IoT デバイスにおける共通鍵暗号と完全準同型暗号を組み合わせた暗号化の高速化検討

松本茉倫　（指導教員：小口正人）

1　はじめに

IoT デバイスの普及によって，様々なセンサデータをクラウドサービス上で統計分析を行い，分析結果を活用することが期待されている．センサデータの中には，秘匿性が高いデータが存在しており，安全とは言えないクラウドサービス上では情報漏洩に備えて，個人情報を保護する必要があるため，暗号文同士の加算・乗算が可能な完全準同型暗号 (以下 FHE: Fully Homomorphic Encryption) が注目されている．しかし，一般的に共通鍵暗号に比べて公開鍵暗号は低速で，公開鍵暗号である FHE による暗号化は共通鍵暗号に比べて処理に時間がかかる．また，FHE の暗号文サイズが大きいため通信量が大きくなってしまうことが計算能力の低い IoT デバイス上での実装の課題となっている．そこで，現在主流な共通鍵暗号である AES と FHE を組み合わせた暗号化の高速化と通信量削減が提案されている [1][2]．本研究では，計算能力の低いスマートフォンのような IoT デバイスにおいて，共通鍵暗号と FHE を組み合わせることで FHE による暗号化の高速化と通信量削減を提案する．共通鍵暗号には AES と eSTREAM という欧州における共通鍵暗号の一種，ストリーム暗号の評価プロジェクトで選定された暗号の一つである TRIVIUM[3] を使用し，それぞれと FHE を組み合わせて比較を行う．

2　従来手法

提案手法の比較として，FHE のみを暗号化に使用する従来システムの概要を図 1 に示し，FHE Only とする．

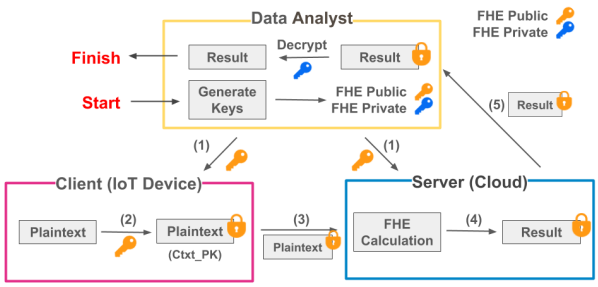

図 1: 従来システム (FHE Only)

(1) Data Analyst が FHE の公開鍵と秘密鍵を生成し，Client(IoT デバイスユーザ) と Server(クラウドサービス) に公開鍵を送信．
(2) Client は FHE の公開鍵で暗号化された暗号文 (Ctxt_PK) を生成．
(3) Ctxt_PK を Server に送信．
(4) Server は Ctxt_PK を使って分析．
(5) Server は分析結果を Data Analyst に送信．

従来システムの問題点として，Client における暗号化に時間がかかること，暗号文サイズが大きくなることが挙げられる．

3　提案手法

3.1　概要

図 2 中の Common Key は共通鍵暗号の AES･TRIVIUM の 2 通りで，それぞれの暗号を使った提案システムを AES+FHE，TRIVIUM+FHE とする．

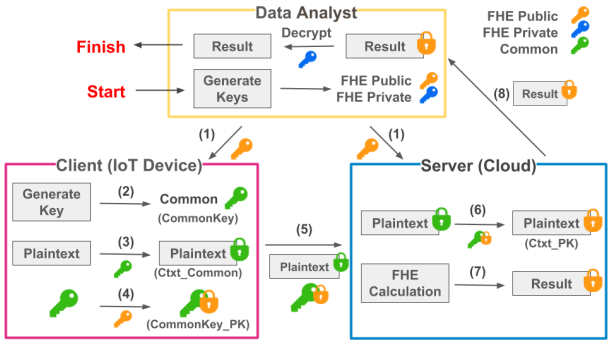

図 2: 提案システム (AES+FHE，TRIVIUM+FHE)

(1) Data Analyst が FHE の公開鍵と秘密鍵を生成し，Client(IoT デバイスユーザ) と Server(クラウドサービス) に公開鍵を送信．
(2) Client は共通鍵 (CommonKey) を生成．
(3) Client は共通鍵で暗号化された暗号文 (Ctxt_Common) を生成．
(4) Client は FHE の公開鍵で暗号化された共通鍵 (CommonKey_PK) を生成．
(5) Client は Ctxt_Common と CommonKey_PK を Server に送信．
(6) Server は Ctxt_Common を CommonKey_PK で共通鍵暗号の復号を行い，FHE の暗号文 (Ctxt_PK) を取得．
(7) Server は Ctxt_PK を使って分析．
(8) Server は分析結果を Data Analyst に送信．

平文が増加しても共通鍵で暗号化すれば良いため，従来手法よりも高速に暗号化できることが予想される．

3.2　FHE による AES 暗号文の復号処理

AES では，16B ずつ区切った平文を 16B の鍵を使って暗号化し，16B ずつの暗号文を生成する．復号には，鍵と暗号文の XOR 演算や定数と暗号文の AND 演算を行う．FHE では暗号化したままで XOR 演算も AND 演算も行うことができるため，暗号化された AES の鍵と暗号文があれば，AES のみの復号を行い，FHE の暗号文にすることが可能である．

3.3　FHE による TRIVIUM 暗号文の復号処理

TRIVIUM では，80bit の鍵と 80bit の初期化ベクトルから平文と同じ長さの Key stream を生成して平文と XOR 演算を行い，平文と同じ長さの暗号文を生成

する．復号には Key stream と暗号文の XOR 演算を行う．FHE で暗号化された鍵から Key stream を生成して暗号文と XOR 演算を行えば，FHE で暗号化したまま TRIVIUM のみの復号を行い，FHE の暗号文にすることが可能である．

4 実験

4.1 実験概要

Client は Google Pixel 3，Server は MacBook Pro を使用した．平文サイズを 16B，64B，128B，192B，256B，320B と変化させ，従来手法 (FHE Only) と提案手法 (AES+FHE，TRIVIUM+FHE) で以下の 3 つの比較を行った．

(1) Client における実行時間
(2) Client から Server に送信するファイルサイズ
(3) Server における共通鍵暗号の復号時間

実験に使用したマシンの性能を表 1 に示す．

表 1: マシン性能

Device	OS	Number of Cores	Processor Speed	RAM
Google Pixel 3	Android 9.0	8	2.5 GHz 1.6 GHz	4GB
MacBook Pro	OS X 10.15	4	1.4GHz	8GB

4.2 実験結果

図 3 に (1)Client における実行時間 を示す．

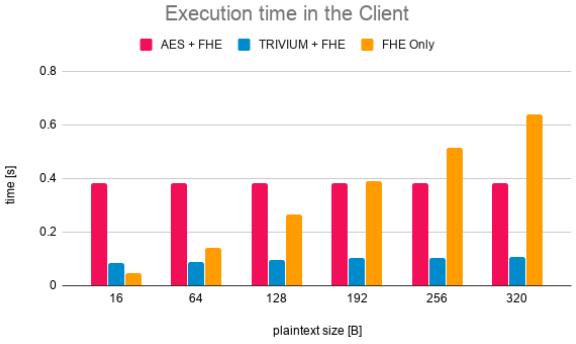

図 3: Client における実行時間

TRIVIUM+FHE が AES+FHE より高速な理由は，TRIVIUM+FHE で暗号化される共通鍵が AES+FHE で暗号化される共通鍵より短いためである．一方で，平文サイズが小さい場合は，FHE Only の方が高速であるという結果になった．提案手法で FHE によって暗号化される共通鍵よりも平文が短い場合にそのような結果になる．

図 4 には (2)Client から Server に送信するファイルサイズ を示す．TRIVIUM+FHE より AES+FHE のファイルサイズが小さいのは，TRIVIUM の共通鍵が AES の共通鍵より短いためである．また，従来手法が提案手法よりファイルサイズが小さくなるのは，平文が提案手法の共通鍵より短い場合である．

図 5 に (3)Server における共通鍵暗号の復号時間を示す．

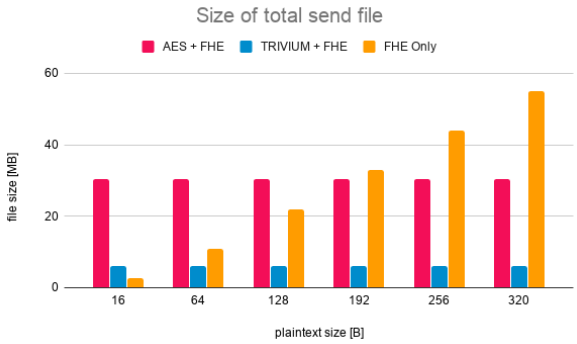

図 4: Client から Server に送信するファイルサイズ

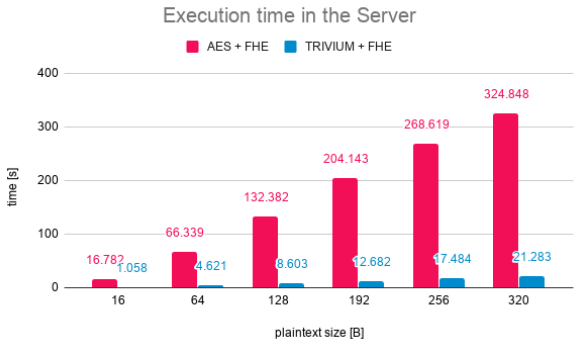

図 5: Server における共通鍵暗号の復号時間

TRIVIUM+FHE の方が AES+FHE に比べてサーバへの負荷が少ないことが分かる．

5 まとめと今後の課題

スマートフォンを Client として，共通鍵暗号の AES･TRIVIUM と FHE を組み合わせた暗号 (AES+FHE，TRIVIUM+FHE) を実装した．その結果，平文が長い場合は従来手法よりも提案手法，特に TRIVIUM+FHE の方が Client における暗号化が高速で，通信量を削減することが可能になった．また AES+FHE より TRIVIUM+FHE の方がサーバへの負荷が少ないことが分かった．今後は AES･TRIVIUM 以外の共通鍵暗号を使った暗号の実装を検討している．

謝辞

本研究は一部，JST CREST JPMJCR1503 の支援を受けたものである．

参考文献

[1] 佐藤 宏樹，馬屋原 昂，石巻 優，今林 広樹，山名早人: 完全準同型暗号のデータマイニングへの利用に関する研究動向，第 15 回情報科学技術フォーラム，2016

[2] Craig Gentry, Shai Halevi, Nigel P. Smart: Homomorphic Evaluation of the AES Circuit, January 3, 2015

[3] Christophe De Canni‘ere, Bart Preneel: Trivium Specifications, eSTREAM, ECRYPT Stream Cipher Project, 2006