M_i] = \sum_{j=0}^{\ell-1} m_i(j) \oplus E[m(j)]$ を計算する．

step3: サーバは $E\left[k = \frac{1}{2}n(n-1) - \sum_{i=1}^{n} iM_i^*\right]$ を計算してユーザに送る．

step4: ユーザは $E[k]$ を復号し，$k > 0$ ならば $k \to m \in P$ であり，$k = 0$ ならば $k \to m \notin P$.

step3 における $E[M_i^*]$ の計算には，暗号化された整数 $E[M]$ に対するビット分解プロトコルをサブルーチンとして用いている．

3.2　実験

$XY \to Z_k$ の秘匿計算部分のプログラムを作成し，動作実験を行った．処理の高速化のために 4 スレッドで並列処理を行い，各データサイズに対して 4 文字 (32bit) のデータで 10 回処理を繰り返した．実験環境は，Intel Core i3-4130 CPU 3.40GHz/4GB RAM である．平均処理時間を表 1 に示す.

表 1: 平均処理時間

データ数	平均時間 [s]
10	0.119
100	1.311
1000	9.727
10000	98.071

4　まとめ

本稿では，ビット分解プロトコルを用いて文字列の類似度を秘匿計算する手法を提案した．また，生成規則を秘匿計算する処理を実装し，その処理時間を計測した．

今後の課題としては，L_1 距離の近似値計算などの秘匿計算全体の実装を行うこと，ユーザ側だけでなくデータベース側の秘匿性を担保すること，ElGamal などの他の手法を用いることで高速化を達成すること，また，他の手法との比較を行い性能を評価を行うことである．

参考文献

[1] Berry Schoenmakers, Pim Tuyls: Efficient Binary Conversion for Paillier Encrypted Values. EUROCRYPT 2006: 522-537

[2] Shantanu Rane, Wei Sun, Anthony Vetro: Privacy-preserving approximation of L1 distance for multimedia applications.

[3] Pascal Paillier: Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. EUROCRYPT 1999: 223-238 2010 IEEE ICME 2010:492-497

[4] Graham Cormode, S. Muthukrishnan: The string edit distance matching problem with moves. ACM Transactions on Algorithms 3(1) (2007)

[5] Shirou Maruyama, Masaya Nakahara, Naoya Kishiue, Hiroshi Sakamoto: ESP-index: A compressed index based on edit-sensitive parsing. J. Discrete Algorithms 18: 100-112 (2013)