ミドルボックスを経由する通信の安全性を保証するための要件の定義とプロトコルの設計に関する一考察

北 健太朗[1,a)]　武政 淳二[1,b)]　小泉 佑揮[1,c)]　長谷川 亨[1,d)]

概要：インターネット上の通信の多くは，通信の品質やホストの安全性の向上などの機能を提供するミドルボックスを経由しているが，ミドルボックスを経由する通信の安全性を保証する手法は十分には確立されていない．具体的には，通信の安全性を保証するプロトコルとして広く利用されている Transport Layer Security (TLS) は，2 台のホストのみが参加する通信を考慮しており，平文データの読み書きを必要とするミドルボックスが参加する通信の安全性は保証できない．さらに，既存研究ではミドルボックスを経由する通信の安全性を保証するためのプロトコルが設計されているが，(1) プロトコルの要件が整理されていない，(2) 通信に参加するノードがオネストな場合のみを考慮しているため，それらのうち一つが攻撃者にコンプロマイズされた場合に通信全体の安全性が損なわれる，という課題がある．本研究では，既存研究で提案された要件を整理した上で，それらの要件を満たしつつ，通信に参加するあるノードがコンプロマイズされることが，通信の安全性に最小限の影響のみを及ぼすようにプロトコルを設計する．

キーワード：セキュリティ，ミドルボックス，プロトコル

A Study on Requirements and a Protocol for Secure Communication through Middleboxes

Kentaro Kita[1,a)]　Junji Takemasa[1,b)]　Yuki Koizumi[1,c)]　Toru Hasegawa[1,d)]

Abstract: Internet communication today involves middleboxes, aiming at improving quality of communication and security of hosts. Transport Layer Security (TLS) protocols have been designed only for communication between exactly two hosts, and thus, they cannot provide security in the cases that middleboxes require permission to read and write plaintext data. Several studies have designed protocols to provide security for such communication, however, they have the following problems: Requirements for protocols have not been systematically organized. They assume only the case that all hosts and middleboxes are honest. Therefore, if one of the hosts and middleboxes is compromised by an adversary, security of the session can completely be compromised. In this paper, we organize the requirements proposed in existing studies and design a protocol in which an adversary can only cause minimal impacts on security of a session even if the adversary compromises any of the hosts and middleboxes in the session.

Keywords: security, middlebox, protocol

1. はじめに

現在，インターネット上の多くの通信は，通信の品質やホストの安全性の向上を目的とし，ミドルボックスと呼ばれる，パケット転送以外の付加的な機能を提供する中継機器を経由している．そのようなミドルボックスの例とし

[1] 大阪大学大学院情報科学研究科
Graduate School of Information Science and Technology, Osaka University, Japan
[a)] k-kita@ist.osaka-u.ac.jp
[b)] j-takemasa@ist.osaka-u.ac.jp
[c)] ykoizumi@ist.osaka-u.ac.jp
[d)] t-hasegawa@ist.osaka-u.ac.jp

て，サーバが送信したデータを圧縮するデータ圧縮プロキシや，通信を監視して外部ネットワークからの攻撃を検知する侵入検知システムなどがある．以降では，特にアプリケーション層データの読み込み，書き換えを行うミドルボックスを想定する．ミドルボックスは有益だが，経由する通信の安全性を保証する手法は十分に確立されていない．

Transport Layer Security (TLS) [1] は，通信の安全性の保証を目的とするプロトコルのうち，最も広く使用されているものの一つである．TLS では，公開鍵証明書などによるホストの認証と，平文データの暗号化と MAC タグの付与によるデータの秘匿性と完全性を達成する．しかし，TLS は 2 台のホストのみがセッションに参加する通信のみを対象に設計されているため，ミドルボックスがセッションに参加し，平文データの読み込みや書き換えを行う通信に TLS を適用したとしても，ミドルボックスが機能を提供できない，あるいは，通信全体の安全性を保証できない．例えば，2 台のホスト間の通信全体を TLS で保護するとミドルボックスが機能を提供できず，逆に，ホストと 1 ホップ目のミドルボックス間の通信のみを TLS で保護したとしても，それ以降の通信の安全性が保証されているか (データが暗号化されているか，どのようなミドルボックスを経由するかなど) をホストが知ることができない．

このような課題に対して，ミドルボックスを経由する通信の安全性を保証するプロトコルとして，mcTLS [2]，TLMSP [3]，mbTLS [4]，maTLS [5] が TLS を基に設計されている．これらのプロトコルでは，ホストは通信が経由するすべてのミドルボックスを認証した後，各ミドルボックスと共通鍵を交換する．この共通鍵を使用し，各ホップ (クライアント - ミドルボックス間，ミドルボックス - ミドルボックス間，ミドルボックス - サーバ間) では TLS と同様にデータの暗号化と MAC タグの付与を行う．一方，認証されたミドルボックスは，交換した共通鍵を用いてデータの復号と MAC タグの検証を行った後，データの読み込みや書き換えを行い，再び暗号化と MAC タグの付与を行ったデータを次ホップへ転送する．以上のようなデータ転送により，認証されたミドルボックスが機能を提供できるようにしながら，認証されたミドルボックス以外に対してはデータの秘匿性と完全性を達成する．

しかし，既存研究には 2 つの課題がある．まず，プロトコルが満たすべき要件が述べられているが，それらが独立に定義されているため，統一的に整理されていない．次に，あるセッションに参加するノード (ホストとミドルボックス) がすべてオネストである場合しか考慮していない．そのため，あるノードの秘密情報が攻撃者にコンプロマイズされた場合，通信全体の安全性が損なわれる．近年，TLS などの通信の安全性を保証するプロトコルでは，forward secrecy [6] や post-compromise security [7] の達成，key compromise impersonation 攻撃 [8] に対する防御の重要性が着目されるなど，あるノードの長期的な秘密鍵やあるセッションの共通鍵がコンプロマイズされた場合にも，一定の通信の安全性を保証するように設計する必要性があることが指摘されており，ミドルボックスを経由する通信の安全性を保証するプロトコルの設計においてもそのような場合の安全性を考慮すべきである．

本稿では，プロトコルの安全性を，セッションに参加する全ノードが攻撃者にコンプロマイズされていない場合に達成される安全性と，一つ以上のノードがコンプロマイズされている場合に達成される安全性に分類し，前者を保証するための要件を既存研究に基づき整理し，後者をあるノードがコンプロマイズされた場合に通信全体の安全性に最小限の影響しか与えないこととして新たに定義する．さらに，それらの要件に基づきミドルボックスを経由する通信の安全性を保証するプロトコルを設計する．

本稿の構成は以下の通りである．まず，2 章で関連研究について述べる．次に，3 章で本研究で想定する攻撃者モデルを説明する．4 章ではプロトコルの要件の定義と設計を行い，5 章で設計プロトコルの安全性を検証する．最後に，6 章で本稿をまとめる．

2. 関連研究

2.1 TLS

TLS は 2 台のホスト (クライアントとサーバ) 間の通信でホストの認証，データの秘匿性と完全性を達成する．TLS は主に，認証付き暗号化により保護されたレコード (一度に送信されるデータのフラグメント) を送受信するレコードプロトコルと，レコードプロトコルで使用する共通鍵や暗号スイートなどをホスト間で合意するためのハンドシェイクプロトコルからなる．以下では，TLS 1.2 でやりとりされるメッセージの典型的な例を説明する．

まず，クライアントは `ClientHello` をサーバへ送信することでハンドシェイクプロトコルを開始する．`ClientHello` はクライアントがそのセッションのみで使用するランダム値，クライアントが対応する暗号スイートのリストを含む．サーバはセッションで使用する暗号スイートを受信した暗号スイートのリストから選択した後，クライアントと同様にそのセッション用のランダム値を選択し，それらを `ServerHello` でクライアントに送信する．さらに，サーバは `ServerCertificate` で自身の公開鍵証明書を，`ServerKeyExchange` で自身が選択したディッフィー・ヘルマン公開値とそれに対する署名を送信することで，クライアントがサーバを認証することを可能にする．その後，サーバは `ServerHelloDone` を送信し，この時点で送信すべきメッセージを送信し終えたことを通知する．クライアントはサーバの署名を検証し，署名が正当ならば，自身が選択したディッフィー・ヘルマン公開値を `ClientKeyExchange` で送信する．この時点で，クライアントとサーバは受信し

たディッフィー・ヘルマン公開値と自身のディッフィー・ヘルマン秘密値から共通鍵を導出することが可能である．その後，クライアントは ChangeCipherSpec と Finished を送信する．ChangeCipherSpec はこれ以降の送信データを暗号化することを通知するメッセージであり，Finished はトランスクリプト (ホストがこれまでに送受信した全メッセージのハッシュ値) に対する MAC タグを含む．この MAC タグを検証することで，サーバはハンドシェイク中にやりとりしたメッセージが改ざんされていないかを検証できる．MAC タグが正当ならば，サーバもクライアントへ ChangeCipherSpec と Finished を送信する．この後，両ホストはレコードプロトコルに移行する．

2.2 ミドルボックスを経由する通信の安全性の保証

ミドルボックスを経由する通信の安全性を保証するための既存プロトコルを説明する．これらのプロトコルはすべて 1 章で説明したホップバイホップのデータ保護を行うが，それぞれ設計方針が以下のように異なる．

2.2.1 mcTLS，TLMSP

mcTLS では，ミドルボックスに対して，以下で説明するようなコンテキストアウェアなデータを保護を行うことを目的とする．一般に，平文データのどの部分に対して読み込みや書き込み処理を行う必要があるかはミドルボックスごとに異なる．例えば，データ圧縮プロキシの場合，HTTP レスポンスのボディ部分の読み込みと書き換え権限を必要とするが，一方，侵入検知システムは HTTP リクエストとレスポンス全体の読み込み権限のみを必要とする．したがって，データの各部分に対してミドルボックスに付与する読み込み，書き換え権限をホストが制御し，各ミドルボックスに最小限の権限のみを付与できるようにすることが望ましい．

コンテキストアウェアなデータ保護を実現するために，mcTLS ではデータを複数のコンテキストと呼ばれるフラグメント $\{c_1, \cdots, c_n\}$ に分割する．さらに，通信経路上のミドルボックスを，各コンテキスト c_i の読み込み権限のみ与えるミドルボックス (リーダー) と読み込みと書き換えの両方の権限を与えるミドルボックス (ライター) に分類し，ホストがリーダーとは共通鍵 k_r^i を，ライターとは共通鍵 k_r^i と k_w^i をそれぞれ交換する．そして，ホストは自身が送信するコンテキスト c_i を k_r^i で認証付き暗号化し，さらに，k_w^i で MAC タグの付与を行う．このとき，リーダーは k_r^i を持つためコンテキストを復号でき，コンテキストの読み込みが可能であるのに対して，それ以外のノードは復号することができない．一方，ライターは k_r^i と k_w^i を持つため，復号に加えて正当な MAC タグの生成が可能であるため，コンテキストの読み込みと書き換えの両方が可能であるのに対して，それ以外のノード (リーダーやホストに認証されていないノード) がコンテキストを書き換えた場合，正当な MAC タグを生成できず，ライターやホストに検知されずにコンテキストを書き換えることができない．

TLMSP [3] は，mcTLS を改良したプロトコルであり，ETSI が標準化した．主な改良点として，mcTLS では各レコードに一つのコンテキストしか含まれないようにする必要があったのに対して，トラフィックの最適化を行うことを可能とするために，TLSMSP は複数のコンテキストを一つのレコードで送信できるように設計され，さらに，データのオリジンのノードを示すフラグが各コンテキストとともに暗号化して送信されるように設計されている．本稿では，mcTLS や TLMSP と同様にコンテキストアウェアなデータ保護を行うことを目的とする．

2.2.2 mbTLS

mbTLS は，mcTLS と同様に各ミドルボックスにデータへの最小限のアクセス権限のみを与えることを目的とするが，mcTLS とはアプローチが異なる．具体的には，mbTLS では，各ミドルボックスのソフトウェアを，Intel Software Guard Extensions (Intel SGX) が提供する，暗号化と MAC タグの付与により保護された安全なメモリ領域上で動作させることで，ミドルボックスが動作するプラットフォーム上のシステムソフトウェアなどが信頼できない場合にも，ホストが認証した信頼できるミドルボックスのソフトウェアのみが平文データの読み込みや書き換え可能とする．ただし，欠点としては，Intel SGX に対応する特別なハードウェアが具備されたプラットフォームが必要であることが挙げられる．

2.2.3 maTLS

mcTLS と mbTLS がレコードプロトコルにおいて各ミドルボックスにデータへの最小限のアクセス権限のみを与えることを目的としていたのに対して，maTLS は主に，ハンドシェイクプロトコルの安全性の向上を目的とする．具体的には，mcTLS と mbTLS のハンドシェイクプロトコルではホストのみがセッションで使用される暗号スイートを決定するため，各ミドルボックスがその暗号スイートに対応していなければセッションに参加できず，逆に，各ミドルボックスがより安全性の高い暗号スイートに対応していたとしても，ホストが決定した，より安全性の低い暗号スイートを使用しなければならない．この課題を解決するために，maTLS ではセッション中の隣接するノード間で独立に暗号スイートを決定する．

3. システムモデルと攻撃者モデル

3.1 システムモデル

ネットワーク管理者やコンテンツプロバイダはネットワーク上にミドルボックスを配置しており，それらは個別の IP アドレスと公開鍵証明書を割り当てられていると仮定する．各ミドルボックスの公開鍵証明書はそのミドルボックスのソフトウェアの開発元の識別子とミドルボック

スが必要とするデータへのアクセス権限を含む．ホストの通信が経由するミドルボックスはルーティング時に決定されており，経路上の隣接したノードは TCP コネクションを確立していると仮定する．また，既存研究と同様に，アプリケーション層データの読み込みや書き換えを行うミドルボックスのみを想定し，NAT などの下位層で動作するミドルボックスやキャッシュプロキシなどのデータキャッシングを行うミドルボックスは想定しない．

3.2 攻撃者モデル

攻撃者の目的は，ホストになりすまして他のホストと通信する，あるいは，ホストがやりとりするデータの盗聴，改ざん，並び替え，削除などを行うことである．本稿では，攻撃者はサイドチャネル攻撃やノード上で実行中のプログラムの脆弱性を突くことにより，(1) あるセッションで使用される共通鍵，(2) あるノードの長期的な秘密情報 (公開鍵証明書に含まれる公開鍵に対応する秘密鍵など) を取得することができると仮定する．(1) の場合，あるセッションでやりとりされるデータをネットワーク上で盗聴，改ざんすることが可能となり，(2) の場合，それに加えて，攻撃者がそのノードになりすますことが可能となる．あるセッションに参加するノードの秘密情報が漏洩したとき，そのノードとセッションはコンプロマイズされたという．

まず，セッションに参加するノードがすべてコンプロマイズされていない場合に，TLMSP のようなコンテキストアウェアなデータ保護に対して可能な攻撃を整理する．これらの攻撃は，各コンテキストとそれが含まれるレコード，そのレコード内での順序が紐付いていないため引き起こされる．以降では，あるホストがもう一方のホストに，それぞれ n 個のコンテキストからなる 2 つのレコード $R_1 = \{c_{1,1}, \cdots, c_{1,n}\}$ と $R_2 = \{c_{2,1}, \cdots, c_{2,n}\}$ を連続して送信する場合を想定する．

コンテキストの削除:攻撃者がホストの送信した R_1 から $c_{1,i}$ を取り除き，R_1 の代わりに $c_{1,1}, \cdots, c_{1,i-1}, c_{1,i+1}, \cdots, c_{1,n}$ をミドルボックスまたはホストへ送信する．

コンテキストの再送:攻撃者がホストの送信した R_1 に含まれる $c_{1,i}$ を保存しておき，次に送信される R_2 に含まれる $c_{2,i}$ を $c_{1,i}$ に置き換え，$\{c_{2,1}, \cdots, c_{2,i-1}, c_{1,i}, c_{2,i+1}, \cdots, c_{2,n}\}$ を R_2 の代わりにミドルボックスまたはホストへ送信する．

コンテキストの並び替え:攻撃者が R_1 に含まれる 2 つのコンテキスト $c_{1,i}$ と $c_{1,i+1}$ の順序を入れ替え，$\{c_{1,1}, \cdots, c_{1,i+1}, c_{1,i}, \cdots, c_{1,n}\}$ を R_1 の代わりにミドルボックスまたはホストへ送信する．

次に，セッションに参加するノードの一つが攻撃者にコンプロマイズされている場合に，上記の攻撃に加えて可能となる攻撃の例を示す．

不正なコンテキストの送信:攻撃者がコンプロマイズした書き換え権限を持つミドルボックスになりすまし，クライアントに悪意のあるデータを送信する．TLSMSP では各コンテキスト内のフラグがデータのオリジンを示す情報として含まれているが，ライターをコンプロマイズした攻撃者はこのフラグを書き換えることが可能であり，フラグが書き換えられた場合，クライアントは受信したデータはサーバが送信したものであると判定してしまう．

4. プロトコル

4.1 要件

本稿では，コンテキストアウェアなデータ保護を行う．設計プロトコルの要件を以下のように定義する．ただし，括弧内は要件を達成できる既存プロトコルを示す．

まず，コンプロマイズされていないセッションに対して以下を達成する．

ホストとミドルボックスの認証 (mcTLS, TLMSP, mbTLS, maTLS): 通信するホストとセッションに参加するミドルボックスがお互いを認証する．これにより，データがどのミドルボックスを経由するかをホストが確認できるようにし，また，各ミドルボックスが通信しているホストに応じた処理を実行することを可能にする (例：ファイアウォールが危険だと判断する web サイトの送信データにはより厳しい制限を設ける場合)．ただし，TLS と同様に，負荷を軽減するためにサーバやミドルボックスがクライアントを認証しない場合がある．

データの秘匿性 (mcTLS，TLMSP，mbTLS，maTLS): 認証されたホストとミドルボックス以外のノードが平文データを読み込むことができない．これにより，平文データが攻撃者に漏洩することを防ぐ．

データの完全性 (mcTLS，TLMSP，mbTLS，maTLS): 認証されたホストとミドルボックス以外のノードが平文データを書き換えた場合，ホストまたはミドルボックスが検知できる．これにより，平文データが攻撃者に改ざんされることを防ぐ．

ホップごとの暗号スイートの選択 (maTLS): ホストのみが暗号スイートを決定するのではなく，セッションに参加する，隣接する各ノード間で暗号スイートを決定することで，ホップごとに最も強度の高い暗号スイートを選択する．

ホップごとのハンドシェイクの完全性 (maTLS): 各ホップでのハンドシェイク中のメッセージの改ざんをホストまたはミドルボックスが検知できる．これにより，攻撃者がハンドシェイク中のメッセージを改ざんし，ホストやミドルボックスに弱い暗号スイートを使用させるなどの攻撃を防ぐ．

データのオリジンの検証 (TLMSP，maTLS): 各データの送信元ホストを受信ホストが検証可能とする．

最小限のアクセス権限の付与 (mcTLS，TLMSP，mbTLS): ホストは各ミドルボックスにその機能を提供するために必要な最低限の読み込み，書き換え権限しか与えない．

表 1 表記の説明
Table 1 Summary of notation.

Notation	Description
$[n]$	$\{1, \cdots, n\}$
c_i	コンテキスト
c,m,s	クライアント，ミドルボックス，サーバを表すノード識別子
pk_i, sk_i	ノード i の長期的な公開鍵ペア ($i \in \{c, m, s\}$)
$Cert_{pk_i}$	pk_i の公開鍵証明書 ($i \in \{c, m, s\}$)
r_{i-j}	ノード i，j 間のセッションで使用するランダム値 ($i, j \in \{c, m, s\}$)
$Suite_i$	ノード i が対応する暗号スイートのリスト ($i \in \{c, m, s\}$)
$suite_{i-j}$	ノード i，j 間で使用する暗号スイート ($i, j \in \{c, m, s\}$)
DH^+_{i-j},DH^-_{i-j}	ノード i，j 間で使用するディッフィー・ヘルマンペア ($i, j \in \{c, m, s\}$)
k_h	クライアント - サーバ間で交換する共通鍵
$k^i_{r_j}$	i 番目のコンテキストに対応する，j 番目のホップのリーダー用の共通鍵 ($i, j \in \mathbb{N}$)
$k^i_{w_j}$	i 番目のコンテキストに対応する，j 番目のホップのライター用の共通鍵 ($i, j \in \mathbb{N}$)
k_{i-j}	ノード i，j 間で交換する共通鍵 ($i, j \in \{c, m, s\}$)
$k^i_{r_j,c}$,$k^i_{r_j,s}$	i 番目のコンテキストに対応する，ホストが j 番目のホップのリーダーへ送信する秘密情報 ($i, j \in \mathbb{N}$)
$KeyGen()$	共通鍵生成アルゴリズム
$Sign_{sk_i}()$	sk_i を用いて生成した署名
$AuthEnc_k()$	共通鍵 k を用いた認証付き暗号化により生成された暗号文と MAC タグ
$Mac_k()$	共通鍵 k を用いて生成された MAC タグ
$\|\|$	バイト列の連結

経路の完全性 (TLMSP，mbTLS，maTLS): データが経由するミドルボックスの順序が固定される．これにより，通信が認証したすべてのミドルボックスを経由し，かつ，ホストが意図した順序でミドルボックスの機能が提供されることを保証する．経路の完全性はデータのフィルタリングや匿名化を行うミドルボックスを利用する場合に，ホストの安全性に影響を及ぼす．

さらに，セッションがコンプロマイズされた場合に以下を達成する．

最小限の安全上の被害 (なし): 通信に参加するノードの秘密情報が漏洩した場合に，そのノードに与えられた読み込み，書き換え権限以上の被害を及ぼさない．例えば，mcTLS や TLMSP ではあるミドルボックスの秘密情報が漏洩した場合，あるコンテキストに当たるデータを削除するなどしてもそのことを検知できず，mbTLS や maTLS ではあるノードがコンプロマイズされた場合に攻撃者はすべてのデータに対して任意の読み込みや書き換えが可能となる．

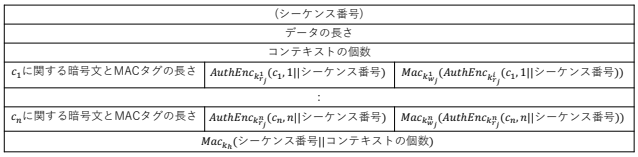

図 1 レコードの構造
Fig. 1 Structure of a record.

4.2 レコードプロトコル

プロトコルの説明に使用する記号を **表 1** にまとめる．まず，レコードプロトコルにおいて送信されるレコードの構造を **図 1** に示す．セッションに参加するホストとミドルボックスはハンドシェイク終了時にそのセッションでの受信，送信用のシーケンス番号をそれぞれ 0 で初期化し，レコードを受信，送信するたびに 1 だけインクリメントする．ただし，シーケンス番号は実際に送信されることはなく，各ノードがセッションの終了時まで保持する．シーケンス番号により，リプレイ攻撃などを検知することが可能となる．データの長さはレコード内のそれ以降の要素の長さを足し合わせたものである．さらに，各レコードはレコード内のコンテキストの個数を含み，コンテキストの個数は，シーケンス番号とともに，通信する 2 台のホストのみがハンドシェイク時に交換した共通鍵 k_h を用いて生成された MAC タグ Mac_{k_h}(シーケンス番号 || コンテキストの個数) により，ホスト以外による書き換えから保護される．k_h はホストのみが生成できるため，この MAC タグにより各ホストは正当なホストが送信したレコードを受信したことを検証できる．データの長さや各コンテキストがミドルボックスにより書き換えられうる要素であるのに対して，コンテキストの個数 (とシーケンス番号) はミドルボックスにより書き換えられない要素である．

次に，リーダーとライターが行うコンテキストの暗号化と MAC タグの付与に関連するレコードプロトコルの処理の概要を **図 2** に示す．ただし， 4.3 章で説明するハンドシェイクプロトコルにより，i 番目のコンテキスト c_i のリーダーのうち経路上の j 番目のものには共通鍵 $k^i_{r_{j-1}}$ と $k^i_{r_j}$ が，ライターのうち経路上の j 番目のものには共通鍵 $k^i_{w_{j-1}}$ と k_{w_j} がそれぞれ割り当てられていると仮定する．レコードプロトコルでは，権限を与えられたミドルボックスのみが読み込み，書き換え処理を行えること，ハンドシェイク時に各ミドルボックスに割り当てられた共通鍵 $\{k^i_{r_0}, \cdots, k^i_{r_{m_r}}\}$，$\{k^i_{w_0}, \cdots, k^i_{w_{m_w}}\}$ ($i \in [n]$，m_r:リーダーの台数，m_w:ライターの台数) に従う順序でのみデータが転送されることが保証される．

まず，j 番目のリーダーは，$j-1$ 番目のリーダーが送信した，認証付き暗号化された i 番目のコンテキスト $AuthEnc_{k^i_{r_{j-1}}}(c_i, i||$ シーケンス番号) の復号を行う．認証付き暗号化は平文データの暗号化と MAC タグの生成を統一的に行う暗号利用モードであり，暗号化時には入力され

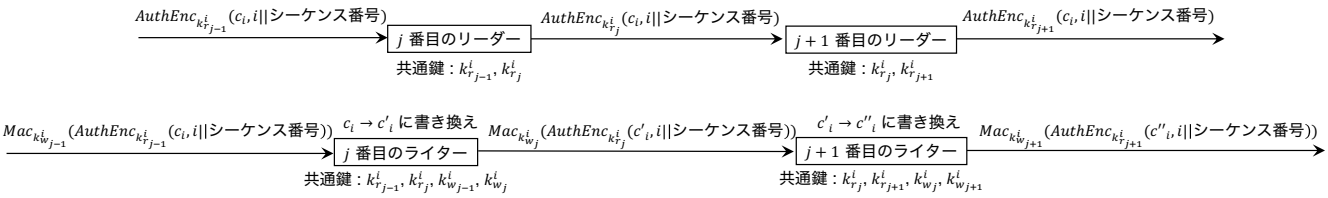

図 2 各ミドルボックスでの暗号化と MAC タグの付与
Fig. 2 Encryption/decryption and MAC tag generation/verification at middleboxes.

た平文データと共通鍵に対して，暗号文データ，データに対する MAC タグを生成し，復号時には入力された暗号文データと MAC タグに対して，データが改ざんされていない場合には平文データを，改ざんされている場合にはエラーを出力する．さらに，認証付き暗号化はオプションとして関連データを入力としてとることができる．関連データは暗号化はされないが，暗号文のデータと紐付けられ，改ざんから保護される．上記の例では，$(i||$ シーケンス番号$)$ が関連データに当たり，これにより，各コンテキストとそのコンテキストのレコード内での順序，レコードのシーケンス番号が紐付けられるため，攻撃者は 3.2 章で説明したような，1 つのレコード内のコンテキストの並び替えや，あるレコード内のコンテキストを異なるレコードで再送するなどの，レコードの構造を書き換える攻撃を行うことができなくなる．認証付き暗号化されたコンテキストの復号が成功した場合，ミドルボックスは平文コンテキストを読み込んだ後，ミドルボックスとしての処理を実行する．その後，自身が持つもう一方の共通鍵 $k^i_{r_j}$ で c_i の認証付き暗号化を行い，生成された $AuthEnc_{k^i_{r_j}}(c_i, i||$ シーケンス番号$)$ を次ホップへ送信する．

j 番目のライターは，上記のようなリーダーとしての処理に加えて，$j-1$ 番目のライターが共通鍵 $k^i_{w_{j-1}}$ を用いて生成された MAC タグ $Mac_{k^i_{w_{j-1}}}(AuthEnc_{k^i_{r_{j-1}}}(c_i, i||$ シーケンス番号$))$ の検証を行う．これにより，書き換え権限を持つミドルボックス以外がコンテキストを改ざんしていないかを検証できる．また，レコードを転送する際には書き換え後のコンテキスト c'_i に対して，共通鍵 $k^i_{w_j}$ を用いて MAC タグ $Mac_{k^i_{w_j}}(AuthEnc_{k^i_{r_j}}(c'_i, i||$ シーケンス番号$))$ を付与する．

4.3 ハンドシェイクプロトコル

ハンドシェイクプロトコルでやりとりされるメッセージの例を **図 3** に示す． 図 3 では一つのリーダーを経由する場合を説明するが，複数のリーダーやライターを経由する場合にも容易に拡張可能である． 図 3 は，サーバがクライアントを認証せず，クライアントは公開鍵証明書でサーバを認証し，共通鍵交換にディッフィー・ヘルマン共通鍵交換手法を使用する場合の例であり，赤矢印，青矢印，黒矢印はそれぞれ，クライアント - ミドルボックス間，ミドルボックス - サーバ間，クライアント - サーバ間でやりとりするメッセージを表す．

まず，各ノードはハンドシェイク用のフレッシュなランダム値 r_{i-j} を生成する．ただし，i は送信側ノード，j は受信側ノードを示す．クライアントは `ClientHello` をミドルボックスとサーバへ送信する．`ClientHello` は選択した r_{c-m}，r_{c-s} に加え，自身が対応する暗号スイートのリスト $Suite_c$ を含む．これらを受け取ったミドルボックスは，サーバ宛の `ClientHello` をサーバへ転送すると同時に，`MiddleboxHello` をサーバへ送信する．この `MiddleboxHello` は `ClientHello` と同様にミドルボックスが選択した r_{m-s} とミドルボックスが対応する暗号スイートのリスト $Suite_m$ を含む．これらのメッセージを受信したサーバは，`ServerHello`，`ServerCertificate`，`ServerKeyExchange`，`ServerHelloDone` をミドルボックスとサーバへそれぞれ送信する．`ServerHello` はクライアント - サーバ間，ミドルボックス - サーバ間で使用する暗号スイートを，`ServerCertificate` はサーバの公開鍵証明書を，`ServerKeyExchange` はサーバが選択したディッフィー・ヘルマン公開値とそれに対する署名をそれぞれ含む．

さらに，これらのメッセージを受信したミドルボックスは，クライアント宛のメッセージを転送しつつ，同様の役割を持つ 4 つのメッセージ `MiddleboxHello`，`MiddleboxCertificate`，`MiddleboxKeyExchange`，`MiddleboxHelloDone` をクライアントへ送信し，また，`MiddleboxKeyExchange` をサーバへ送信する．サーバ宛の `MiddleboxKeyExchange` はミドルボックスの選択したディッフィー・ヘルマン公開値を含むため，ミドルボックスとサーバはそれぞれ，共通鍵 k_{m-s} を $KeyGen(r_{m-s}, r_{s-m}, DH^+_{s-m}, DH^-_{m-s})$，$KeyGen(r_{m-s}, r_{s-m}, DH^+_{m-s}, DH^-_{s-m})$ により導出できる．$KeyGen()$ は，2 つのノードのディッフィー・ヘルマン公開値，秘密値の組から 2 台のノードが共有する秘密値を生成し，その秘密値と 2 台のノードが選択したランダム値からセッション用の共通鍵を生成する．共通鍵 k_{m-s} はミドルボックスにデータの読み込みを行うために必要な共通鍵 $k^i_{r_j}$ を割り当てる際に使用される．

サーバからの 4 つのメッセージを受信したクライアン

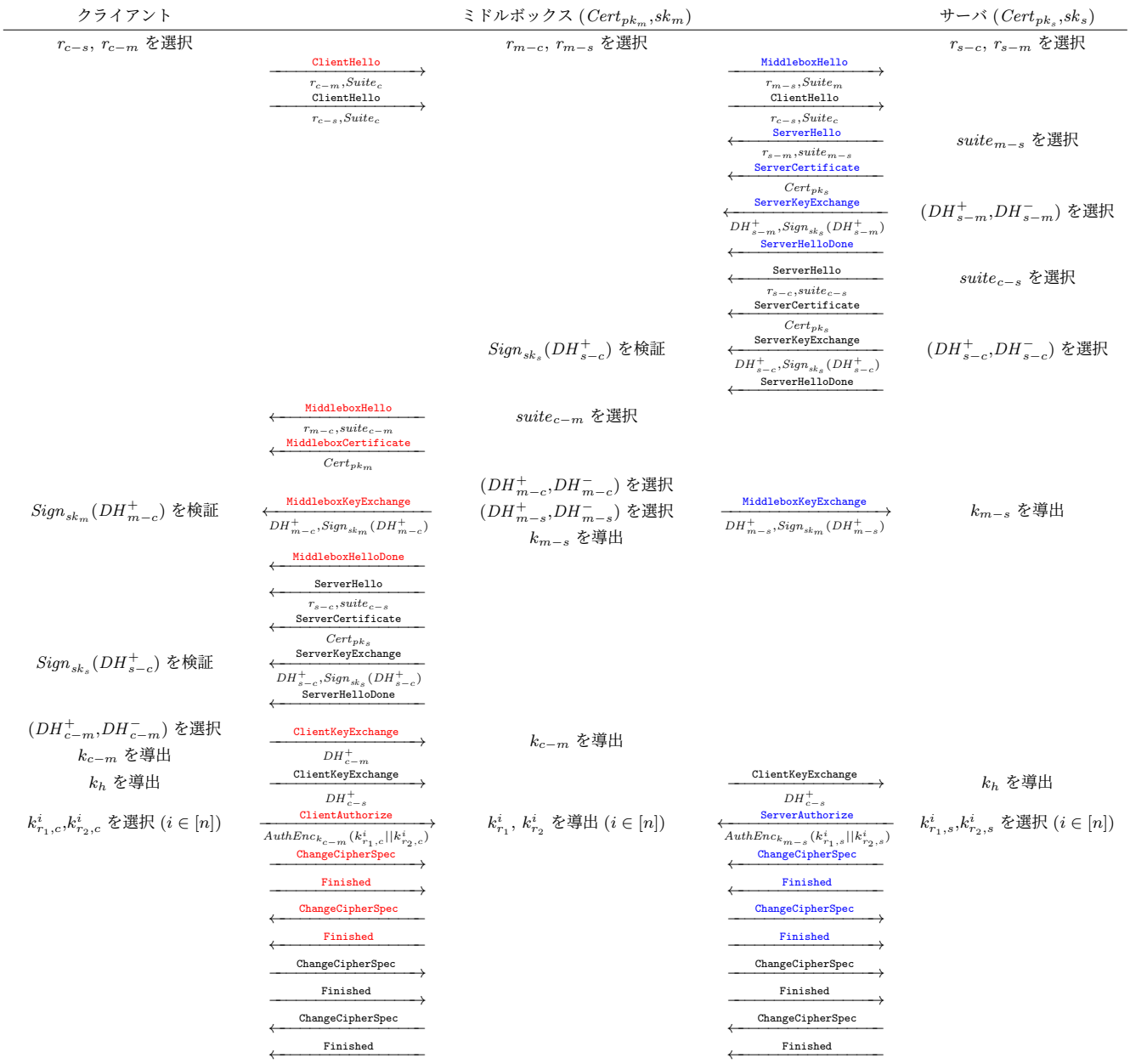

図 3 ハンドシェイクプロトコル
Fig. 3 Handshake protocol.

トは，自身の選択したディッフィー・ヘルマン公開値を含む ClientKeyExchange をミドルボックスとサーバへ送信する．ミドルボックスとサーバが選択したディッフィー・ヘルマン秘密値をクライアントの送信したディッフィー・ヘルマン公開値と組み合わせることで，ミドルボックスとサーバは共通鍵 k_{c-m} と k_h を $KeyGen()$ によりそれぞれ導出できる． 4.2 章で述べたように，共通鍵 k_h はデータのオリジンの検証とコンテキストの構造のエンドツーエンドでの保護を行うために使用され，共通鍵 k_{c-m} は，k_{m-s} と同様に，ミドルボックスへデータの読み込み権限を与えるために使用される．具体的には，各コンテキスト c_i に対して，ホストは自身の選択した秘密値 $k^i_{r_1,c}$，$k^i_{r_2,c}$，$k^i_{r_1,s}$，$k^i_{r_2,s}$ を，共通鍵 k_{c-m} と k_{m-s} を用いて認証付き暗号化したものを，それぞれ ClientAuthorize，ServerAuthorize によりミドルボックスへ送信する．これらの秘密値をもとに，ミドルボックスはクライアント - ミドルボックス間，ミドルボックス - サーバ間で使用する，c_i の読み込みを行うための共通鍵 $k^i_{r_1}$，$k^i_{r_2}$ を生成する．最後に，クライアント - ミドルボックス間，ミドルボックス - サーバ間，クライアント - サーバ間で ChangeCipherSpec，Finished を送信しあい，セッションの確立を終了する．TLS 1.2 と同様に，ChangeCipherSpec はこれ以降のメッセージは暗号化されることを通知するメッセージであり，Finished はハンドシェイクプロトコルで各ノードがこれまでに送受信しあった全メッセージのハッシュ値が一致するかを確認しあうためのメッセージである．それらのハッシュ値が一致

する場合，ハンドシェイクプロトコルは正常終了する．

5. 安全性の検証

ホストとミドルボックスの認証: ホストとミドルボックスは自身の公開鍵証明書とそこに含まれる公開鍵に対応する秘密鍵を用いて生成した署名を送信し合うことで互いを認証する．

データの秘匿性: ハンドシェイクプロトコルで認証されたホストミドルボックスのみとディッフィー・ヘルマン共通鍵交換手法に基づく共通鍵交換を行い，その共通鍵を用いた認証付き暗号化を行うため，認証されておらず，共通鍵を交換していないノードに対してデータの秘匿性が達成される．

データの完全性: データの秘匿性の同様に，認証されていないノードに対してデータの完全性が達成される．

ホップごとの暗号スイートの選択: 各ホップ (図 3 の例ではクライアント - ミドルボックス間，ミドルボックス - サーバ間，クライアント - サーバ間) で個別の暗号スイートのネゴシエーションを行うため，各ホップごとに最適な暗号スイートを選択できる．

ハンドシェイクの完全性: 各ホップのハンドシェイクが終了した際にそれぞれ `Finished` メッセージを送信しあうことで互いのトランスクリプトが一致するかを確認するため，ハンドシェイク中にやりとりしたメッセージが改ざんされていないことを検証できる．

データのオリジンの検証: 各レコードにはホストのみが持つ共通鍵 k_h を用いて生成された MAC タグが含まれるため，この MAC タグを検証することで，ホストは各レコードのオリジンがハンドシェイク中に認証したホストであることを検証できる．

最小限のアクセス権限の付与: 各ホストは，コンテキスト c_i の読み込み (や書き換え) を行うために使用する共通鍵 $k^i_{r_j}$(書き換えの場合は $k^i_{w_j}$) を導出するために必要な秘密情報 $k^i_{r_j,c}$，$k^i_{r_j,s}$(書き換えの場合 $k^i_{w_j,c}$，$k^i_{w_j,s}$) をそれぞれ暗号化してミドルボックスへ送信する．これにより，両ホストが秘密情報を送信したミドルボックスのみが読み込みや書き換え権限を与えられ，どのような権限をあるミドルボックスに与えるかをホストが管理することができる．

経路の完全性: コンテキストの読み込み，書き換え用の共通鍵 $k^i_{r_j}$，$k^i_{w_j}$ が各ホップごとに異なるため，ハンドシェイクプロトコルで各ミドルボックスに割り当てられた共通鍵に従う順序でのみデータを転送することが可能となる．

最小限の安全上の被害: まず，ミドルボックスがコンプロマイズされた場合，各ミドルボックスがコンテキストの読み込みや書き換えなどの自身に与えられた権限に基づく動作を行うことはできるが，あるコンテキストの削除やコンテキストの入れ替えなどそれ以外の攻撃を行うことができない．これは，それらの情報がホストのみが持つ共通鍵 k_h を用いて生成された MAC タグにより，エンドツーエンドで保護されているからである．次に，クライアント，サーバのいずれか一方のホストがコンプロマイズされた場合，ホストはミドルボックスになりすましてデータの書き換えなどを行うことはできない．これは各ミドルボックスに割り当てられる読み込み，書き換え用の共通鍵 $k^i_{r_j}$，$k^i_{w_j}$ を導出するために必要な秘密情報を両ホストが送信するからである．つまり，攻撃者がクライアントをコンプロマイズしたとしてもサーバ側の送信した秘密情報を知ることはできないため，攻撃者は $k^i_{r_j}$ や $k^i_{w_j}$ を導出することはできない．

6. おわりに

本稿では，ミドルボックスを経由する通信の安全性を保証するプロトコルの要件の整理とプロトコルの設計を行った．要求条件の整理では複数の既存研究を参考に必要な要件を選択し，プロトコルの設計ではそれらの要件を満たすプロトコルを TLS 1.2 を参考に設計した．

謝辞 本研究は科研費 19K22843，および，JSPS 特別研究員奨励費 21J12448 によるものである．

参考文献

[1] T. Dierke and E. Rescorla: The Transport Layer Security (TLS) Protocol Version 1.2, RFC 5246, 2008.

[2] D. Naylor, K. Schomp, M. Varvello, I. Leontiadis, J. Blackburn, D. López, K. Papagiannaki, P.R. Rodriguez, and P. Steenkiste: Multi-Context TLS (mcTLS): Enabling Secure In-Network Functionality in TLS. SIGCOMM Comput. Commun. Rev. 45, 4 (October 2015), 199–212.

[3] ETSI: Middlebox Security Protocol; Part2: Transport layer MSP, profile for fine grained access control. ETSI TS 103 523-2 v1.1.1, 2021.

[4] D. Naylor, R. Li, C. Gkantsidis, T. Karagiannis, and P. Steenkiste: And Then There Were More: Secure Communication for More Than Two Parties, in Proceedings of the 13th International Conference on emerging Networking EXperiments and Technologies (CoNEXT '17). Association for Computing Machinery, New York, NY, USA, 88–100.

[5] H. Lee, Z. Smith, J. Lim, G. Choi, S. Chun, T. Chung, and T. T. Kwon: maTLS: How to Make TLS middlebox-aware?. In NDSS '19.

[6] C. Cremers, and M. Feltz: Beyond eCK: Perfect Forward Secrecy under Actor Compromise and Ephemeral-key Reveal. Designs, Codes and Cryptography, 74(1), 183-218.

[7] K. Cohn-Gordon, C. Cremers, and L. Garratt: On Post-compromise Security. In 2016 IEEE 29th Computer Security Foundations Symposium (CSF) (pp. 164-178).

[8] C. Hlauschek, M. Gruber, F. Fankhauser, and C. Schanes: Prying Open Pandora's Box:KCI Attacks against TLS. In 9th USENIX Workshop on Offensive Technologies (WOOT 15).