放送サービスに適したセキュアメッセージング用グループ鍵共有

梶田　海成[1,a)]　江村　恵太[2]　小川　一人[2]　野島　良[2]　大竹　剛[1]

概要：セキュアメッセージング（SM）は，信頼できない通信環境でも安全な通信を可能とする技術である．SM は，TLS とは異なり，長期間のセッションや非同期性といった特徴を持つ．現在 IETF がセキュアグループメッセージング（SGM）の標準化を進めており，SGM のグループ鍵共有方式として Alwen ら（CRYPTO 2020）の Continuous Group Key Agreement（CGKA）が議論されている．我々は電子署名方式を Alwen らの方式に組み込んだ放送型 CGKA を提案する．Alwen らの CGKA では全ユーザが同じ権限を持つが，放送型 CGKA では，アルゴリズムの処理に認証機能が付与されることで，一人のユーザをグループマネージャー（GM）とし，グループを GM とその他に分け，それぞれ実行可能な処理を柔軟に設定することが可能となる．また，Alwen らの方式と異なり，秘密情報が第三者によって管理されることはない．放送型 CGKA ではグループ内のユーザの権限を設定可能であることから，例えば放送分野と親和性が高く，新たな放送サービスの実現へ繋がる可能性がある．

キーワード：セキュアメッセージング，Continuous Group Key Agreement，電子署名，公開鍵基盤

Key Agreement of Secure Group Messaging for Broadcasting Services

Kaisei Kajita[1,a)]　Keita Emura[2]　Kazuto Ogawa[2]　Ryo Nojima[2]　Go Ohtake[1]

Abstract: Secure messaging (SM) protocols allows users to communicate securely over untrusted infrastructure. In contrast to other secure communication protocols such as TLS, SM sessions may be long-lived and are asynchronous. The IETF currently works on the standardization of secure group messaging (SGM), and continuous group key agreement (CGKA) proposed by Alwen et al. (CRYPTO 2020) is discussed. We propose a broadcast CGKA protocol that incorporates a digital signature scheme into Alwen et al.'s scheme. All of the group members have the same right in Alwen et al.'s CGKA protocol, but in our protocol, one of the group members is set as a group manager (GM) and the group members are divided into GM and users by adding an authentication function to CGKA. In addition, a third party does not manage confidential information of users unlike to Alwen et al's CGKA protocol. The above characteristics of our protocol may lead to create new broadcasting services.

Keywords: Continuous group key agreement，Secure messaging, Digital signature，PKI functionality

1. はじめに

1.1 背景

セキュアメッセージング（SM）は，信頼できない通信環境においてもユーザの端末間でセキュリティを担保し，安全な通信を可能とする技術である．SM には，TLS や SSH

[1] 日本放送協会
Japan Broadcasting Corporation
[2] 国立研究開発法人情報通信研究機構
National Institute of Information and Communications Technology
[a)] kajita.k-bu@nhk.or.jp

などの一般的なセキュア通信プロトコルとは異なる性質がある．まず，ユーザーが通信から遮断されオフライン状態でのメッセージ交換を想定し，さらに長期間の通信セッション（例えば，スマートフォンを買ってから捨てるまで）を想定した設計になっている．したがって，一般的なセキュア通信プロトコルよりもユーザーの秘密情報の漏洩に対して強い安全性を考慮する必要がある．そのため SM は，前方秘匿性（FS）と漏洩後秘匿性（PCS）を満たすことが安全性要件となっている．前方秘匿性は，ユーザーの鍵が漏洩した場合に，その時点より過去のメッセージが秘匿されることを意味し，漏洩後秘匿性は，ユーザーの鍵が漏洩した場合に，プロトコルを続けることで安全性を回復することを意味する．

SM の動向: 近年，Marlinspike と Perrin[3] が提案したダブルラチェットプロトコルによって 2 パーティの SM プロトコルの設計・解析が進んでいる．ダブルラチェットプロトコルは，Signal[2] をはじめ，多くのメッセンジャーアプリケーションで利用されている暗号化方式である．ダブルラチェットプロトコルは 2 パーティを想定して設計されており，2 人以上のユーザがいるグループでは，全てのユーザのペアでプロトコルを実行しなければならないため，鍵情報を更新するための計算量は，グループサイズ n の 2 乗に比例して増加する．したがって，ダブルラチェットは多数のユーザがいる環境では非効率であり，動作困難であることが知られている [5].

SGM の動向: ユーザが n 人 ($n \geq 2$) 以上いる場合の SM をセキュアグループメッセージング（SGM）という．現在，Internet Engineering Task Force（IETF）では，Messaging Layer Security（MLS）ワーキンググループを立ち上げ，同名の MLS 規格として SGM に関する標準化を目指している．MLS 規格の中心技術となるのが，TreeKEM と呼ばれる，複数人でセキュアメッセージング用の鍵共有を行うための Continuous Group Key Agreement（CGKA）プロトコルである．MLS 規格では，CGKA プロトコルの他に，疑似ランダム関数-疑似ランダム生成器 (PRF-PRNG)，n パーティの前方秘匿認証付き暗号（FS-AEAD），電子署名によって SGM が構成される．本稿では，MLS の安全性・効率性に大きく関わる CGKA プロトコルに焦点を当てる．CGKA プロトコルは，エポックと呼ばれる処理単位において，グループメンバーの削除・追加などの処理を行いながら，非同期的にグループ鍵となるランダム値を生成する．新しいグループ鍵は，高次レイヤーのアプリケーションメッセージ（チャットのテキストなど）を暗号化するために使用される．日本国内で広く普及している LINE で用いられている Letter Sealing もグループ機能を有するが，2 パーティがベースとなっており，グループ内におけるユーザ追加・削除が行われる度にグループ鍵をはじめから作成する必要があるため，グループサイズが大きく制限されている [1]．一方，CGKA プロトコルではグループ鍵を効率的に更新可能であり，例えば放送サービスなど，グループサイズが大きい場合に特に効果を発揮することが期待される．

CGKA の問題点: CGKA プロトコルでは誰でも他のユーザの追加や削除などが可能であるなど，グループ内のメンバーが持つ権限は同じであり，放送サービスのように一対多のグループ構成を持つようなシステムの構成が難しいことが課題である．また，MLS で議論されている CGKA プロトコルでは，公開鍵基盤（PKI）を信頼できる第三者と仮定し，グループメンバーが使用する一次的な鍵を生成する際に PKI を用いて実現している．PKI に全ユーザの秘密鍵を記録する構成であることから，もし PKI の情報が漏洩した場合，グループが生成されて PKI の記録する秘密鍵が更新されるまでの間，全ユーザのメッセージを復号できてしまう．この性質は，実運用において脅威になる可能性が低いが，信頼できる第三者の存在を仮定することはシステムセキュリティ上好ましくない．したがって，一対多のグループ構成を目指す場合においては，グループ内にグループマネージャー（GM）のような管理者を設置し，PKI を用いない構成が望ましい．

1.2 貢献

本稿では，SGM の放送分野への応用を目指し，Alwen ら [5] の CGKA プロトコルに焦点を当てる．我々は, CGKA プロトコルで生成されるコントロールメッセージに新たに電子署名を付与することで，グループ内の一人のユーザをグループマネージャー（GM）として見なし，グループ内のメンバーを GM とその他のユーザに区別する．これにより，GM 以外のユーザが特定の操作のみ実行できる権限を安全に提供可能な放送型 CGKA プロトコルを実現する．CGKA プロトコルのどのアルゴリズムに電子署名を付与するかによって，柔軟にグループ内の権限を設定することができる．我々の提案する CGKA プロトコルは，放送型暗号 [11] やマルチキャスト暗号 [9] と同様に GM が存在する構成であるが，それらとは異なり，各ユーザーがグループ鍵を更新可能であるため，SGM において必須の安全性である FS と PCS を満たすことができる．

また，従来の CGKA では，グループメンバーが使用する一次的な鍵を生成する際に PKI を用いて実現している．我々の放送型 CGKA は，GM が存在する一対多の構成であるから，これまで PKI が行っていた秘密鍵の管理を GM が行い，各グループメンバーは GM へ鍵を問い合わせるようにプロトコルを修正する．これにより PKI を用いずに実行可能なプロトコルを構成する．ここで，PKI と GM が秘密鍵の管理を行うことには次の違いがある．グループメンバーの秘密鍵を PKI が保有すると，その秘密鍵が更新されるまでグループ外にメッセージを復号できる存在があ

ることを意味する．一方，GM が PKI の機能を持ったとしても，GM はグループ内に属しているため元々メッセージを復号できるエンティティであることから，情報漏洩のリスクは GM の PKI 機能の有無に関係しない．したがって，GM が PKI の機能を持つことには明確な利点がある．

応用: 放送型 CGKA は，GM とその他のユーザーが存在し，柔軟なグループ構成および安全かつ効率的な鍵共有が可能であることから，放送分野など一対多の構成を持つサービスと親和性が高く，新たな放送サービスの実現へ繋がる可能性がある．例えば，テレビ収録等におけるリモート観覧・出演等において，放送局と視聴者で一対多を構成する．リモート観覧・出演では，視聴者はテレビ収録にリモートで参加（観覧）し，放送局は視聴者に収録の映像を送る．ここで，例えば放送局はグループ構成や不正な視聴者の削除を行い，視聴者は別の視聴者の追加や自身の離脱を行う、といった柔軟な制御が可能となるように，電子署名を各アルゴリズムに適切に導入する．このとき，グループは番組毎に構成され長期間継続するため，鍵更新の機能が安全性を担保する上で効果的である．現在でも SM を用いたリモート観覧・出演は行われているが，放送型 CGKA を用いることで，より柔軟なグループ構成および安全かつ効率的な鍵共有を可能としたセキュアシステムを構成できる．

1.3 関連研究

ダブルラチェットは，OTR (off-the-record) プロトコル [6] のアイデアをもとに，2016 年に Marlinspike と Perrin[3] によって提案された．Alwen ら [4] は，これまで曖昧であったダブルラチェットアルゴリズムのモジュール設計とセキュアメッセージング用の 2 パーティでの鍵共有を Continuous Key Agreement（CKA）として厳密に定義した．2020 年に Alwen ら [5] は IETF MLS で議論されている TreeKEM を CGKA プロトコルとして定義し，IETF MLS の TreeKEM の Forward Secrecy の脆弱性を指摘し，Updatable Encryption を用いてその安全性を改良した．SGM においては，Cremers ら [8] が複数パーティにおける PCS の問題点を指摘し，Weider[14] は，セキュアメッセージング以外への分野へ応用するために通常の TreeKEM を改良して汎用的なグループ鍵共有プロトコルである Casual TreeKEM を提案した．また，MLS のメーリングリストにおいても，新しい TreeKEM が提案されている [7], [13].

Fiat と Naor が提案した（共通鍵）放送型暗号（Broadcast Encryption)[11] では，SGM と異なり，信頼されたグループマネージャー（GM）がメッセージの内容だけでなく，全ての秘密鍵を配布する．さらに，GMがユーザを自由に追加したり，参加を取り消したりすることができるため，ユーザは秘密鍵を更新する必要がない．Dodis と Fazio[10] によって導入された公開鍵ベースの放送型暗号では，誰でもコンテンツを配信することができるが，復号できる利用者のグループ管理は信頼された第三者によって行われる．一方，マルチキャスト暗号 [9], [12] は，放送型暗号の効率性を改良した方式であるが，GM はユーザーを追加または削除するための「メンバーシップ更新情報」を送信するモデルである．放送型暗号やマルチキャスト暗号は，グループ内で鍵を共有する点は SGM と共通する一方，GM が存在し，さらに安全性要件である FS と PCS は考慮されていない．

2. 準備

本稿で使用する表記及び暗号学的プリミティブを説明する．正整数 a において，$[a]$ を集合 $\{1, 2, \ldots, a\}$ とする．整数 n において，$\mathsf{mp2}(n)$ を n の約数の内，最大の 2 のべき乗とする．本稿におけるセキュリティゲームは辞書を伴って行われる．辞書 D において，鍵 x によって記録された値を $D[x]$ とする．任意の初期値 y において，辞書 D を初期化する記述を $D[\cdot] \leftarrow y$ とする．

2.1 2 分木

全てのノードが 0 から 2 つの子ノードを持つ根付き二分木では，子ノードを持たないノードを葉ノード，その他のノードを中間ノードと呼ぶ．二分木の高さとは，根（ルート）にあたる頂点から葉ノードまでの最大のパス長である．高さが h で，2^h 個の葉ノードを持つとき，二分木は完全であるという．特に，高さ $h = 0$ の二分木は根のみから成る．$h \geq 0$ において，FT_h を高さ h の完全二分木とする．ある葉ノード ℓ, ℓ' において，$\mathsf{LCA}(\ell, \ell')$ を ℓ, ℓ' の最近共通祖先，すなわち，それぞれの葉ノードから根へのパスが交わるノードとする．

Left-Balanced Binary Tree (LBBT): ノード数 $n \in \mathbb{N}$ の LBBT_n は，次の性質を満たす二分木である．(1) LBBT_1 は一つのノード（根）から成る．(2) $x = \mathsf{mp2}(n)$ とするとき，LBBT_n の根が，左の子ノードに FT_x を持ち，右の子ノードに LBBT_{n-x} を持つ．

$\mathsf{LBBT}_n = \tau$ とする．このとき，τ に ID がラベル付けされたノードを $\tau.ID$ とする．また，v を τ のノードとするとき，v にラベル付けされた pk を $v.\mathsf{pk}$ とする．

2.2 電子署名

電子署名 Σ は 3 つのアルゴリズム $(\mathsf{KGen}_\Sigma, \mathsf{Sign}_\Sigma, \mathsf{Vrfy}_\Sigma)$ から成る．鍵生成アルゴリズム KGen_Σ は，セキュリティパラメータ λ を入力にとり，署名鍵 sk_{sig} と，検証鍵 vk_{sig} を出力する．署名アルゴリズム Sign_Σ は署名鍵 sk_{sig} とメッセージ m を入力にとり，署名 σ を生成する．検証アルゴリズム Vrfy_Σ は，検証鍵 vk_{sig} と署名 σ とメッセージ m を入力にとり，検証が受理されれば 1 を，検証が受理されなければ 0 を出力する．攻撃者 $\mathcal{A}$ が EUF-CMA ゲームに勝

つ確率を $\mathsf{Adv}_{cma}(\mathcal{A})$ とする．Σ が時間 t, 偽造成功確率 ϵ とした場合，$\mathsf{Adv}_{\Sigma}(\mathcal{A}) \leq \epsilon$ であるとき，(t,ϵ)-cma-安全という．ただし，本稿において，メッセージはコントロールメッセージに相当することに注意する．

2.3 疑似ランダム生成器

疑似ランダム生成器 $\mathsf{prg} : \mathcal{W} \to \mathcal{W} \times \mathcal{K}$ は，一様ランダムな $U \in \mathcal{W}$ と $U' \in \mathcal{W} \times \mathcal{K}$ において，$\mathsf{prg}(U)$ が U' と識別不可能である関数である．U, U' を識別する攻撃者 $\mathcal{A}$ の成功確率を $\mathsf{Adv}_{\mathsf{prg}}(\mathcal{A})$ とする．prg が時間 t, 成功確率 ϵ とした場合，$\mathsf{Adv}_{\mathsf{prg}}(\mathcal{A}) \leq \epsilon$ であるとき，(t,ϵ)-安全という．

2.4 更新可能暗号

更新可能暗号 UPKE は次の 3 つのアルゴリズム $\mathsf{UPKE} = (\mathsf{UKGen}, \mathsf{UEnc}, \mathsf{UDec})$ から成る．鍵生成アルゴリズム $\mathsf{pk}_0 \leftarrow \mathsf{UKGen}(\mathsf{sk}_0)$ は，一様ランダムな $\mathsf{sk}_0 \leftarrow \mathcal{SK}$ を入力にとり，初期公開鍵 pk_0 を生成する．暗号化アルゴリズム $(c, \mathsf{pk}') \leftarrow \mathsf{UEnc}(\mathsf{pk}, m)$ は，公開鍵 pk によってメッセージ m を暗号化し，暗号文 c を出力し，同時に pk を更新し，更新公開鍵 pk' を出力する．復号アルゴリズム $(m, \mathsf{sk}') \leftarrow \mathsf{UDec}(\mathsf{sk}, c)$ は，秘密鍵 sk によって c を復号し，元の m を出力し，同時に sk を更新し，更新秘密鍵 sk' を出力する．攻撃者 $\mathcal{A}$ が IND-CPA ゲームに勝つ確率を $\mathsf{Adv}_{cpa}(\mathcal{A})$ とする．UPKE が時間 t, 成功確率 ϵ とした場合，$\mathcal{A}$ に対して，$\mathsf{Adv}_{cpa}(\mathcal{A}) \leq \epsilon$ であるとき，(t,ϵ)-CPA-安全という．

3. Continuous Group Key Agreement（CGKA）

本章では，CGKA プロトコルの構成と安全性について記述する．

3.1 CGKA の構造

CGKA プロトコル $\mathsf{CGKA} = (\mathsf{init}, \mathsf{create}, \mathsf{add}, \mathsf{rem}, \mathsf{upd}, \mathsf{proc})$[5] は以下のアルゴリズムから成る．コントロールメッセージには，グループに新たに参加するユーザーに送信される歓迎メッセージ W と，すでにグループに参加しているユーザーに送信される通常のメッセージ T の 2 種類ある．

- init：ユーザの ID ID を入力にとり，ユーザ毎に異なる初期状態 γ を出力する．
- create：状態 γ と ID のリスト $\mathsf{G} = (\mathsf{ID}_1, \ldots, \mathsf{ID}_n)$ を入力にとり，新しい状態 γ' とコントロールメッセージ W を出力する．
- add：状態 γ と ID ID' を入力にとり，新しい状態 γ' とコントロールメッセージ W, T を出力する．
- rem：状態 γ と ID ID' を入力にとり，新しい状態 γ' とコントロールメッセージ T を出力する．
- upd：状態 γ を入力にとり，新しい状態 γ' とコントロールメッセージ T を出力する．
- proc：状態 γ とコントロールメッセージ T を入力にとり，新しい状態 γ' と更新秘密情報 I を出力する．

状態変数 γ は, 各アルゴリズムによって生成される LBBT を保存する．CGKA プロトコルの流れは次の通りである．まずグループに参加する各ユーザは init によって初期化を行う．あるユーザーが create を使用してグループが生成されると，グループメンバーは，メンバーの追加や削除，更新を行うために，いずれかのアルゴリズムを呼び出すことができる．各アルゴリズムは，エポックと呼ばれる処理単位で管理される．結果として生じるコントロールメッセージを（送信者を含む）すべての現在のグループメンバーに送信する．コントロールメッセージが送信される度に，全てのグループメンバーはアルゴリズム proc を実行し更新秘密情報 I を共有する．

3.2 CGKA の安全性

安全性要件: CGKA は次の性質を満たす [5].

- 正当性（Correctness）：全てのグループメンバーは各エポックにおいて同じ更新秘密情報 I を得る．
- 秘密性（Privacy）：コントロールメッセージを受け取ったときに生成される更新秘密情報 I は一様ランダムと区別できない．
- 前方秘匿性（Forward secrecy）：あるグループメンバーの状態が漏洩したとき，それ以前の更新秘密情報は第三者から秘匿される．
- 漏洩後秘匿性（Post-compromise security）：あるグループメンバーの状態が漏洩したとき，状態が漏洩した全てのグループメンバーが更新を行うと再び更新秘密情報は秘匿される．

上記の性質は, [5] において記述される 12 個のオラクルによる CGKA ゲームによって厳密に定義される．CGKA ゲームでは，攻撃者は CGKA プロトコルを実行するためのオラクルへのアクセス権が与えられ，real-or-random チャレンジを行う．

安全性定義: 高々時間 t，チャレンジクエリ数 c，グループメンバー数が n における (t,c,n)-攻撃者を $\mathcal{A}$ とする．CGKA ゲーム [5] において，ランダムビットを正しく推測したとき，攻撃者の勝利（$\mathcal{A}$ wins）とする．このとき，攻撃者 $\mathcal{A}$ の成功する確率を

$$\mathsf{Adv}_{\mathrm{CGKA}}(\mathcal{A}) := \left| \Pr[\mathcal{A} \text{ wins}] - \frac{1}{2} \right|$$

とする．任意の (t,c,n)-攻撃者 $\mathcal{A}$ において，$\mathsf{Adv}_{\mathrm{CGKA}}(\mathcal{A}) \leq \epsilon$ であるとき，CGKA は (t,c,n,ϵ)-安全であるという．

3.3 TreeKEM [5]

IETF の MLS 規格として，SGM 用の鍵共有プロトコル TreeKEM が議論されている．Alwen らは SGM 用の鍵共有

プロトコルを厳密に CGKA プロトコルとして定義し，さらに UPKE を公開鍵暗号の代わりに用いることで TreeKEM の安全性を改良した [5]．TreeKEM プロトコルは，（バイナリー）ラチェットツリー（RT）に基づき構成される．TreeKEM の RT は，グループメンバーは葉ノードに，ルートを除いた全てのノードは公開鍵暗号の鍵ペアがラベル付けされた LBBT である．Alwen らの TreeKEM を**図 1** に示す．ここで，TreeKEM にて使用されるサブアルゴリズム Init, AddID, Blank, Pub, RemID, UpGen, UpPro を以下に示す．

Init$(G, \mathsf{pk}, j, \mathsf{sk}_j)$:

ユーザーのリスト $G = (\mathsf{ID}_0, \mathsf{ID}_1, \ldots, \mathsf{ID}_n)$，公開鍵 $\mathsf{pk} = (\mathsf{pk}_0, \mathsf{pk}_1, \ldots, \mathsf{pk}_n)$，整数 j，秘密鍵 sk_j を入力とし，次を満たす LBBT_{n+1} を新しい RT として初期化する．

- 全ての中間ノードとルートを白紙（blank）とする．
- 全ての葉ノード i は $(\mathsf{ID}_i, \mathsf{pk}_i, \perp)$ がラベル付けされる．
- 葉ノード j には秘密鍵 sk_j が追加でラベル付けされる．

AddID$(\tau, \mathsf{ID}, \mathsf{pk})$:

RTτ，ユーザ ID, pk を入力としてとり，τ の最初の空白のノードに $(\mathsf{ID}, \mathsf{pk}, \perp)$ をラベル付けし，新しい RTτ' を出力する．このとき，$\tau = \mathsf{LBBT}_n$ に空白のノードがないとき，AddLeaf を呼び出す．

AddLeaf(τ):

RT$\tau = \mathsf{LBBT}_n$ を入力にとり，以下のように葉ノード z を加え，新しい RTτ' を出力する．

- n が 2 のべき乗のとき，τ のルートを左の子ノード，z を右の子ノードとなるように，新しいノード r' を τ' に追加する．
- n が 2 のべき乗ではないとき，r を τ のルートとし，τ_L, τ_R をそれぞれ左右の部分木とする．
- 再帰的に，z を τ_R に挿入して新しい τ'_R とし，τ' のルートを r とし，τ_L, τ_R をそれぞれ左右の部分木とする．

Blank(τ, ID):

ユーザ ID ID と RTτ を入力にとり，ID がラベル付けされた葉ノードからルートまでの全てのノードを空白とする．

Pub(τ):

RTτ を入力にとり，全ての秘密鍵のラベルを $\perp$ として τ を複製し，τ' を出力する．

RemID(τ, ID):

τ, ID を入力にとる．ID がラベル付けされた葉ノードを空白にし，最も右側にある空白でない葉ノードが最後のノードとなるように，再帰的な枝刈り処理 Trunc(v) を呼び出す．

Trunc(τ):

τ の最も右側の葉ノード v を入力とし，次を満たす τ' を出力する．

- v が空白かつルートでないとき，v とその親ノードを削除し，兄弟ノード v' を親ノードがあったノードに配置する．Trunc を実行する．
- v が空白でなく，かつルートであるとき，最も右側の葉ノード v'' で Trunc を実行する．
- 上記以外のとき，処理を終える．

UpGen(τ, ID):

RTτ とユーザ ID ID を入力にとる．ユーザ ME は次の用に処理を行い，新しい τ' と更新秘密情報 U を出力する．$\tau' \leftarrow$PropUp(τ, v, s_0) を行い，得られた τ' において，

- パス上の秘密を暗号化する．v からルートまでのパス上の兄弟ノードを $v'_0, \ldots, v'_{d-1}$ とする．全ての値 s_i と全てのノード $v_j \in \mathsf{Res}(\mathsf{v}'_{\mathsf{i}-1})$ において，$c_{ij} \leftarrow \mathsf{UEnc}(v_j.\mathsf{pk}, s_i)$ を実行する．
- 全ての暗号文 c_{ij} と公開鍵 $(\mathsf{pk}_0, \ldots, \mathsf{pk}_{d-1})$ をそれぞれ $\mathbf{c}, \mathsf{PK}$ とする．$U \leftarrow (\mathcal{PK}, \mathbf{c})$ を残りのグループメンバーの更新秘密情報とする．

この処理のコントロールメッセージは単に ME の ID と U とする．

PropUp(τ, v, s_0):

RTτ，葉ノード v，一様ランダムな値 s_0 を入力にとり，あるユーザ ME が新しい鍵ペアを次のように選ぶことで更新を行い，τ' を出力する．

- パス上の秘密鍵を計算する．ME の葉ノード v からルートまでノードを順に $v_0(=v), v_1, \ldots, v_d$ とする．一様ランダムな s_0 を選び，以下を実行する．

$$\mathsf{sk}_i || s_{i+1} \leftarrow \mathsf{prg}(s_i) \quad \text{for } i = 0, \ldots, d-1$$

- RT のラベルを更新する．$i = 0, \ldots, d-1$ において，ME は $\mathsf{pk}_i \leftarrow \mathsf{UKGen}(\mathsf{sk}_i)$ を計算し，v_i の鍵のラベルを $(\mathsf{pk}_i, \mathsf{sk}_i)$ に更新する．
- ルートノードにおいて，$I := s_d$ とする．

UpPro$(\tau, \mathsf{ID}, \mathsf{ID}', U)$:

コントロールメッセージ $T = (\mathsf{up}, \mathsf{ID}, U)$ を処理するとき，葉ノード v のユーザー ID によって発行された U=(pk,$\mathbf{c}$) を受け取った葉ノード ℓ' のユーザー ID$'$ は，$\tau, \mathsf{ID}, \mathsf{ID}', U$ を入力とし，次のように新しい RTτ' を生成する．

- $w = \mathsf{Rep}(v, \ell')$ とする．
- ID$'$ を持つユーザーは $w.sk$ を用いて c_{ij} を復号し，s_i を得る．
- $\mathcal{PK}$ によって v からルートまでのパス上の全ての公開鍵のラベルを上書きし，$\tau' \leftarrow$ PropUp$(\tau,$

TK-init((ID))
- ME ← ID
- $\tau \leftarrow \perp$
- ctr ← 0
- $\tau'[\cdot], \mathsf{conf}[\cdot] \leftarrow \perp$

TK-create(G)
- ctr + +
- $\mathsf{ID}_0 \leftarrow$ ME
- $\mathsf{sk}_0 \leftarrow \mathcal{SK}, \mathsf{pk}_0 \leftarrow$ UKGen
- for $i = 1, \ldots, |G|$
  - $\mathsf{pk}_i \leftarrow$ get-pk($|G|.i$)
- $G' \leftarrow (\mathsf{ID}_0, G)$
- **pk'** $\leftarrow (\mathsf{pk}_0, \mathsf{pk})$
- $\tau'[\mathsf{ctr}] \leftarrow$ INIT$(G', \textbf{pk'}, 0, \mathsf{sk}_0)$
- $W \leftarrow (\mathsf{create}, G', \textbf{pk'})$
- conf[ctr] ← W
- **return** W

TK-add(ID′)
- ctr++
- $\mathsf{pk}' \leftarrow$ get-pk(ID′)
- $\tau'[\mathsf{ctr}] \leftarrow$ ADDID$(\tau, \mathsf{ID}', \mathsf{pk}')$
- $\tau'[\mathsf{ctr}] \leftarrow$ BLANK$(\tau'[\mathsf{ctr}], \mathsf{ID}')$
- $W \leftarrow$ (wel,PUB$(\tau'[\mathsf{ctr}])$)
- $T \leftarrow (\mathsf{add}, \mathsf{ME}, \mathsf{ID}', \mathsf{pk}')$
- conf[ctr] ← T
- **return** (W, T)

TK-rem(ID′)
- ctr++
- $\tau'[\mathsf{ctr}] \leftarrow$ BLANK(τ, ID')
- $\tau'[\mathsf{ctr}] \leftarrow$ REMID$(\tau'[\mathsf{ctr}], \mathsf{ID}')$
- $T \leftarrow (\mathsf{rem}, \mathsf{ME}, \mathsf{ID}')$
- conf[ctr] ← T
- **r**$eturn$ T

TK-upd
- ctr++
- $(\tau'[\mathsf{ctr}], U) \leftarrow$ UPGEN(τ, ME)
- $T \leftarrow (\mathsf{up}, \mathsf{ME}, U)$
- conf[ctr] ← T
- **r**$eturn$ T

TK-proc(T, W)
- if $\exists j : T = \mathsf{conf}[j]$
  - $\tau \leftarrow \tau'[j]$
- else
  - proc(T, W)
- ctr ← 0
- $\tau'[\cdot], \mathsf{conf}[\cdot] \leftarrow \perp$
- **return** $(\tau.I)$

proc($W = (\mathsf{create}, G, \mathsf{pk})$)
- let j s.t. $G.\mathsf{ID}_j =$ ME
- $\mathsf{sk}_j \leftarrow$ get-sk(pk.j)
- $\tau \leftarrow$ INIT$(G, \mathsf{pk}, j, \mathsf{sk}_j)$

proc($T = (\mathsf{add}, \mathsf{ID}, \mathsf{ID}', \mathsf{pk}')$)
- $\tau \leftarrow$ ADDID$(\tau, \mathsf{ID}', \mathsf{pk}')$
- $\tau \leftarrow$ BLANK(τ, ID')

proc($W = (\mathsf{wel}, \tau')$)
- $\tau \leftarrow \tau'$
- $\tau.\mathsf{ME.sk} \leftarrow$ get-sk(τ.ME.pk)

proc($T = (\mathsf{rem}, \mathsf{ID}, \mathsf{ID}')$)
- $\tau \leftarrow$ BLANK(τ, ID')
- $\tau \leftarrow$ REMID(τ, ID')

proc($T = (\mathsf{up}, \mathsf{ID}, U)$)
- $\tau \leftarrow$ UPPRO$(\tau, \mathsf{ID}, \mathsf{ME}, U)$

図 1 TreeKEM プロトコル．ある ID が get-pk(ID′) を呼び出すと，PKI によって (pk, sk) が生成され，pk を ID に返す．このとき，PKI は (pk, sk, ID′) を記録する．ID′ は get-sk(pk) によって PKI に問い合わせを行うことができ，(pk, sk, ID′) が記録されていれば，ID′ に sk を返す．

LCA(v,ℓ'),s_i) によって新しい RT を生成する．

定理 3.1

疑似ランダム生成器 prg が $(t_{prg}, \epsilon_{prg})$-安全であり，更新可能暗号 UPKE が $(t_{cpa}, \epsilon_{cpa})$-安全であるとき，*TreeKEM* は，$(t, c, n, \epsilon)$-適応的安全な *CGKA* である．ただし，$\epsilon = O(cn^{\log n})(\epsilon_{prg} + \epsilon_{cpa})$，$t \approx t_{prg} \approx t_{cpa}$ を満たす．

本稿では，定理 3.1 における安全性ゲームの記述を省略した．厳密な定理とその証明は [5] を参照されたい．

4. 放送型 CGKA プロトコル

我々は Alwen らの CGKA プロトコルである TreeKEM をベースとし，まず電子署名を付与し，ユーザーの認証を可能とすることで，グループ内の一人のユーザをグループマネージャー（GM）と見なし，グループ内のメンバーを GM とその他のユーザに区別する．これにより，GM 以外のユーザが特定の操作のみ実行できる権限を安全に提供可能な一対多の放送型 CGKA プロトコルを実現する．さらに，TreeKEM では，RT の構成のために必要な鍵を生成するために PKI を用いているが，我々の放送型 CGKA は，GM が存在する一対多の構成を持つことから，これまで PKI が行っていた鍵管理を GM が行い，各グループメンバーは GM へ鍵を問い合わせるようにプロトコルを修正する．

4.1 放送型 CGKA

提案方式では，GM を新たに配置し，GM が PKI の機能を行う．このとき，GM は次の要件を満たす必要がある．

- GM はサービスレベルに応じて十分な帯域幅を有し，輻輳が発生しないこと．
- GM は常にオンラインであること．

グループサイズが大きい場合，GM には多くの問い合わせ

が来るため，ネットワークの輻輳を防ぐ必要がある．また，他のユーザーからの問い合わせに迅速に対応するため，常にオンラインである必要がある．例えば GM が放送局のように大規模かつクラウドサーバーの役割を担うとき，これらの要件は容易に満たされる．

安全性要件

放送型 CGKA は，3.2 章の CGKA と同様に，Correctness, Privacy, FS，PCS を満たす．さらに次の性質を満たす．

- Non-impersonation of GM（NIG）：各ユーザは，受け取ったコントロールメッセージが，GM から送られたものであるか識別可能であり，GM へのなりすましが不可能である．

4.2 TreeKEM$^*_\Sigma$ の構成

提案 CGKA プロトコル TreeKEM$^*_\Sigma$ を**図 2** に示す．Alwen らの CGKA プロトコル TreeKEM との違いは，次の二つである．一つ目は，TreeKEM に署名方式 Σ を組み合わることで，各アルゴリズムにおいて出力されるコントロールメッセージに対して署名 σ の生成機能を追加し，処理 proc ではまず署名の検証を行う．これにより，GM のコントロールメッセージにのみ署名を付与することで，GM がアルゴリズムを実行したのか，グループメンバーは確かめることができる．二つ目は，TK-create，TK-add において，従来は PKI が生成した pk を問い合わせていたが，GM が生成する pk を問い合わせ，その処理 proc においても同様に，GM に sk を問い合わせて取得する．これにより，信頼できる第三者エンティティを増やすことなくシステムを構成することができる．また，TreeKEM$^*_\Sigma$ を一般的構成として，SGM の構成要素である CGKA として組み込むことで放送サービスに適した SGM を構成可能である．

提案 CGKA プロトコル TreeKEM$^*_\Sigma$ の安全性を示すために，新たにオラクル **ctrl-chall**(**図 3**) と安全性ゲーム **dis**(**図 4**) を定義する．

オラクル ctrl-chall: $\mathsf{GM}[\cdot]$ を GM の ID を記録する辞書とする．オラクル **ctrl-chall**(t^*) では，実行者 ID が GM の ID_0 であるとき，電子署名 σ を返し，そうでないときは 0 を返すオラクルである．

安全性ゲーム dis: クエリ $q_1, \ldots, q_q$ に対し，攻撃者は，あるエポック t^* において $q_i =$ **ctrl-chall**(t^*) を満たす i 番目のクエリにおいて，署名 σ_i の検証を行う．任意の i において検証式 $\mathsf{Vrfy}(\sigma_i) = 1$ が成り立つとき，放送型 CGKA の安全性要件である NIG を満たす．

TreeKEM$^*_\Sigma$ は，TreeKEM に電子署名を加えた一般的構成であるため，同様に定理 3.1 を満たす．さらに，安全性ゲーム **dis** より，電子署名方式 Σ が $(t_{\mathsf{cma}}, \epsilon_{cma})$-安全であるとき，TreeKEM$^*_\Sigma$ が NIG を満たす．具体的には下記の定理が成り立つ．

定理 4.1

疑似ランダム生成器 prg が $(t_{prg}, \epsilon_{prg})$-安全で，更新可能暗号 UPKE が $(t_{cpa}, \epsilon_{cpa})$-*cpa*-安全で，電子署名方式 Σ が $(t_{\mathsf{cma}}, \epsilon_{cma})$-*cma*-安全であるとき，TreeKEM$^*_\Sigma$ は，(t, c, n, ϵ)-適応的安全な放送型 *CGKA* である．ただし，$\epsilon = O(cn^{\log n})(\epsilon_{prg} + \epsilon_{cpa}) + \epsilon_{cma}$, $t \approx t_{prg} \approx t_{cpa} \approx t_{cma}$ を満たす．

5. まとめ

我々は本稿で電子署名方式を Alwen らの方式に組み込んだ放送型 CGKATreeKEM$^*_\Sigma$ を提案した．Alwen らの CGKA では全ユーザが同じ権限を持つが，TreeKEM$^*_\Sigma$ では，アルゴリズムの処理に認証機能が付与されることで，一対多の構成をとることができる．また，Alwen らの方式と異なり，秘密情報が第三者によって管理されることはない．放送型 CGKA ではグループ内のユーザの権限を設定可能であることから，例えば放送分野と親和性が高く，新たな放送サービスの実現へ繋がる可能性がある．

参考文献

[1] `https://line.me/ja/`

[2] `https://signal.org/docs/`

[3] M. Marlinspike and T. Perrin, “The double ratchet algorithm”, 2016. `https://whispersystems.org/docs/specifications/doubleratchet/doubleratchet.pdf`

[4] Joël Alwen, Sandro Coretti, and Yevgeniy Dodis. “The double ratchet: Security notions, proofs, and modularization for the signal protocol”. In Yuval Ishai and Vincent Rijmen, editors, EUROCRYPT 2019, Part I, volume 11476 of LNCS, pages 129-158. Springer, Heidelberg, May 2019.

[5] Joël Alwen, Sandro Coretti, Yevgeniy Dodis, and Yiannis Tselekounis, “Security analysis and improvements for the IETF MLS standard for group messaging”. In Annual International Cryptology Conference, pp. 248-277. Springer, Cham, 2020.

[6] Nikita Borisov, Ian Goldberg, and Eric A. Brewer. “Off-the-record communication, or, why not to use PGP”. In Proceedings of the 2004 ACM Workshop on Privacy in the Electronic Society, WPES 2004, October 28, pp. 77-84, 2004.

[7] Richard Barnes. Subject: [MLS] Remove without double-join (in TreeKEM). MLS Mailing List. Mon, 06 August 2018 13:01 UTC, 2018. `https://mailarchive.ietf.org/arch/msg/mls/Zzw2tqZC1FCbVZA9LKERsMIQXik`.
Part III, volume 10403 of LNCS, pages 619-650. Springer, Heidelberg, August 2017.

[8] Cas Cremers, Britta Hale, and Konrad Kohbrok. “Revisiting post-compromise security guarantees in group messaging”. Cryptology ePrint Archive, Report 2019/477, 2019. https://eprint.iacr.org/2019/477.

[9] Ran Canetti, Juan A. Garay, Gene Itkis, Daniele Micciancio, Moni Naor, and Benny Pinkas. “Multicast security: A taxonomy and some efficient constructions”. In IEEE INFOCOM’99, pages 708-716, New York, NY, USA, March 21-25, 1999.

TK-init-sig((ID))
- $\mathsf{ME} \leftarrow \mathsf{ID}$
- $\tau \leftarrow \perp$
- $\mathsf{ctr} \leftarrow 0$
- $\tau'[\cdot], \mathsf{conf}[\cdot] \leftarrow \perp$
- $(\mathsf{sk}_{sig}, \mathsf{vk}_{sig}) \leftarrow \mathsf{KGen}_\Sigma$

TK-create-sig(G)
- $\mathsf{ctr}++$
- $\mathsf{ID}_0 \leftarrow \mathsf{ME}$
- $\mathsf{sk}_0 \leftarrow \mathcal{SK}, \mathsf{pk}_0 \leftarrow \mathsf{UKGen}$
- for $i = 1, \ldots, |G|$
  - $\mathsf{sk}_0 \leftarrow \mathcal{SK}, \mathsf{pk}_0 \leftarrow \mathsf{UKGen}$
- $\mathbf{pk'} \leftarrow (\mathsf{pk}_0, \mathsf{pk})$
- $\tau'[\cdot] \leftarrow \textsc{Init}(G', \mathbf{pk'}, 0, \mathsf{sk}_0)$
- $W \leftarrow (\mathsf{create}, G', \mathbf{pk'})$
- $\mathsf{conf}[\mathsf{ctr}] \leftarrow W$
- $\sigma \leftarrow \mathsf{Sign}_\Sigma(W, \mathsf{sk}_{sig})$
- **return** (W, σ)

TK-add-sig(ID')
- $\mathsf{ctr}++$
- $\mathsf{pk}' \leftarrow \mathsf{GMget\text{-}pk}(\mathsf{ID}')$
- $\tau'[\mathsf{ctr}] \leftarrow \textsc{AddID}(\tau, \mathsf{ID}', \mathsf{pk}')$
- $\tau'[\mathsf{ctr}] \leftarrow \textsc{Blank}(\tau'[\mathsf{ctr}], \mathsf{ID}')$
- $W \leftarrow (\mathsf{wel}, \textsc{Pub}(\tau'[\mathsf{ctr}]))$
- $T \leftarrow (\mathsf{add}, \mathsf{ME}, \mathsf{ID}', \mathsf{pk}')$
- $\mathsf{conf}[\mathsf{ctr}] \leftarrow T$
- $\sigma \leftarrow \mathsf{Sign}((W, T), \mathsf{sk}_{sig})$
- **return** $((W, T), \sigma)$

TK-rem-sig(ID')
- $\mathsf{ctr}++$
- $\tau'[\mathsf{ctr}] \leftarrow \textsc{Blank}(\tau, \mathsf{ID}')$
- $\tau'[\mathsf{ctr}] \leftarrow \textsc{RemID}(\tau'[\mathsf{ctr}], \mathsf{ID}')$
- $T \leftarrow (\mathsf{rem}, \mathsf{ME}, \mathsf{ID}')$
- $\mathsf{conf}[\mathsf{ctr}] \leftarrow T$
- $\sigma \leftarrow \mathsf{Sign}(T, \mathsf{sk}_{sig})$
- **return** (T, σ)

TK-upd-sig
- $\mathsf{ctr}++$
- $(\tau'[\mathsf{ctr}], U) \leftarrow \textsc{UpGen}(\tau, \mathsf{ME})$
- $T \leftarrow (\mathsf{up}, \mathsf{ME}, U)$
- $\mathsf{conf}[\mathsf{ctr}] \leftarrow T$
- $\sigma \leftarrow \mathsf{Sign}(T, \mathsf{sk}_{sig})$
- **return** (T, σ)

TK-proc-sig(T, W)
- if $\exists j : T = \mathsf{conf}[j]$
  - $\tau \leftarrow \tau'[j]$
- else
  - $\mathsf{proc}(T, W)$
- $\mathsf{ctr} \leftarrow 0$
- $\tau'[\cdot], \mathsf{conf}[\cdot] \leftarrow \perp$
- **return** (τ, I)

---

proc($W = (\mathsf{create}, G, \mathsf{pk})$)
- $1/0 \leftarrow \mathsf{Vrfy}_\Sigma(\sigma, \mathsf{vk}_{sig})$
- let j s.t. $G.\mathsf{ID}_j = \mathsf{ME}$
- $\mathsf{sk}_j \leftarrow \mathsf{GMget\text{-}sk}(\mathsf{pk}.j)$
- $\tau \leftarrow \textsc{Init}(G, \mathsf{pk}, j, \mathsf{sk}_j)$

proc($T = (\mathsf{add}, \mathsf{ID}, \mathsf{ID}', \mathsf{pk}')$)
- $1/0 \leftarrow \mathsf{Vrfy}_\Sigma(\sigma, \mathsf{vk}_{sig})$
- $\tau \leftarrow \textsc{AddID}(\tau, \mathsf{ID}', \mathsf{pk}')$
- $\tau \leftarrow \textsc{Blank}(\tau, \mathsf{ID}')$

proc($W = (\mathsf{wel}, \tau')$)
- $1/0 \leftarrow \mathsf{Vrfy}_\Sigma(\sigma, \mathsf{vk}_{sig})$
- $\tau \leftarrow \tau'$
- $\tau.\mathsf{ME.sk} \leftarrow \mathsf{GMget\text{-}sk}(\tau.\mathsf{ME.pk})$

proc($T = (\mathsf{rem}, \mathsf{ID}, \mathsf{ID}')$)
- $1/0 \leftarrow \mathsf{Vrfy}_\Sigma(\sigma, \mathsf{vk}_{sig})$
- $\tau \leftarrow \textsc{Blank}(\tau, \mathsf{ID}')$
- $\tau \leftarrow \textsc{RemID}(\tau, \mathsf{ID}')$

proc($T = (\mathsf{up}, \mathsf{ID}, U)$)
- $1/0 \leftarrow \mathsf{Vrfy}_\Sigma(\sigma, \mathsf{vk}_{sig})$
- $\tau \leftarrow \textsc{UpPro}(\tau, \mathsf{ID}, \mathsf{ME}, U)$

図 2 TreeKEM$^*_\Sigma$ プロトコル．GMget-pk(ID')：GM から ID' に対応する pk' を取得する．
GMget-sk($\mathsf{pk}.j$)：sk_j を GM に問い合わせて取得する．

ctrl-chall(t)
- if $G.\mathsf{ID}_0 = \mathsf{GM}[t]$
  - return σ
- else
  - return 0

図 3 オラクル ctrl-chall

dis($q_1, \cdots, q_q$)
- for i s.t. $q_i =$ **ctrl-chall**(t^*) for some t^*
- if $\mathsf{Vrfy}(\sigma_i) \neq 1$
  - return 0

図 4 安全性ゲーム dis

[10] Yevgeniy Dodis and Nelly Fazio. “Public key broadcast encryption for stateless receivers”. In Joan Feigenbaum, editor, Digital Rights Management, pages 61-80, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[11] Amos Fiat and Moni Naor. “Broadcast encryption”. In Douglas R. Stinson, editor, CRYPTO’93, volume 773 of LNCS, pages 480-491. Springer, Heidelberg, August 1994.
pages 33-62. Springer, Heidelberg, August 2018.

[12] Suvo Mittra. Iolus: A framework for scalable secure multicasting. In Proceedings of ACM SIGCOMM, pages 277-288, Cannes, France, September 14-18, 1997.

[13] Eric Rescorla. Subject: [MLS] TreeKEM: An alternative to ART. MLS Mailing List. Thu, 03 May 2018 14:27 UTC, 2018. `https://mailarchive.ietf.org/arch/msg/mls/WRdXVr8iUwibaQu0tH6sDnqU1no`.

[14] Matthew Weidner. “Group messaging for secure asynchronous collaboration. MPhil Dissertation”, 2019. Advisors: A. Beresford and M. Kleppmann, 2019. `https://mattweidner.com/acs-dissertation.pdf`.