Vol.2022-CSEC-97 No.11
Vol.2022-IOT-57 No.11
2022/5/19

準同型暗号を用いたウェブ音声通話の実装

上野 真奈[1,a)]　光成 滋生[2]　村上 啓造[1]　小林 鉄太郎[1]

概要：著者はこれまで準同型暗号を用いた音声及び画像データの暗号重ね合わせの検討を行ってきた．本研究ではウェブ会議音声通話について準同型暗号方式 (楕円 Lifted-ElGamal 暗号方式) を用いた音声通話システムの構築、実装を行った．本発表はその概要と動作状況及び性能評価結果について報告する．

キーワード：準同型暗号, 音声通話, 実装

E2EE Online Web Meeting System with Somewhat Homomorphic Encryption

Mana Ueno[1,a)]　Shigeo Mitsunari[2]　Keizo Murakami[1]　Kobayashi Tetsutaro[1]

Abstract: We have been studying the addition of encrypted voice and image data with somewhat homomorphic encryption. In this study, a multi-party voice communication system using somewhat homomorphic cryptosystem (elliptic curve Lifted-ElGamal cryptosystem) was constructed and implemented for web conference voice communication. This presentation will give an overview of the system, its operation status, and the results of performance evaluation.

Keywords: Somewhat homomorphic cryptosystem, Online web meeting system, Implementation

1. はじめに

1.1 研究背景

2019 年に感染が拡大した COVID–19 以降, 様々な分野で生活様式の見直しが広く行われてきた．「ポストコロナ」時代においては多くの職種でリモートワークが広く浸透し，それに合わせて様々なサービスが展開されてきた．ウェブ会議サービスも単なるビデオ会議のためのツールとしてだけでなく，様々なコミュニケーションツール機能を取り込むことでより便利になりつつある．その一方で， 通信の内容の第三者への漏洩リスクがより重要な課題として浮き上がってきた．コロナパンデミック以前より, 元 NSA 局員である Snowden による NSA の広範な盗聴活動の暴露 [1] に知られるような, 政府機関やサービス提供者による通信の盗聴は問題視されてきた．このような課題の解決方法の一つがエンドツーエンド暗号化 (End-to-end Encryption, E2EE) の技術の利用である．すなわち，サービス提供者への信頼に依存しない，エンドユーザのみが鍵をもち，通信の内容を知ることができるシステム構成が広く行われつつある．この流れはウェブ会議システムにおいても進められている [2].

1.2 本研究の目標

従来のウェブ会議システムでは暗号化処理な共通鍵暗号方式を用いて E2EE 通信を実装する方法が主流である．しかし，共通鍵暗号方式を用いた方式ではデータ通信量およびクライアント端末の性能がボトルネックになり，ユーザスケーラビリティの限界があると考えられる．そこで我々はこれらの課題を解決し, E2EE と多人数の同時接続可能性を両立したウェブ会議システムを実現するため, サー

[1] NTT 社会情報研究所, 〒 180-8585 東京都武蔵野市緑町 3-9-11,
NTT Social Informatics Laboratories, 3-9-11 Midori-cho, Musashino-shi, Tokyo 180-8585, Japan
[2] サイボウズ・ラボ株式会社, 〒 103-6028 東京都中央区日本橋 2-7-1 東京日本橋タワー 27 階,
Cybozu Labs, Tokyo-Nihonbashi-Tower 27th floor, 2-7-1 Nihonbashi, Chuo-ku, Tokyo 103-6028
[a)] mana.ueno.sw@hco.ntt.co.jp

Vol.2022-CSEC-97 No.11
Vol.2022-IOT-57 No.11
2022/5/19

バでデータの重ね合わせを行う Multipoint Control Unit (MCU) 方式に準同型暗号を適用することを検討してきた. 本方式に関連して, CSS2022 では楕円 Lifted-ElGamal 暗号を用いた音声通話の実現性について評価，報告を行なっている [3]．本論文では上記で報告した方式を用いて, 準同型暗号を用いたウェブ音声通話システムを実装し，検証する.

1.3 本論文の構成

本論文の構成を以下に示す．2 章では準備として, ウェブ会議システムの構成, 音声データおよび加法準同型暗号の楕円 Lifted-ElGamal 暗号と並列処理について述べる．3 章では従来方式の課題を確認し, 本研究のアイデアとその実現性ついて示す．4 章では今回の主題であるプロトタイプ実装のシステム全体の構成について説明したのち, クライアント端末とサーバ端末それぞれの動作について説明する．また，今回の実装で用いた環境とライブラリについてもここで述べる．5 章では, 今回の実装の評価のため，暗号演算の処理速度をまとめる．最後に 6 章で考察と今後の課題を示した上で, 7 章でまとめとする.

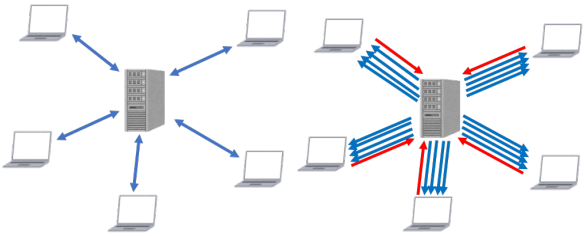
2. 準備

2.1 ウェブ会議システムの構成

ウェブ会議システムは, Peer to Peer(P2P) 通信を基本として構成されていた．P2P 方式はネットワークに接続されたコンピュータ同士が対等の立場，機能で直接通信を行うもので, 会議に接続するクライアントの数が増加しても, 特定クライアントへのアクセス集中が発生しにくいメリットを持つ．一方で, 複数ユーザ間で通信を行う場合, フルメッシュ構造を構成することになり, 各端末の負荷が重くなり通信品質が劣化する．そのため, 複数クライアントの接続が前提となるウェブ会議システムの構成においては P2P 方式ではなくクライアント-サーバ方式が広く用いられる．クライアント-サーバ方式はユーザの持つ端末同士を中継するクライアント・サーバを介してデータの送受信を行うことで回線の問題を解決する．現在普及しているウェブ会議サービスは WebRTC[4] などの API, Zoom Meeting や Cisco Webex の場合は独自のプロトコルを用いて実装されるが, いずれもクライアント・サーバ方式を用いている．クライアント・サーバ方式には Multipoint Control Unit(MCU) 方式と Selective Forwarding Unit(SFU) 方式がある．これらの方式を図 1 に示し, それぞれ説明する.

MCU 方式はクライアントから集めた音声や映像などのデータをサーバ側で処理してクライアント側に流す方式である (図 1a)．クライアントが取り扱うデータや通信に乗るデータのサイズが接続人数によらず常に一定になることから, 最大同時接続可能人数は問題ないが, サーバでデータの重ね合わせを行う際に平文が必要なことから E2EE とは相性が悪い.

SFU 方式はサーバがクライアントから集めたデータを別のクライアントに受け流す中継点となる方式である (図 1b)．サーバ側は平文を必要とする処理を行わないため E2EE はできるが, クライアント端末は同時接続ユーザ数に比例した量の音声・動画を同時再生することとなるため, スケーラビリティを保つことができない．2021 年 12 月現在の E2EE 時同時接続可能最大人数は 200 人 (Zoom meeting, Webex) である.



(a) MCU 方式.　　(b) SFU 方式.

図 1: WebRTC の通信方式.

2.2 音声データ

音声は符号化され音声ファイル形式に保存される．音声ファイル形式には符号化した音声をそのまま保存する非圧縮形式と, 圧縮して保存する圧縮形式がある．WAV 形式 (RIFF waveform Audio Format)[5] は通常非圧縮, リニア PCM のサンプリングデータ用のフォーマットである．リニア PCM 方式は符号化方式の一つで, サンプリングされた音声データは時間に対して線形に保存される．WAV 形式は, その本体である音声データ部分が 16 ビットを 1 単位としたブロックになっている．WAV 形式の音声データ本体はブロックごとに加算することで 2 つ以上の WAV ファイルを 1 つに重ね合わせることができる.

2.3 楕円 Lifted-ElGamal 暗号

Lifted-ElGamal 暗号 [6] は, ElGamal 暗号 [7] を基本とした乗法準同型暗号である．楕円 Lifted-ElGamal 暗号は乗法準同型暗号である ElGamal 暗号方式に, 楕円曲線を組み合わせることで, 加法準同型暗号にしている．セキュリティパラメータを k, 平文空間を $\mathcal{M}$ とする．このとき, 楕円 Lifted-ElGamal 暗号は以下の 4 つのアルゴリズムにより定められる.

Gen: 素数位数 p の楕円曲線上の巡回群を $\mathbf{G} = \langle P \rangle$ とする．s を $0 \leq s < p$ の整数とし, これを秘密鍵とする．また, sP を公開鍵とする.

Enc: 平文を m とする．乱数 r を $0 \leq r < p$ からとり, $Enc(m) = (mP + rsP, rP)$ とする.

Dec: $c = (S, T)$ に対して, $S - sT = (mP + rsP) -$

Vol.2022-CSEC-97 No.11
Vol.2022-IOT-57 No.11
2022/5/19

$s(rP) = mP$ を計算する．最後に離散対数問題 (Discrete logarithm Problem, DLP) を解いて m を得る．

Eval: 2 個の平文 m_1, m_2 に対応する暗号文 $C_1 = Enc(m_1)$, $C_2 = Enc(m_2)$ の各成分を加算し，加算結果として C を出力する．

$$
\begin{aligned}
&Enc(m_1) + Enc(m_2) \\
&= (m_1P + r_1sP, r_1P) + (m_2P + r_2sP, r_2P) \\
&= ((m_1 + m_2)P + (r_1 + r_2)sP, (r_1 + r_2)P) \\
&= Enc(m_1 + m_2)
\end{aligned}
$$

以上より，加法準同型性が確認できる．

2.4 並列処理

本研究ではクライアント端末における暗号化及び復号処理とサーバにおける加算処理の高速化のため OpenMP[8] を使用した並列計算を行う．OpenMP は非営利団体 OpenMP Architecture Review Board(ARB) によって規定されている業界標準規格である．共有メモリ型並列計算機用のプログラムの並列化を記述するための指示文，ライブラリ関数，環境変数などを規格化したもので，これを用いることでユーザはマルチスレッドプログラムを平易なコードで記述することができる．例としてアルゴリズム 1 に OpenMP で for 文によるループ計算を並列化する際の記述を示す．なお，OpenMP は C 言語及び Fortran 言語での実装が可能だが，ここでは C 言語で実装する場合を示す．また，並列処理で使用するスレッド数は環境変数 OMP_NUM_THREADS で指定する．

アルゴリズム 1 OpenMP 利用の例

```
#pragma omp parallel for
for(i = 0; i < N; i++){
    (ループさせたい演算)
}
```

3. 提案手法

3.1 従来法の課題

従来のクライアント・サーバ方式で E2EE と最大同時接続可能人数の増加の両立を図る一般的な方法は SFU 方式のデータ通信量の効率化が主であるが，この方法についても最終的にはクライアント端末の性能および通信経路に乗せるデータサイズがボトルネックとなる．また，データ通信量の効率化を行うにあたってはユーザ端末における処理が複雑になるため，独自アプリケーションを導入する必要があり，MCU 方式と比較して利用環境の柔軟性が劣る．同時接続人数が 200 人を超える大規模なウェブ会議の E2EE を実現するためには，別の手段を用いる必要がある．

3.2 提案法

我々は MCU 方式のウェブ会議システムと準同型暗号方式の組み合わせによる，E2EE ウェブ会議の構成を提案する．すなわち，データ本体を準同型暗号方式で暗号化することで，MCU 方式のウェブ会議システムにおいてサーバに鍵や平文を渡すことなくデータの処理を行うことができるようにする．MCU 方式は 2.1 項で示したように，同時接続ユーザ数が増加してもユーザ端末の負担が変化しないという特徴をもつ．本提案方式は，ウェブ会議システムにおける E2EE と最大同時接続可能人数の両立の困難性を解決する．なお，本提案方式では暗号文の加算者の制限については言及していない．安全性を確保するためには江村らが示すような準同型演算が可能な者を制限することができる機能を持つ暗号方式 [9] を用いる必要があるが，本稿では触れない．

3.3 提案法の実現性

MCU 方式と準同型暗号方式を組み合わせたウェブ会議システムを検討するにあたり問題となのは処理速度である．我々は [3] にて，加法準同型性を持つ楕円 Lifted-ElGamal 暗号を用いた暗号化済み非圧縮音声の重ね合わせの実験を行い，その処理速度が音声通話に適用できる性能であることを示した．また，データサイズ及びレイテンシについては机上検討を行い，音声通話が可能であることを示した．本稿では，これらの方式に基づき，実際に 2 ユーザ以上の E2EE 音声通話システムを構成する．

4. プロトタイプの実装

4.1 システム全体構成

準同型暗号を用いた E2EE 音声通話システムの構成を図 2 に示す．システムはサーバ端末一台と複数のクライアント端末で構成される．ここでは簡単のため，クライアント端末は 2 台のみの記載になっているが，実際には 1 台以上あればよい．また，秘密鍵はクライアント端末間で事前に共有しているものとし，サーバ端末には鍵を秘匿しているものとする．

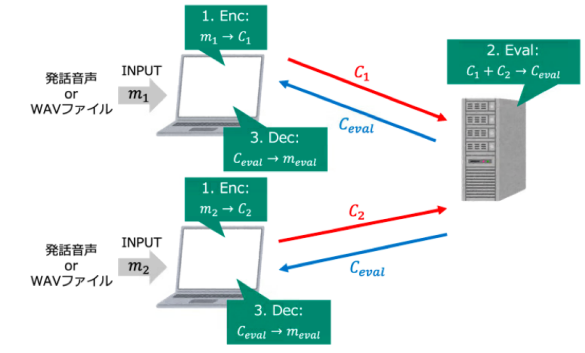


図 2: システム概要．

以下, 4.2 項でクライアント端末の動作, 4.3 項でサーバ端末の動作についてそれぞれ述べる.

4.2 クライアント端末の動作

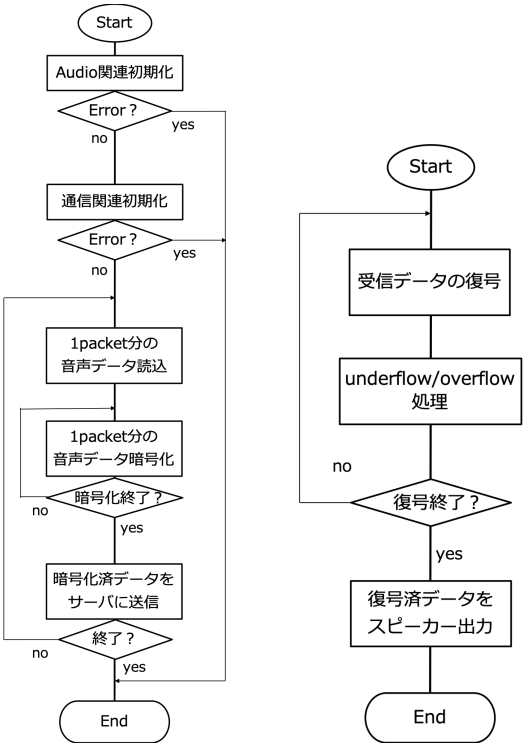
クライアント端末の動作はメイン処理と受信イベント処理の二つに分けられる (図 3).

メイン処理はクライアント端末の所有者の発話音声を暗号化し，サーバに送る処理である．メイン処理のフローチャートを図 3a に示す. クライアント端末は, 起動と共にオーディオ関連と通信関連の初期化を行う. オーディオの初期化はマイク入力の開始を含む. 初期化が完了したら, 入力から 1 パケットごとに読み込み，音声データの暗号化を行う. 暗号化処理が完了したデータはサーバに送信し, 次のパケットの処理に移る. 本フローにおいて, 1 パケット分の音声データ暗号化処理箇所は OpenMP を用いた並列処理とし, 1~4 の範囲で変化させた.

受信イベント処理はサーバから戻ってきたパケットを受信し, 復号して再生する処理である. 受信イベント処理のフローチャートを図 3b に示す. 本処理はメイン処理が既に動作している状態のものとし, よってオーディオ，通信関連の初期化は完了していることを前提とする. 受信したデータを 1 パケットごとに復号する. 復号した値は加算によって WAV ファイルの表現可能範囲を超える可能性があるため, アンダーフロー・オーバーフローの判定及び処理を行う．ここまでが正常に完了したパケットについて，最後にスピーカーから出力する. 本フローにおいて, 1 パケット分の音声データ復号処理箇所は OpenMP を用いた並列処理とし, 1~4 の範囲で変化させた.

4.3 サーバ端末の動作

サーバ端末の基本動作は暗号文の加算処理である. 図?? にサーバ端末の動作フローチャートを示す. サーバ端末は起動時に通信関連の初期化を行う．また, 加算処理のため暗号化関連の初期化を行う. 次にクライアントからのデータ受信の待ち合わせを行う．受信データの待ち合わせ時間は 20ms とした. これは 1 パケットあたりに詰められる音声データの実時間と同じ長さである. 受信待ち合わせ時間内に同一クライアントから複数のパケットが届いた場合は, 受信タイミングが早いものを優先し, 後続のパケットはユーザごとのキューに一時保存する. また, 受信待ち合わせ時間内に全てのクライアントからデータを受信できなかった場合には，受信したクライアントのデータのみで次の動作に移行する. 待ち受け時間が経過したら, 暗号文加算処理を行う. ここまでの処理が完了したら, 加算暗号文をクライアントに送信する処理を行い, 加算処理のため暗号化関連の初期化に戻る. 本フローにおいて, 1 パケット分の暗号化音声データ加算処理は OpenMP を用いた並列処理とし, 1~8 の範囲で変化させた.



(a) メイン処理.　(b) 受信イベント処理.

図 3: クライアント端末における処理.

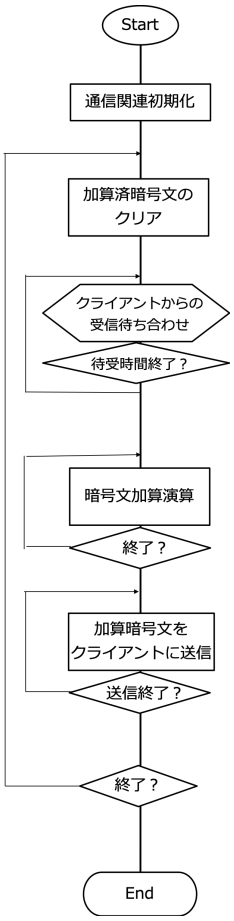


図 4: サーバ処理.

4.4 使用ライブラリ

実装に際しては下記のライブラリを使用した.

- mcl：ペアリング暗号ライブラリ [10]
- openssl-1.1.1m：暗号ライブラリ [11]
- gmp：算術ライブラリ [12]
- boost：Boost c++ライブラリ [13]
- websocketpp：websocket c++ ライブラリ [14]
- alsa：オーディオ制御ライブラリ [15]
- tclap：コマンドライン引数パーサー

mcl ライブラリでは楕円曲線として secp256k1[16] を使用した.

4.5 使用環境

本実験ではサーバ端末及びクライアント端末として下記の性能の端末を使用した．クライアント端末は同一の端末を 2 台準備し，それぞれをクライアント 1, 2 として使用した．また, サーバ端末，クライアント端末共にハイパースレッディングは使用しない.

4.5.1 サーバ端末

- CPU：AMD Ryzen 7 PRO 4750G
- 動作周波数：1.4 GHz
- コア数：8
- OS：Ubuntu20.04 LTS

4.5.2 クライアント端末

- CPU：11th Gen Intel Core i5-11320H
- 動作周波数：2.5 GHz
- コア数：4
- OS：Ubuntu20.04 LTS

5. 測定結果と評価

クライアント, サーバにおいて 1 パケットあたりの暗号処理速度を測定した. なお, 1 パケットあたりの音声データは 20ms とした．すなわち, 暗号化の際の最小ユニットサイズは 4 byte であり，これを 1 ブロックとすると, 1 パケットあたりに含まれる音声ブロックの数は, サンプリング周波数が 8kHz の時は 160 個, 16kHz の時は 320 個, 48kHz の時は 960 個となる．これらのブロック数は 1 パケットあたりの暗号化，加算，復号の処理回数に等しい．以上を前提として, クライアント端末，サーバ端末における 1 パケットあたりの演算処理にかかった時間を評価する.

5.1 クライアント処理速度

クライアント端末で行う暗号関連の処理は 4.2 項に記載の通り, 入力音声の暗号化とサーバから受信した音声の復号処理である.

表 1 に入力音声データ 1 パケットあたりの暗号化処理の平均処理時間をまとめた．測定はサンプリング周波数を 8, 16, 48kHz，暗号化処理の並列スレッド数を 1, 2, 4 に変更し, それぞれについて測定した．これらの結果を図 5 にグラフとして示す．図 5 より, 入力音声のサンプリング周波数が大きいほど，1 パケットあたりの平均処理時間が大きくなることがわかる．また，同一サンプリング周波数のデータを比較した時，並列スレッド数の増加によって処理時間が減少しているのがわかる．暗号化の処理が遅延しないための最低目標となる処理時間は 1 パケットあたりのデータ取得時間である 20ms であるが, 最も処理時間のかかる 48kHz 音声の暗号化処理でも 0.8ms で最低速度を大きく下回っている.

表 1: クライアント端末における暗号化処理速度

		クライアント端末並列数		
		1	2	4
サンプリング	8	1541	965	761
周波数	16	7385	1846	1310
[kHz]	48	8350	4381	3172

[μs]

暗号化処理と同様に, 表 2 に入力音声データ 1 パケットあたりの復号処理の平均処理時間をまとめた．測定はサン

Vol.2022-CSEC-97 No.11
Vol.2022-IOT-57 No.11
2022/5/19

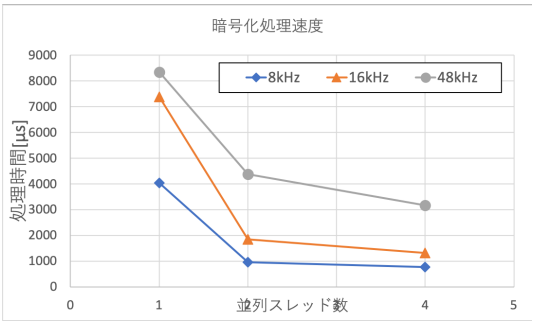


図 5: 1 パケットあたりの暗号化処理時間.

プリング周波数を 8, 16, 48kHz，暗号化処理の並列スレッド数を 1, 2, 4 に変更し，それぞれについて測定した．これらの結果を図 6 にグラフとして示す．図 6 より，入力音声のサンプリング周波数が大きいほど，1 パケットあたりの平均処理時間が大きくなることがわかる．また，同一サンプリング周波数のデータを比較した時，並列スレッド数の増加によって処理時間が減少しているのがわかる．暗号化の処理が遅延しないための最低目標となる処理時間は 1 パケットあたりのデータ取得時間である 20ms であるが，最も処理時間のかかる 48kHz 音声の暗号化処理でも 0.3ms で最低速度を大きく下回っている．

クライアント端末で行われる暗号化，復号の処理速度を比較すると，サンプリング周波数の変更及び並列スレッド数の変更による処理時間の変化の傾向は似ている．また，暗号化速度が復号速度に対して 3 倍程度大きいが，これは 2.3 項より妥当である．

表 2: クライアント端末における復号処理速度

		クライアント端末並列数		
		1	2	4
サンプリング	8	4035	446	293
周波数	16	2747	765	503
[kHz]	48	2730	1711	1243

[μs]

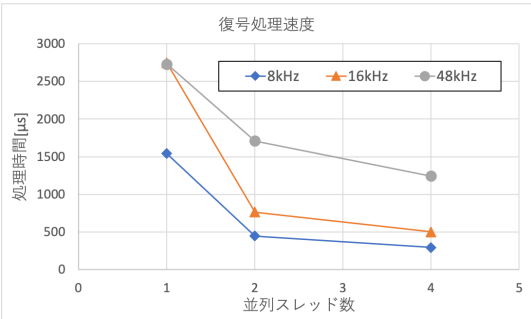


図 6: 1 パケットあたりの復号処理時間.

5.2 サーバ処理速度

表 3 に 2 ユーザ接続時の入力音声データ 1 パケットあたりの加算処理の平均処理時間をまとめた．測定はサンプリング周波数を 8, 16, 48kHz，暗号化処理の並列スレッド数を 1, 2, 4, 8 に変更し，それぞれについて測定した．これらの結果を図 7 にグラフとして示す．図 7 より，入力音声のサンプリング周波数が大きいほど，1 パケットあたりの平均処理時間が大きくなることがわかる．また，同一サンプリング周波数のデータを比較した時，並列スレッド数の増加によって処理時間が減少しているのがわかる．加算の処理が遅延しないための最低目標となる処理時間は 1 パケットあたりのデータ取得時間である 20ms であるが，2 ユーザ接続時の加算時間は暗号化，復号の処理と比較して非常に小さく，遅延の問題はない．

表 3: サーバ端末における加算処理速度

		クライアント端末並列数			
		1	2	4	8
サンプリング	8	84	44	23	16
周波数	16	164	86	44	28
[kHz]	48	404	201	107	77

[μs]

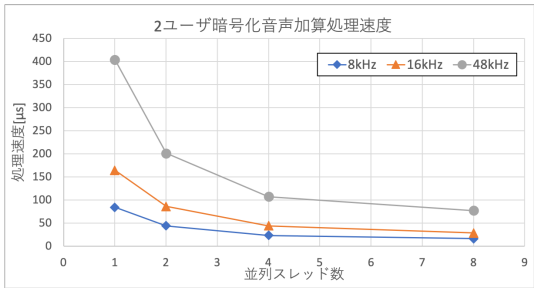


図 7: 2 ユーザ接続時のサーバにおける加算時間.

6. 考察と今後の課題

5 節より，準同型暗号に関連する演算処理による遅延は生じていないことがわかった．実装した準同型暗号 E2EE 音声通話システムの使用感としても，音声が乱れたり，途切れたりすることもなく，通話が可能であった．一方で，発話音声がもう一方のユーザに届くまでのタイムラグ，すなわちレイテンシは体感として 1 秒近くあるように感じた．本稿では議論を行なっていないが，レイテンシについては ITU-T G.114[17] に定められるように最良で 150ms，最悪でも 400ms 以下に収める必要がある．レイテンシの要因には暗号演算処理，通信遅延，その他の実装によるものが含まれる．暗号演算処理による遅延は十分小さい．また，同一端末内にサーバとクライアントの両方を立ち上げて，通

信経路における遅延を取り除いた場合にも遅延が感じられた．以上のことから，現在のプロトタイプ実装でレイテンシが大きく感じられる要因は暗号演算と通信以外の，実装周りにあると考えられる．より実用に近いものを作るためには, 遅延の原因の精査及び改善検討を行なっていく必要がある．

7. おわりに

E2EE と多人数の同時接続可能性を両立したウェブ会議システムを実現するため, Multipoint Control Unit (MCU) 方式に準同型暗号を適用し, 準同型暗号を用いた暗号化メディアデータのリアルタイム重ね合わせの検討を行なってきた．本研究ではウェブ会議音声通話について準同型暗号方式 (楕円 Lifted-ElGamal 暗号方式) を用いた音声通話システムの構築、実装を行った．結果として，リアルタイム音声通話が可能なことがわかったが，実際にサービス導入するためにはレイテンシの性能がボトルネックになりそうなことがわかった．

参考文献

[1] G. Greenwald. ”No Place to Hide: Edward Snowden, the NSA, and the U.S. Surveillance State”, Metropolitan Books, USA, 2014.
[2] Josh Blum, et al., ”E2E Encryption for Zoom Meetings”, https://github.com/zoom/zoom-e2e-whitepaper, 2020.
[3] 上野真奈, 光成滋生, 小林鉄太郎, 村上啓造,“準同型暗号を用いたスケーラブルかつ E2EE な音声重ね合わせの実装,” CSS2021，1E3-1.
[4] WebRTC Org., https://webrtc.org/
[5] Microsoft, ”Extensible Wave-Format Descriptors”, https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/extensible-wave-format-descriptors
[6] 光成 滋生,「クラウドを支えるこれからの暗号技術」, 秀和システム, 2015.
[7] T. ElGamal, A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms, IEEE Transactions on Information Theory, Vol.31, No.4, pp.469-472, 1985, doi:10.1109/TIT.1985.1057074.
[8] OpenMP HP, ”https://www.openmp.org/”
[9] Keita Emura, Goichiro Hanaoka, Go Ohtake, Takahiro Matsuda, and Shota Yamada, “Chosen Ciphertext Secure Keyed-Homomorphic Public-Key Encryption,” Designs, Codes, and Cryptography，vol. 86, issue 8, pages 1623–1683, 2018
[10] mcl ライブラリ, https://github.com/herumi/mcl
[11] Openssl arg., https://www.openssl.org
[12] gmplib org., https://gmplib.org/
[13] boost org., https://www.boost.org/
[14] websocketpp HP, https://github.com/zaphoyd/websocketpp
[15] alsa-project org., https://www.alsa-project.org/wiki/Main_Page
[16] Standards for Efficient Cryptography Group, ”SEC 2: Recommended Elliptic Curve Domain Parameters”, Certicom Research, 2010, Version 2.0.
[17] ITU-T, Rec. G.114, https://www.itu.int/rec/T-REC-G.114