



INFORMATION-TECHNOLOGY PROMOTION AGENCY, JAPAN

15 情経第 1403 号

## **セキュアプロトコルに対する攻撃法等に関する技術調査**

2004 年 3 月

独立行政法人 情報処理推進機構

## 本書の構成

本書は「セキュアプロトコルに対する攻撃法等に関する技術調査」における調査報告書である。

第1部のプロトコル調査編では、調査対象とするセキュアプロトコルの概要および既知の攻撃法についてまとめる。第2部のプロトコル検証編では、セキュアプロトコルに対するフォーマルメソッドを用いた検証結果を示す。第3部の総合分析編では、セキュアプロトコルに対する攻撃法の調査結果およびプロトコルの検証結果をもとに攻撃法の分類、フォーマルメソッドの適用可能範囲等について総合的な分析を行う。



# 目次

第 1 章	はじめに	1
1.1	背景	1
1.2	目的	1
1.3	仮説	1
第 I 部	プロトコル調査編	3
第 2 章	調査研究対象	5
2.1	調査範囲	5
2.2	調査方法	6
第 3 章	セキュアプロトコルの概要	7
3.1	TLS/SSL プロトコル	7
3.1.1	TLS の策定経緯	8
3.1.2	TLS プロトコル仕様	9
3.1.2.1	プロトコルの全体構成	9
3.1.2.2	ハンドシェイクプロトコル	10
3.1.2.3	暗号仕様変更プロトコル	12
3.1.2.4	警告プロトコル	13
3.1.2.5	アプリケーションデータプロトコル	14
3.1.2.6	レコードプロトコル	14
3.1.3	TLS で利用される暗号技術	14
3.1.4	TLS と SSL の相違点	17
3.2	IPsec プロトコル	18
3.2.1	IPsec の策定経緯	18
3.2.2	プロトコル仕様	19
3.2.2.1	プロトコルの全体構成	19
3.2.2.2	IKE プロトコル	22
3.2.2.3	ESP プロトコル	24
3.2.2.4	AH プロトコル	27

3.2.2.5	IPcomp プロトコル . . . . .	28
3.3	SSH プロトコル . . . . .	29
3.3.1	SSH の策定経緯 . . . . .	29
3.3.2	SSH プロトコル仕様 . . . . .	29
3.3.2.1	プロトコルの全体構成 . . . . .	30
3.3.2.2	SSH トランスポート層プロトコル . . . . .	30
3.3.2.3	SSH ユーザー認証プロトコル . . . . .	35
3.3.2.4	SSH コネクションプロトコル . . . . .	37
3.3.3	SSH1 の SSH2 に対する主な暗号利用の相違点 . . . . .	37
第 4 章	セキュアプロトコルに関する既知の攻撃法	39
4.1	攻撃法の調査範囲 . . . . .	39
4.2	TLS/SSL . . . . .	40
4.3	IPsec . . . . .	45
4.4	SSH . . . . .	49
第 II 部	プロトコル検証編	53
第 5 章	フォーマルメソッドによるプロトコル検証の動向	55
第 6 章	前提となる脅威モデルと検証の限界	59
第 7 章	検証対象	61
7.1	対象プロトコル . . . . .	61
7.2	検証すべき性質 . . . . .	61
第 8 章	代数仕様言語 CafeOBJ によるプロトコル検証	65
8.1	代数仕様言語 CafeOBJ の概要 . . . . .	65
8.2	検証法の概要 . . . . .	66
8.2.1	検証フロー . . . . .	66
8.2.2	観測遷移機械 (観測遷移システム) . . . . .	68
8.2.2.1	モデル . . . . .	68
8.2.2.2	検証法 . . . . .	69
8.3	プロトコル検証の構成と検証におけるリスク要因 . . . . .	69
8.4	TLS ハンドシェイクプロトコルのモデル化と検証 . . . . .	70
8.4.1	プロトコルの抽象化 . . . . .	70
8.4.2	モデルの構成 . . . . .	71
8.4.3	メッセージのモデル化 . . . . .	72
8.4.4	ネットワークのモデル化 . . . . .	74

8.4.5	プロトコルのモデル化 . . . . .	74
8.4.6	攻撃者のモデル化 . . . . .	77
8.4.7	検証ゴール . . . . .	78
8.4.8	証明譜 . . . . .	80
8.4.8.1	述語の定義 . . . . .	81
8.4.8.2	構造帰納法 . . . . .	83
8.4.9	証明実行結果 . . . . .	85
8.5	検証結果の分析 . . . . .	88
8.5.1	本モデルで保証されない安全性 . . . . .	88
8.5.2	モデルの拡張 . . . . .	89
8.5.3	メッセージ送受信観測のモデル化 . . . . .	89
8.5.4	課題 . . . . .	90
<b>第 III 部</b>	<b>総合分析編</b>	<b>91</b>
<b>第 9 章</b>	<b>攻撃法の分類とフォーマルメソッドによる検証可能領域</b>	<b>93</b>
9.1	攻撃法の分類 . . . . .	93
9.2	フォーマルメソッドによる検証可能領域 . . . . .	94
9.3	プロトコル仕様に係わる一般的な留意点 . . . . .	95
9.3.1	攻撃の対象となるプロトコル脆弱性の傾向 . . . . .	95
9.3.2	プロトコル仕様の記述範囲の境界 . . . . .	96
9.3.3	プロトコルの前提と安全性の選択 . . . . .	97
9.3.4	フォーマルメソッドを用いたプロトコル検証の効果 . . . . .	97
<b>第 10 章</b>	<b>まとめ</b>	<b>99</b>
<b>第 IV 部</b>	<b>付録</b>	<b>101</b>
付録 A	セキュアプロトコルに関連する用語の整理	103
付録 B	関連する RFC	107
付録 C	攻撃法の情報源	109
C.1	プロトコル検証・セキュリティ工学分野における主な研究者 . . . . .	109
C.2	プロトコル検証に関連する国際会議・論文誌 . . . . .	109
C.2.1	暗号関連 . . . . .	109
C.2.2	セキュリティ工学関連 . . . . .	109
C.2.3	フォーマルメソッド関連 . . . . .	111
C.3	脆弱性情報データベース . . . . .	111



# 目次

2.1	本調査研究の方法	6
3.1	インターネット通信の階層モデルとプロトコル事例 (出所: [99])	8
3.2	TLS プロトコルの全体構成概略	10
3.3	TLS ハンドシェイクプロトコルによるセッション確立のためのメッセージ交換フロー	11
3.4	TLS ハンドシェイクプロトコルにおける鍵マテリアルの関係	13
3.5	IPsec プロトコルに関する策定経緯 (出所: [99])	19
3.6	IPsec 構成するサブプロトコルの関係とそれらを定義する RFC([99] をもとに作成)	20
3.7	ホスト間のトランスポートモード IPsec 通信 ([99] をもとに作成)	20
3.8	セキュリティゲートウェイを介したトンネルモード IPsec 通信 ([99] をもとに作成)	21
3.9	IPsec による IP データグラムのカプセル化 (IPv4)(出所: [99])	21
3.10	IPsec による IP データグラムのカプセル化 (IPv6)(出所: [99])	21
3.11	IPsec パケット出力処理 (出所: [99])	22
3.12	IPsec パケット入力処理 (出所: [99])	22
3.13	IKE フェーズ 1・事前共有秘密鍵認証方式を利用したメインモード (出所:[99])	24
3.14	ESP ヘッダ	25
3.15	AH ヘッダ	27
3.16	SSH サブプロトコル群	30
4.1	セキュアプロトコルの既知の攻撃法に関する調査手順	39
5.1	主な形式手法と検証対象の関係	55
5.2	検証システムを用いたプロトコル検証の分類と事例変遷	56
7.1	TLS の性質と検証性質の関係	62
8.1	CafeOBJ 仕様定義の一般形	65
8.2	代数仕様における帰納的定義に基づくプロトコル検証フロー	67
8.3	プロトコル検証の構成と検証におけるリスク要因	70
8.4	検証対象とする TLS ハンドシェイクプロトコル	71
8.5	観測遷移システムにおける状態遷移の様子	72

8.6	TLS 検証モデルのモジュール構成	73
8.7	メッセージデータ構成のための演算子の宣言	74
8.8	攻撃者が利用可能なデータを構成する演算子の宣言	75
8.9	プロトコルに対する観測を行う演算子	75
8.10	プロトコル状態遷移を表す状態遷移オペレータ	76
8.11	ServerKeyExchange メッセージに対応する状態遷移演算 <code>kexch</code> の定義	76
8.12	ClientKeyExchange メッセージ偽造に対応する状態遷移規則	77
8.13	検証性質に関する関係	81
8.14	証明した性質（定理）の依存関係	82
8.15	証明書の等しさを示す述語の定義	82
8.16	暗号化された ClientFinish メッセージが、過去にネットワークに送信されたか判定する述語の定義	82
8.17	Premaster secret の存在を判定する述語の定義	83
8.18	状態集合と状態観測に対する述語の関係	84
8.19	オプションなメッセージ交換をすべて含む TLS ハンドシェイクプロトコルのモデル	89
8.20	メッセージ送受信観測のモデル	90
9.1	プロトコルの脆弱性分類と攻撃法の対応関係	93
9.2	攻撃法の分類におけるフォーマルメソッドによる検証可能領域	94

# 表目次

2.1	調査対象プロトコル	5
3.1	TLS で利用される暗号スイート一覧	15
3.2	TLS で使用できる鍵交換アルゴリズム	15
3.3	暗号アルゴリズム	16
3.4	ハッシュアルゴリズム	16
3.5	SSH のバージョン情報の例	31
3.6	SSH パケットのバイナリフォーマット	32
3.7	SSH 暗号アルゴリズム一覧	33
3.8	SSH メッセージ認証アルゴリズム一覧	33
3.9	SSH 公開鍵アルゴリズム一覧	34
3.10	鍵交換パケット	35
3.11	ユーザー認証方式	36
4.1	攻撃法分析表の項目説明	41
4.2	TLS/SSL に対する既知の攻撃法 (1/3)	42
4.3	TLS/SSL に対する既知の攻撃法 (2/3)	43
4.4	TLS/SSL に対する既知の攻撃法 (3/3)	44
4.5	IPsec に対する既知の攻撃法 (1/3)	46
4.6	IPsec に対する既知の攻撃法 (2/3)	47
4.7	IPsec に対する既知の攻撃法 (3/3)	48
4.8	SSH に対する攻撃法 (1/2)	50
4.9	SSH に対する攻撃法 (2/2)	51
B.1	IPsec プロトコルに関連する主要 RFC	107
B.2	TLS プロトコルに関連する主要 RFC	108
B.3	SSH プロトコルに関連するインターネットドラフト	108
C.1	プロトコル検証・セキュリティ工学に関する主要な研究者	110

# 第1章

## はじめに

### 1.1 背景

近年、インターネットは、広く社会基盤として利用されているが、インターネット上での個人情報や商取引情報等の送受信に際して、通信経路のセキュリティを確保することが不可欠となっている。インターネット上でセキュリティを確保する標準的かつ主要なプロトコルとして、IPsec や SSL/TLS 等が用いられる。これらのセキュアプロトコルについて、仕様上の潜在的なリスクおよび問題点について検証することが強く求められている。

### 1.2 目的

インターネットにおいて個人情報や商取引情報等を送受信する際に、通信経路のセキュリティを確保するために用いられる標準的なプロトコルについて、プロトコルの設計における暗号技術の利用法に関する技術調査を行う。また、開発者が各セキュアプロトコルの潜在的リスクを理解するための情報として、プロトコル設計上の諸問題点および攻撃手法に関して網羅的な調査を行った。

本調査では、セキュアプロトコルに関する設計上の問題点や攻撃法に関する文献調査に加え、実際にフォーマルメソッド（形式手法とも呼ぶ）を用いて、通信者どうしの秘密が攻撃者に漏洩しないことなどの安全性に関する検証を行うことで、人手による分析では発見が困難なプロトコル仕様上の潜在的なリスクについて分析する。また、攻撃法を分類した結果に対して、本調査研究で示す検証手法により検証可能な対象領域を示し、また、フォーマルメソッドによる検証が、他の同様のセキュアプロトコルの検証に適用可能であることを示す。本調査で得られた成果は、暗号技術評価事業の成果と合わせてベンダー等開発者に向けた情報提供を行うと共に、暗号技術評価事業における参考資料とすることができる。

### 1.3 仮説

本調査研究は、あらかじめ以下のような仮説を立てその評価検証に向けて実施した。

1. フォーマルメソッドを用いたセキュアプロトコルの検証により、特定の安全性に関する論理的な保証を確立すると共に、プロトコル仕様上の潜在的なリスクを明らかにすることができる。
2. フォーマルメソッドによる検証は、TLSのみならず、IPsec, SSHなどの類似するプロトコルにおけるメッセージ交換等の仕様の論理的な側面について、通信情報の機密性、合意内容の真正性等の検証を行うために応用することが可能である。

これらの仮説を評価検証するために、第2.2章の図2.1に示す方法により調査研究を行う。

## 第I部

# プロトコル調査編



## 第2章

# 調査研究対象

本調査では、インターネット上のセキュアプロトコルのうち標準的かつ代表的なプロトコルについて、各プロトコルに関する技術的構成、プロトコル仕様上の問題点、攻撃法に関して調査する。また、セキュアプロトコルの潜在的なリスクの分析を行うために、フォーマルメソッドを用いたプロトコル仕様の安全性を検証および分析を行う。

調査対象は、インターネット上での個人情報や商取引情報の送受信に際して使われる以下のプロトコルとする。

ネットワーク層	プロトコル	用途
アプリケーション層	SSH	遠隔ログイン、ファイル転送
トランスポート層	TLS/SSL	ブラウザ・サーバ間の通信セキュリティ
IP層	IPsec(IKE)	IPパケットレベルでの認証、暗号化

表 2.1: 調査対象プロトコル

これらのプロトコルは、ネットワーク透過性を考慮して設計されており、たとえば、IPsec プロトコルは、IP 層より上位のプロトコルサービスに対して適用することが可能である。また、TLS/SSL プロトコルは、トランスポートより上位のプロトコル http, SMTP, POP3 等のプロトコルサービスの保護に適用することが可能である。TLS/SSL は、ブラウザと http サーバとの通信の安全性を確保するプロトコルとして、インターネット上でもっとも普及しているセキュアプロトコルのうちの1つであると言える。

### 2.1 調査範囲

フォーマルメソッドを用いた検証に関しては、第5章で示す通り、検証手法によって、対象とするプロトコルの向き不向きがある。本研究では、代数仕様にもとづくフォーマルメソッドを用いたプロトコルの検証対象として、TLS プロトコルを選択する。これ以外の上記のプロトコルについても、本研究で用いる検証手法により検証することは可能であると考えられる。また、代表的なプロトコルを選択し検証することで、他のプロトコルに対する検証の見通しを立てることができ意義があるといえる。これらのプロトコルに対して、代数モデルに基づく形式仕様記述言語を用いてプロ

トコルのモデル化および攻撃者のモデル化をおこない、そのもとで、通信者どうしの秘密情報が攻撃者に漏洩しないことや、プロトコル終了時に通信者間で合意した内容が一致しているかなどのセキュリティの要件が満たされるか、代数仕様言語の検証推論システムを用いて証明する。

## 2.2 調査方法

本調査研究は、図 2.1 (p. 6) のように進める。

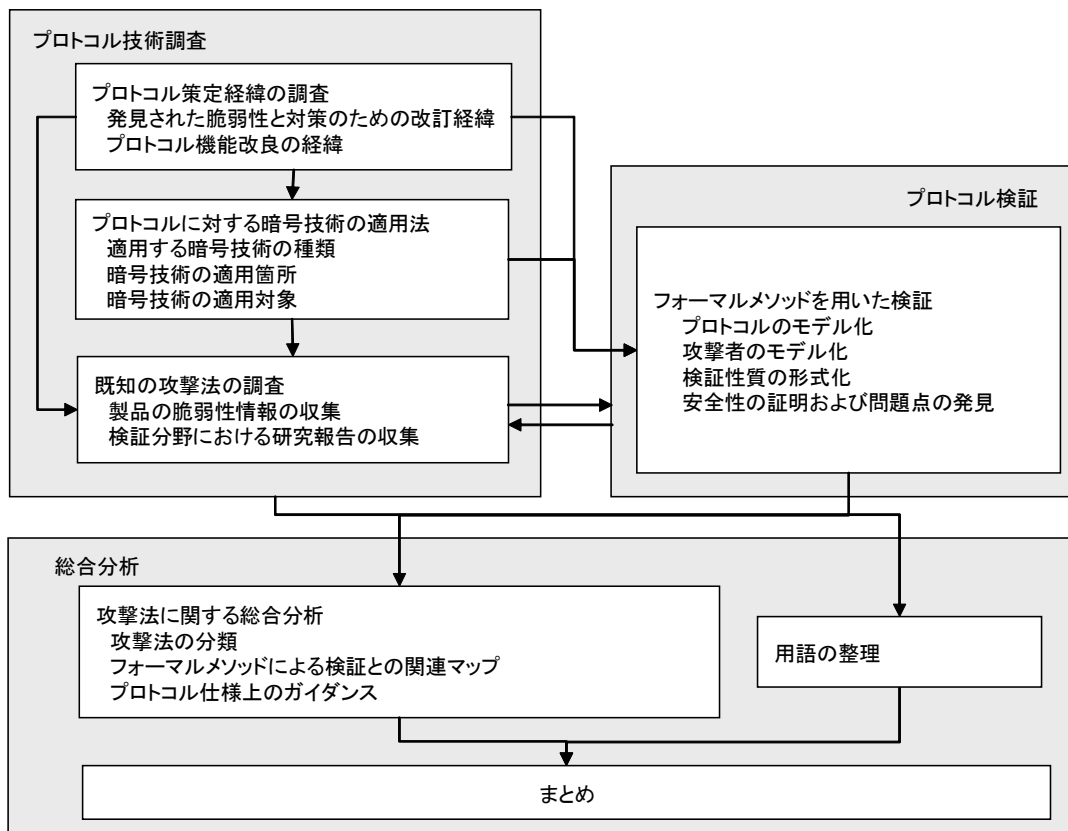


図 2.1: 本調査研究の方法

プロトコル技術調査においては、IETF の RFC および Internet-Draft 等をもとにプロトコルの策定経緯についてまとめる。また、プロトコルの概要をまとめると共に暗号技術の適法法をまとめる。これらの情報を参考とし、学术论文、脆弱性情報データベース等からプロトコルに対する既知の攻撃法を収集し分析する。

一方、プロトコル検証に関しては、フォーマルメソッドによるプロトコル検証の動向をまとめるとともに、代数仕様記述言語 CafeOBJ を用いてプロトコルのモデル化、検証性質の形式的記述および構造帰納法、等式推論を用いたプロトコルの検証を行う。

総合分析に関しては、プロトコル技術調査、プロトコル検証の成果をもとに、攻撃法の分類および、フォーマルメソッドによるプロトコル検証可能領域について分析する。

## 第3章

# セキュアプロトコルの概要

TCP/IP にもとづくインターネット上の通信は、歴史的な経緯から、セキュリティに関する保証機構は考慮されていなかった。近年、インターネットが通信の共通基盤として普及し、個人情報や商取引情報等の送受信がさかんになるにつれて、インターネット通信へのセキュリティ保証機構に対する要求が高まっている。

一般に、通信のセキュリティの要件としては、機密性 (Confidentiality)、完全性 (Integrity)、認証 (Authentication)、否認防止 (Non-repudiation)、可用性 (Availability) が必要とされる。インターネット通信のセキュリティを確保するために、これらの機能を提供するための各種セキュアプロトコルが提案されている。インターネットの通信機構は OSI 参照モデルに基づき構築されている (図 3.1(p. 8))。この図の各層でそれぞれ独立したいくつかのセキュアプロトコルが規定されていることが示されている。それらのセキュアプロトコルは、位置する通信層より上位層の通信内容に対して通信のセキュリティを保証する。図中にある、IPsec, TLS, SSH はそうしたセキュアプロトコルのなかでも特に代表的なもので、広く普及しているものである。

IPsec はネットワーク層プロトコルとトランスポートプロトコルの中間において通信のセキュリティを確保し、トランスポート層以上の情報のセキュリティを保証する。TLS はトランスポート層プロトコルとアプリケーション層プロトコルの間に位置し、アプリケーションによる通信のセキュリティを保証する。SSH は、遠隔地の端末の安全な操作のために利用されるが、このとき作られる安全な通信路を利用してアプリケーション毎の個別通信に対してセキュリティを保証する。本章では、こうした代表的なセキュアプロトコルの策定経緯と、それらの機構、提供する機能、暗号技術の適用法についてまとめる。

### 3.1 TLS/SSL プロトコル

TLS は、トランスポート層プロトコル TCP とアプリケーション層プロトコルの間で動作するセキュアプロトコルで、HTTP, SMTP, POP3, IMAP などの TCP を利用するアプリケーションに対して安全な通信を提供する。TLS は、データ送信元の認証、データの暗号化、メッセージ認証の機能を提供することによりセキュリティを実現する。

TLS を使用するには、アプリケーションのサーバプログラムとクライアントプログラムの両方が

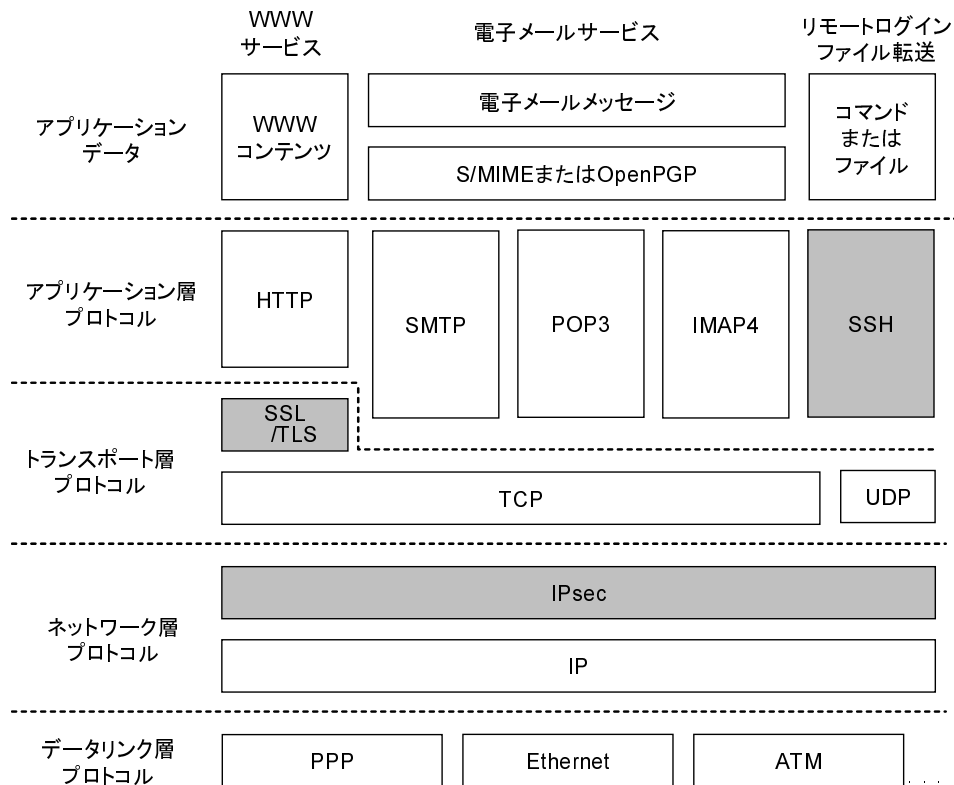


図 3.1: インターネット通信の階層モデルとプロトコル事例 (出所: [99])

TLS に対応する必要がある。TLS 通信を有効にするには、サーバプログラムが通常の TCP 通信とは異なるポート番号で待ち受けする方式と、サーバプログラムは通常の TCP 通信と同じポートでアプリケーションからの通信を受信して、その後、TLS を利用する通信へ遷移する方式がある。TLS の原型となる SSL の開発目的であった WEB サーバでの通信では、http アクセスに対する https アクセスとして、明示的に異なるポートを利用する通信として、前者の方式が一般的であったが、近年では後者の方式もいくつか提案されている [37], [70], [43]。

TLS によるセッション通信は、DES,RC4,RC2,3DES などの秘密鍵暗号方式を利用する。これらの暗号で利用される秘密鍵をクライアント・サーバで共有するために、鍵交換アルゴリズム (TLS ハンドシェイクプロトコル) が提案されている。また、メッセージの保護のために MD5 や SHA-1 といったハッシュ関数を使ってのメッセージダイジェストの計算も行う。鍵交換には、RSA や Diffie-Hellman などの公開鍵暗号が利用される。公開鍵や証明書の交換には、X.509 形式が利用される。

### 3.1.1 TLS の策定経緯

トランスポート層で通信セキュリティを保証するプロトコルとして SSL(Secure Socket Layer) が提案された。SSL は 1994 年に Netscape Communications 社 (米) によって開発されたプロトコルで、主として WWW アクセスの通信セキュリティを確保するために利用されてきた。これま

で、SSL 2.0, SSL 3.0 というバージョンが開発されてきた。SSL は安全な HTTP(Hyper Text Transfer Protocol) 通信を実現するために利用されてきたが、これをトランスポート層での安全な通信を実現するためのセキュアプロトコルとして標準化したものが TLS である。1996 年に IETF 内に TLS WG が用意され、SSL 3.0 をベースにした TLS 1.0 (Transport Layer Security version 1.0) を RFC として標準化した。TLS では、特に利用する暗号方式の選択が柔軟なものとなり、拡張性が増した。TLS WG では、その他にも TLS 関連の RFC をいくつか発行している。

TLS の一般的な実装事例としては、WEB ブラウザへの実装が挙げられる。サーバ側プログラムでは、Microsoft 社 (米) が開発する IIS(Internet Information Server) があり、UNIX 上で動作する代表的な WEB サーバへの実装も完了している。そのため WWW アクセスでは TLS(SSL) の利用は一般的に行われている。近年では、メールサーバソフトウェア sendmail も TLS 対応を果たし、それを受けていくつかのメールクライアントソフトウェアの TLS 対応も進んでいる。このように、HTTP 以外の通信での TLS 利用も進んでおり、今後はより多くのプロトコルに対応したアプリケーションへの TLS 対応が計画されている。

### 3.1.2 TLS プロトコル仕様

#### 3.1.2.1 プロトコルの全体構成

TLS は RFC 2246 で規定されているセキュアプロトコルである [25]。TLS は上位層と下位層の 2 つの通信層に分けて考えることができる。上位層は「警告プロトコル」、「アプリケーションデータ・プロトコル」、「暗合仕様変更プロトコル」、「ハンドシェイクプロトコル」からなり、これらは総合して TLS ハンドシェイクプロトコルと呼ばれる。下位層に位置するサブプロトコルが「レコードプロトコル」である。上位層の TLS ハンドシェイクプロトコルは通信相手の認証やセッション確立後、安全な通信を行うための暗号方式と秘密鍵の合意を行い、下位層は TLS ハンドシェイクプロトコルによって確立された暗号方式と秘密鍵を用いてセッション通信における通信内容の秘匿と認証を行う (図 3.2(p. 10))。

TLS プロトコルの目的は以下のようにまとめることができる：

- 暗号化セキュリティ  
2 つの通信者 (サーバ、クライアント) 間で安全なコネクションを確立する。
- 相互運用性  
各プログラマーが、お互いのコードを知らなくても、TLS を使用して暗号パラメータを交換できるアプリケーションを個別に開発できる。
- 拡張性  
TLS は、新しい方式の公開鍵暗号アルゴリズムや暗号アルゴリズムを必要に応じて組み込めるフレームワークを提供する。これによって、新たな脆弱性をもたらす可能性のある新規プロトコルを開発する必要がなくなり、セキュリティライブラリ全体を実装し直す手間も省ける。
- 相対的効率化  
暗号化処理、特に公開鍵暗号処理は計算負荷が高いという傾向がある。そのため、TLS プロト

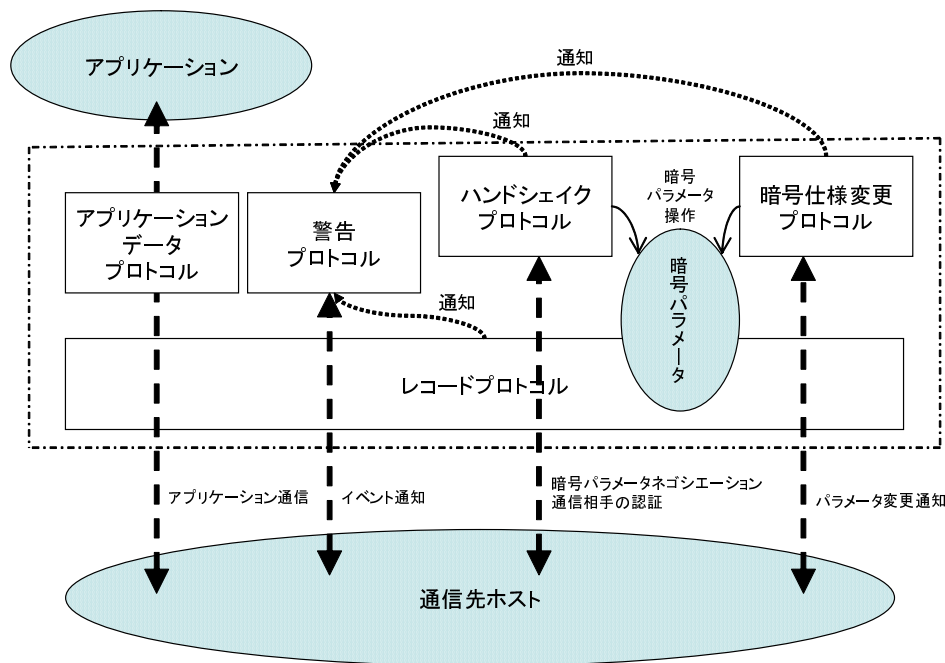


図 3.2: TLS プロトコルの全体構成概略

コルでは付加機能として、取り扱うコネクションの数を減少させるために、セッションのキャッシングという手法を導入した。さらに、通信量を減少させるための工夫も行っている。

TLS プロトコルが実現するセキュリティ機能は以下の3つである。

- 認証  
X.509 証明書を利用して、サーバ、クライアントの認証を可能とする。クライアント認証は任意実装という扱いで、サーバのみを認証する（サーバのみ認証モードと呼ぶ）こともできる。
- 守秘性  
サーバ・クライアント間の通信を暗号化することで、通信内容の漏洩を防ぐ。暗号化は共通の秘密鍵を利用する、共通鍵暗号で行われる。この共通秘密鍵の交換に、公開鍵が利用される。
- 完全性  
サーバ・クライアント間で交換されるメッセージに対して、MAC(Message Authentication Code) を付与することで、通信の完全性を保証する。MAC の計算には選択されたハッシュ関数を利用する。

これらの機能により、TLS は安全な通信を提供する。

### 3.1.2.2 ハンドシェイクプロトコル

TLS プロトコル上位層の TLS ハンドシェイクプロトコルの中核的な機能を実現するのがハンドシェイクプロトコルである。記録プロトコルにおける通信の暗号化において利用する暗号方式

や暗号鍵を決定するのがハンドシェイクプロトコルである。ハンドシェイクプロトコルでは、相互に相手認証を行い、暗号アルゴリズムや認証アルゴリズム、それぞれのパラメータを交換する。

ハンドシェイクプロトコルを中心とするサブプロトコルによって、安全なセッション確立のためのメッセージ交換を示したものが図 3.3 (p. 11) である。

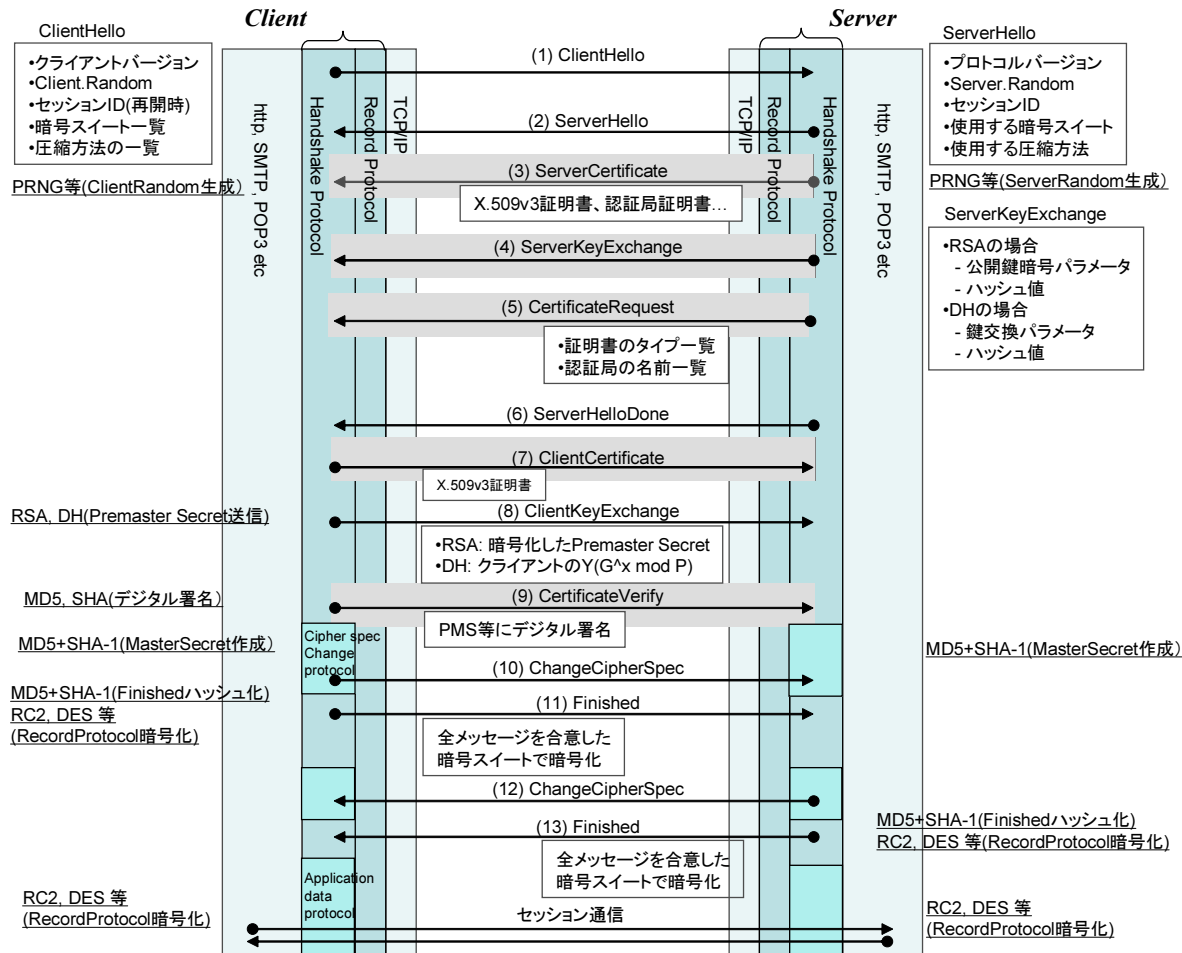


図 3.3: TLS ハンドシェイクプロトコルによるセッション確立のためのメッセージ交換フロー

図中の左側はクライアント、右側はサーバを示し、双方の内側に向かってプロトコルの階層を表している。クライアントについて言えば、最上位に http, SMTP, POP3 等のアプリケーションプロトコルが位置し、その下に TLS ハンドシェイクプロトコルが位置する。その下位層には、レコードプロトコルが位置し、ハンドシェイクプロトコルの初期には、レコードプロトコルにおいて暗号化なしで、パケットを送信する。ハンドシェイクプロトコルによる秘密鍵の交換が完了し、暗号仕様変更プロトコル(図中 Cipher Spec Change Protocol と記載)のメッセージ送出をきっかけとして、交換した秘密鍵によるパケットの暗号化を行う。レコードプロトコルの下位には、TCP/IP プロトコルが位置する。

メッセージ交換は、(1)~(13)までの番号を付した矢印の順で行われる。矢印の上にしたものをメッセージ交換名と呼ぶことにする。矢印の方向はメッセージの送信方向を示し、矢印の元の部分

にある枠には、送信するメッセージ内容を示している。また、交換する各メッセージの横に、適用する暗号技術を示している。メッセージ交換名および矢印に影がかかっているものは、仕様上オプションなメッセージである。

(1) ClientHello は、クライアントからサーバ、セッション確立のためのネゴシエーション開始を通知するメッセージ、(2) は、サーバからクライアントにネゴシエーション開始を受理するメッセージ、(3) は、サーバの公開鍵を示した X.509.v3 に従う証明書の送信、(4) は、サーバが利用可能な鍵交換アルゴリズムのリストの提示、(5) は、サーバが、クライアントに対して、クライアントの証明書を要求する。(6) は、ネゴシエーション開始要求に対するサーバからの一連回答の終了を示す。(7) は、クライアントの証明書を送信する。(8) は、クライアント内部で生成した秘密情報 Premaster Secret (PMS) をサーバと共有した暗号方式で暗号化して送信する。暗号化方式には、RSA によるものと、Diffie-Hellman によるものの2通りが選択できる。(9) クライアントは、公開鍵暗号の秘密鍵の所有をサーバに示すため PMS を暗号化して送信する。(10) クライアントからサーバに、交換した PMS を用いて生成した秘密鍵を用いた暗号通信を開始することを要求する。(11) ネゴシエーションが完了したことを示す。(12) サーバからクライアントに、交換した PMS を用いて生成した秘密鍵を用いた暗号通信を開始することを要求する。(13) ネゴシエーションが完了したことを示す。

メッセージ交換 (8) において利用される RSA は、CRYPTREC が定める推奨暗号リストの RSAES-PKCS1-v1.5 に対応し、Diffie-Hellman は、DH に対応する。

以上のようなプロセスにより、セッション通信の暗号化に必要な暗号スイートおよびその秘密鍵を安全共有することができ、セッション通信を開始することができる。

図 3.4 (p. 13) は、TLS ハンドシェイクプロトコルにおいて交換されるセッション通信の秘密鍵生成に関する鍵マテリアルの関係を示したものである。

ClientHello.Random, ServerHello.Random は、それぞれ ClientHello, ServerHello メッセージにおいてクライアント、サーバが一意的乱数として生成される。Premaster Secret は、クライアントにおいて内部的に生成され、サーバの公開鍵を用いて、サーバ、クライアント以外に漏洩しないように、サーバに送信する。ClientHello.Random, ServerHello.Random, Premaster Secret を用いて、サーバ、クライアントはそれぞれ内部的に Master Secret を生成し、さらに、Master Secret と、ClientHello.Random, ServerHello.Random を用いて、PRF(疑似乱数生成関数)を用いて Key Block を生成し、それを先頭から順に、メッセージ認証コードの鍵、セッション通信の秘密鍵(対象暗号の鍵)、秘密鍵暗号の CBC モードの初期カベクトルとして利用する。

### 3.1.2.3 暗号仕様変更プロトコル

暗号仕様変更プロトコルは、ハンドシェイクプロトコルによって通信相手と利用する暗号方式、暗号鍵のネゴシエーションを行う。

ハンドシェイクプロトコルによって通信相手と決められた合意内容をレコードプロトコルに反映させる。また、自分に反映させるだけでなく、通信相手に対しても「新しい合意内容を使ってこれから通信する」旨の通知も行っている。

ハンドシェイクプロトコルでは、暗号鍵の交換など比較的計算能力を必要とする処理を担当する。

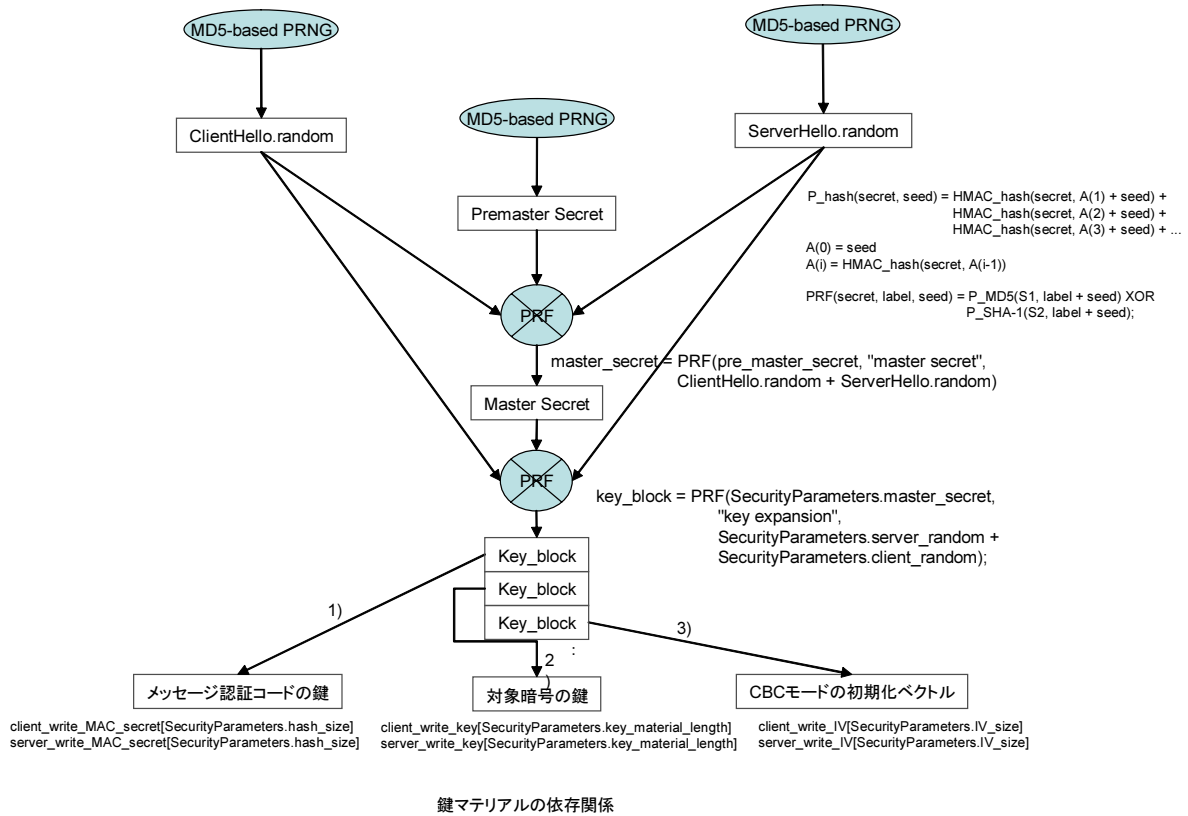


図 3.4: TLS ハンドシェイクプロトコルにおける鍵マテリアルの関係

通信相手とのネゴシエーション処理と、暗号方式の変更処理を分割することで、ネゴシエーション処理によって通信が途絶することを避けるためである。この分離アーキテクチャにより、ネゴシエーション処理とその結果の反映は独立しているため、任意のタイミングでの自由な暗号パラメータの切り替えが可能となる。

### 3.1.2.4 警告プロトコル

TLS サブプロトコルは、処理中にエラーなどのイベントが発生すると、その通知のためにエラー内容を警告プロトコルにより送信する。警告プロトコルは、通知を受けつけたイベント発生を通信相手に対して連絡する。警告プロトコルには安全な通信を実現するため暗号化機能は備えられてないが、TLS プロトコル全体としてみたときの通信処理の信頼性向上に貢献している。

発生するイベントには Warning、Fatal の 2 段階のレベルがある。Warning はユーザーにイベントが起こったことを通知し、Fatal では通信相手のホストへイベント発生を通知し、TLS 通信を切断する。

### 3.1.2.5 アプリケーションデータプロトコル

アプリケーションデータプロトコルは下位層のレコードプロトコルを通じて受信されるデータを上位層のアプリケーションに中継する。アプリケーションデータプロトコルは、TLS/SSL の実装において概念的に規定されているだけで、データに対して特別な処理は行わない。

### 3.1.2.6 レコードプロトコル

レコードプロトコルは、上位層から引き渡されたデータを暗号仕様変更プロトコルによって選択された暗号アルゴリズムと秘密鍵に基づき暗号化する。また通信データの改変を防止するために、認証アルゴリズムを利用したメッセージ認証の機能も持つ。

上記の TLS 上位層サブプロトコルはレコードプロトコルクライアントとして振舞う。将来の拡張性のために、TLS プロトコルではレコードプロトコルに対して新規のレコードタイプを追加することができるようになっている。

### 3.1.3 TLS で利用される暗号技術

TLS では公開鍵暗号を利用して、サーバ・クライアントを認証するために、いくつかの認証モードが用意されている。TLS 通信は 3 つの認証モード上での暗号通信をサポートしている。(1) 通信の双方の認証、(2) サーバのみの認証、(3) 完全な匿名の 3 つである。サーバが認証されると、通信相手を識別することが可能になるため通信はなりすまし攻撃に対して安全となる。サーバの認証には、サーバの証明書が信用のおける認証局へ通じる、有効な認証局チェーンを提供する必要がある。同様に、クライアント認証では、クライアントはサーバが受理可能な証明書を提示する必要がある。クライアントとサーバで完全な匿名セッションを利用する場合、匿名サーバがクライアントを認証することはできない。

- 暗号スイート

TLS が取り扱う暗号スイートの一覧を表 3.1.3(p. 15) に示す。

ここで暗号スイート名は、TLS プロトコルでの一意な暗号識別名である。暗号スイート名の右横の (\*) は米国の暗号輸出法律規制にて当該暗号スイートが輸出可能であることを表す。続く鍵交換・暗号・ハッシュは、それぞれ利用する暗号アルゴリズムを表す。暗号スイートは、利用する鍵交換、暗号、ハッシュアルゴリズムを組にしたもので、暗号スイートの選択によって、利用する鍵交換・暗号・ハッシュアルゴリズムを決定する。

TLS v1.0 では、TLS\_DHE\_DSS\_WITH\_3DES\_EDE\_CBC\_SHA の実装が必須とされている。TLS で用いられる 3DES は、CRYPTREC 推奨暗号リストにおける 3-key Triple DES に対応する。

- 鍵交換アルゴリズム

TLS で暗号鍵の交換に利用される鍵交換アルゴリズムの一覧を表 3.1.3(p. 15) に示す。TLS

表 3.1: TLS で利用される暗号スイート一覧

暗号スイート名	鍵交換	暗号	ハッシュ
TLS_NULL_WITH_NULL_NULL (*)	NULL	NULL	NULL
TLS_RSA_WITH_NULL_MD5 (*)	RSA	NULL	MD5
TLS_RSA_WITH_NULL_SHA (*)	RSA	NULL	SHA
TLS_RSA_EXPORT_WITH_RC4_40_MD5 (*)	RSA_EXPORT	RC4_40	MD5
TLS_RSA_WITH_RC4_128_MD5	RSA	RC4_128	MD5
TLS_RSA_WITH_RC4_128_SHA	RSA	RC4_128	SHA
TLS_RSA_EXPORT_WITH_RC2_CBC_40_MD5 (*)	RSA_EXPORT	RC2_CBC_40	MD5
TLS_RSA_WITH_IDEA_CBC_SHA	RSA	IDEA_CBC	SHA
TLS_RSA_EXPORT_WITH_DES40_CBC_SHA (*)	RSA_EXPORT	DES40_CBC	SHA
TLS_RSA_WITH_DES_CBC_SHA	RSA	DES_CBC	SHA
TLS_RSA_WITH_3DES_EDE_CBC_SHA	RSA	3DES_EDE_CBC	SHA
TLS_DH_DSS_EXPORT_WITH_DES40_CBC_SHA (*)	DH_DSS_EXPORT	DES40_CBC	SHA
TLS_DH_DSS_WITH_DES_CBC_SHA	DH_DSS	DES_CBC	SHA
TLS_DH_DSS_WITH_3DES_EDE_CBC_SHA	DH_DSS	3DES_EDE_CBC	SHA
TLS_DH_RSA_EXPORT_WITH_DES40_CBC_SHA (*)	DH_RSA_EXPORT	DES40_CBC	SHA
TLS_DH_RSA_WITH_DES_CBC_SHA	DH_RSA	DES_CBC	SHA
TLS_DH_RSA_WITH_3DES_EDE_CBC_SHA	DH_RSA	3DES_EDE_CBC	SHA
TLS_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA (*)	DHE_DSS_EXPORT	DES40_CBC	SHA
TLS_DHE_DSS_WITH_DES_CBC_SHA	DHE_DSS	DES_CBC	SHA
TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA	DHE_DSS	3DES_EDE_CBC	SHA
TLS_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA (*)	DHE_RSA_EXPORT	DES40_CBC	SHA
TLS_DHE_RSA_WITH_DES_CBC_SHA	DHE_RSA	DES_CBC	SHA
TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA	DHE_RSA	3DES_EDE_CBC	SHA
TLS_DH_anon_EXPORT_WITH_RC4_40_MD5 (*)	DH_anon_EXPORT	RC4_40	MD5
TLS_DH_anon_WITH_RC4_128_MD5	DH_anon	RC4_128	MD5
TLS_DH_anon_EXPORT_WITH_DES40_CBC_SHA	DH_anon	DES40_CBC	SHA
TLS_DH_anon_WITH_DES_CBC_SHA	DH_anon	DES_CBC	SHA
TLS_DH_anon_WITH_3DES_EDE_CBC_SHA	DH_anon	3DES_EDE_CBC	SHA

による鍵交換には RSA や Diffie-Hellman といった公開鍵暗号が利用される。鍵交換において RSA または Diffie-Hellman を使用することで、完全に匿名なセッションを確立することができる。

表 3.2: TLS で使用できる鍵交換アルゴリズム

アルゴリズム	説明	鍵サイズ制限
NULL	鍵交換無し	指定無し
RSA	RSA 鍵	制限無し
RSA_EXPORT	輸出用に制限された RSA 暗号を利用する方式	鍵長は 512 bits
DHE_DSS	DSS 署名を利用する Diffie-Hellman 方式	制限無し
DHE_DSS_EXPORT	DSS 署名を利用する Diffie-Hellman 方式	DH 鍵長は 512 bits
DHE_RSA	RSA 署名を利用する Diffie-Hellman 方式	制限無し
DHE_RSA_EXPORT	RSA 署名を利用する Diffie-Hellman 方式	DH 鍵長は 512bits,RSA は制限無し
DH_anon	署名無し匿名 Diffie-Hellman 方式	制限無し
DH_anon_EXPORT	署名無し匿名 Diffie-Hellman 方式	DH 鍵長は 512 bits
DH_DSS	DSS 証明書を利用した Diffie-Hellman 方式	制限無し
DH_DSS_EXPORT	DSS 証明書を利用した Diffie-Hellman 方式	DH 鍵長は 512 bits
DH_RSA	RSA 証明書を利用した Diffie-Hellman 方式	制限無し
DH_RSA_EXPORT	RSA 証明書を利用した Diffie-Hellman 方式	DH 鍵長は 512 bits,RSA は制限無し

RSA では鍵交換によってサーバ認証を行うことができる。サーバは公開鍵を、サーバ証明書内か、ネゴシエーション時にやりとりする一時的 RSA 鍵として取得する。一時的に RSA 鍵を利用する場合は、その鍵はサーバの RSA か DSS 証明書によって署名される。署名は、クライアントとのネゴシエーションで送信される乱数を含めて行われるため、古い一時的鍵や署名を利用することができない。こうして、クライアントは正しいサーバの公開鍵を得ることができる。サーバ証明書の検証後、クライアントはサーバの公開鍵で暗号化したメッセージデータをサーバへ送信する。サーバは自身の秘密鍵を使ってメッセージを復号し、サーバ自身であることを証明する。これで、クライアントはアクセス先が正しいサーバであることを知ることができる。

また、TLS での鍵交換では Diffie-Hellman 鍵交換を使用した認証もサポートしている。サーバは、Diffie-Hellman パラメータを含む証明書を提供するか、または RSA または DSS 証明書によって署名された一時的 Diffie-Hellman パラメータを送信する。一時的パラメータは、署名される前にクライアントから得た乱数によってハッシュされる。これでパラメータの再利用は行えない。どちらの方法でも、クライアントは証明書または署名を確認して、正しいサーバから DH パラメータを得ていることを検証できる。

公開鍵暗号を利用した鍵交換プロトコルによって、暗号アルゴリズムで利用される暗号鍵がクライアント、サーバで共有される。

- 暗号アルゴリズム

TLS でメッセージ暗号化に利用される暗号アルゴリズムの一覧を表 3.1.3(p. 16) に示す。TLS ではメッセージの暗号化に、IDEA や、RC2, RC4, DES, Triple-DES などの秘密鍵暗号を利用する。秘密鍵暗号は、あらかじめ通信の送信側と受信側で秘密鍵を共有する必要があるが、これは鍵交換アルゴリズムによって処理される。

表 3.3: 暗号アルゴリズム

アルゴリズム	説明	鍵サイズ制限
NULL	何もしない	-
IDEA_CBC	CBC モードの IDEA	128bits まで
RC2_CBC_40	CBC モードの RC2	40bits まで
RC4_40	RC4	40bits まで
RC4_128	RC4	128bits まで
DES40_CBC	CBC モードの DES	40bits まで
DES_CBC	DES	56bits まで
3DES_EDE_CBC	CBC モードの Triple-DES	168bits まで

- ハッシュアルゴリズム

TLS では、メッセージ内容の保護のためにメッセージからハッシュ関数を使って、MAC(Message Authentication Code) 値を計算し、メッセージと共に送信する。メッセージの受信側でも、同様に受信したメッセージから MAC 値を計算し、受信した MAC 値と比較することでメッセージの完全性を検証できる。TLS で利用されるハッシュ関数は、MD5 と SHA-1 である。

表 3.4: ハッシュアルゴリズム

アルゴリズム	説明
NULL	何もしない
MD5	ハッシュ関数 MD5 を利用する
SHA	ハッシュ関数 SHA-1 を利用する

- アプリケーションデータの保護

送信メッセージは、MAC 値によって完全性が保証される。また、メッセージは暗号化によって秘匿性を保つ。これらによってアプリケーションデータを保護する。

アプリケーションからの通信メッセージは送信される前に、選択されたハッシュアルゴリズム (MD5, SHA) によって MAC 値を付加されることで、メッセージの改変を防ぐ。MAC の計算では、メッセージのシーケンス番号も利用するため、メッセージの再送攻撃にも対応することができる。次に、メッセージに対して、サーバとクライアントで共有する秘密鍵を利用した秘密鍵暗号による暗号化を施す。

### 3.1.4 TLS と SSL の相違点

TLS1.0 は、SSL3.0 をもとにして IETF においてインターネット標準 RFC として策定されており、非常に似ている。主な相違点は以下の通りである。

- バージョン番号

TLS レコードフォーマットは、SSL のレコードフォーマットと基本的に同じである。違いは、レコードヘッダーのバージョン番号の値であり、TLS の場合、メジャー番号が 3、マイナー番号が 1 であり、バージョン 3.1 という位置付けとなっている。

- メッセージ認証コード

TLS はメッセージ認証コードのために RFC2104 に規定されている HMAC アルゴリズムを用いる。SSL3.0 の MAC 計算は、NMAC と呼ばれるスキームと同一方式となっており、また、認証コードの元になるデータのパディングでは、秘密鍵との排他論理和 (XOR) によるものではなく、秘密鍵の接続による点が異なる。

- Master Secret の生成

SSL は、“A”、“BB”、“CCC” という文字列を引数として、ハッシュ関数 MD5, SHA で計算したものを接続することで導出している。TLS は、図 3.4 で示した通り、疑似乱数関数 PRF を定義し、上記の 3 つの文字列と “master\_secret” という文字列から導出している。

- 警告コード

TLS は、SSL が規定している警告コードのうち、no\_certificate を除くすべてコードをサポートしている。さらに、暗号文の復号化が不正であることを意味する decryption\_failed や、ペイロード部の暗号文がオーバーフローしていることを意味する record\_overflow 等の警告コードが追加されている。

## 3.2 IPsec プロトコル

IPsec はネットワーク層において通信セキュリティを保証するプロトコルである。そのため、上位層に位置するアプリケーションの種類に依存せず、全てのアプリケーション通信は透過的に通信セキュリティを確保することができる。

IPsec は、通信送信元の認証、通信の暗号化、メッセージ認証、トンネリングによる安全な通信路の構築の機能を提供する。IPsec は、IPv4 と IPv6 での通信セキュリティを確保するために IETF で標準化活動が進められている。IPsec では、暗号化機能とメッセージ認証機能によって IP パケット全体が、パケット中に格納されているデータ部分を保護する。この章では、IPsec プロトコルの概要と、プロトコルに利用されている暗号技術についてまとめる。

### 3.2.1 IPsec の策定経緯

インターネット上の通信は、データを IP(Internet Protocol) パケットに分解し、共用通信路上で IP パケットを交換することにより実現されている。インターネットでの通信では、セキュリティへの考慮がほとんどされていないため、多くのアプリケーションは通信内容を公開したままインターネットへ送出ししている。こうした仕様に対して、インターネット通信の基盤となる IP 層でセキュリティ機能を提供するために提案されたのが IPsec である。

1992 年の第 24 回 IETF(Internet Engineering Task Force) ミーティングで IP プロトコルセキュリティとして IPsec に関する議論をするために IPSEC BoF が発足した。第 26 回 IETF ミーティングから、IPsec WG として認められ、本格的な活動を開始した。IPsec WG 発足当初は、IPv4 IPv6 のそれぞれ別々に IP セキュリティ機能の実現を目指していたが、IPSEC WG の議論の結果、両方のプロトコルから利用可能な仕様が策定され、1995 年 8 月には IPsec の最初の仕様が RFC として公開された (RFC 1825 ~ RFC 1829)。しかし、この仕様では暗号化や認証のための暗号鍵の自動管理に関する定めはなく、暗号鍵の設定は手動でやる必要があるであったため、その運用負担から当初の IPsec では実用的な利用は難しかった。

そこで鍵管理プロトコルの標準化も同時に議論された。鍵管理プロトコルとして、Sun Microsystems 社からは SKIP(Simple Key-Management for Internet Protocols) が提案された。1995 年には、米国 NSA(National Security Agency) より ISAKMP(Internet Security Association and Key Management Protocol) が、1996 年に Hilarie Orman より Oakley が提案された (p. 19, 図 3.5)。

ISAKMP (Internet SA and Key Management Protocol) は暗号鍵交換の具体的な仕組みは定めておらず、通信相手との認証、SA の生成と管理、鍵生成技法などの手順のみを規定している。Oakley プロトコルは Diffie-Hellman 鍵交換アルゴリズムを利用した、鍵交換プロトコルを規定している。これらの二つを組み合わせた ISAKMP/Oakley が提案された。IPsec の鍵交換プロトコルは、Photuris、SKIP、ISAKMP/Oakley の 3 つが候補とされ、最終的には ISAKMP/Oakley を IPsec の標準鍵管理プロトコルとして決定した。ISAKMP/Oakley は IKE(Internet Key Exchange)[34] として 1999 年 11 月に提案され、ISAKMP は RFC2408[55]、Oakley は RFC 2412[76] として提案

された。

1998年11月にはそれらの自動鍵管理プロトコルの仕様も含む形で、新たなIPsecアーキテクチャが提案された(RFC 2401 RFC 2412, RFC 2451)。この新しいIPsecアーキテクチャは、以前のものと比べて通称IPsecバージョン2と呼ばれる。

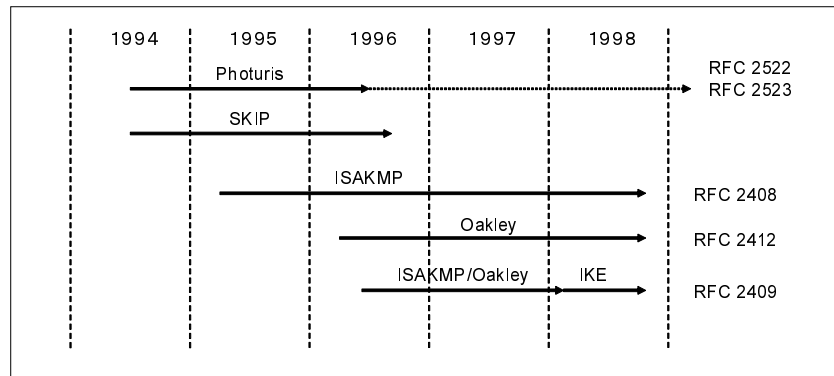


図 3.5: IPsec プロトコルに関する策定経緯 (出所 : [99])

IPsec WG とは別に、1999年には、IPsec で利用するセキュリティポリシーを扱うための方式について議論を行うIPSP(IP Security Policy) BoF や、リモートアクセス環境でのIPsecの利用について議論を行うIPSRA(IP Security Remote Access) BoF が開催され、2000年の第47回IETFミーティングで、それらのBoFは正式なWGとして活動を始めた。

IPsecは、FreeBSDやLinuxやwindowsなど、商用・非商用OSを問わず、そのネットワークスタック内に実装されている。また、ルータへの実装も登場しており、標準化されたプロトコルとして利用するが可能となっている。

## 3.2.2 プロトコル仕様

### 3.2.2.1 プロトコルの全体構成

IPsecは以下のサブプロトコル群から構成されている。

- IKE (Internet Key Exchange) プロトコル  
暗号化鍵の交換管理に利用される。
- ESP (Encapsulatin Security Payload) プロトコル  
IPパケットの暗号化データの転送に利用される。
- AH (Authentication Header) プロトコル  
IPパケットのデータの完全性と認証のために利用される。
- IPcomp (IP compression) プロトコル  
転送データのパケット単位での圧縮のために利用される。

IPsecを構成するこれらの仕様は付録中の表B.1 (p. 107)のRFCによって規定されている。これらのサブプロトコルの関係をRFCの関係に基づき示したものが図3.6である。図中の矢印はRFC

間の参照関係を示している。

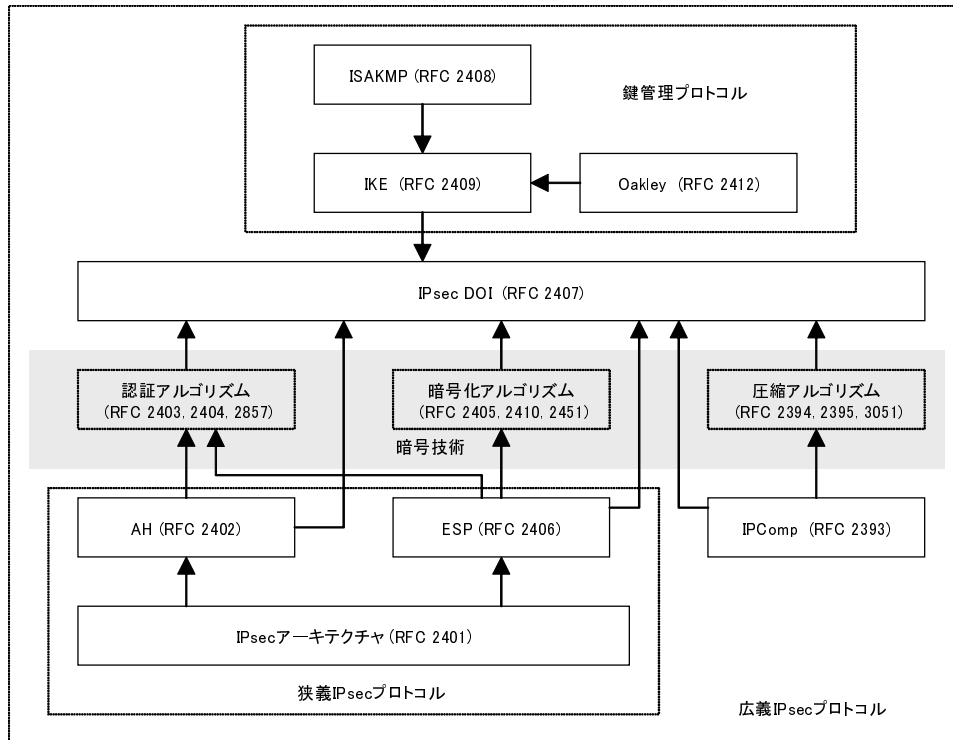


図 3.6: IPsec 構成するサブプロトコルの関係とそれらを定義する RFC([99] をもとに作成)

IPsec による通信には、トランスポートモード (図 3.7) とトンネルモード (図 3.8) がある。前者は、ホスト間で IPsec パケット出力処理と IPsec パケット入力処理を行うもので、このとき IPsec 暗号化、復号化処理が送信・受信ホストで行われるならば、図 3.7(p. 20) のホスト間トランスポート IPsec 通信となる。この場合は、それぞれのホストが IPsec 実装を備えることが必要となる。

後者は、ホスト間にセキュリティゲートウェアを挟み、セキュリティゲートウェアが、IPsec パケット出力処理と IPsec パケット入力処理を行うものである。このときはホストには IPsec 実装は必要ない。これらの処理は IPsec サブプロトコルによって実現されている。



図 3.7: ホスト間のトランスポートモード IPsec 通信 ([99] をもとに作成)

IPsec 通信は IPv4 と IPv6 のいずれもサポートしている。図 3.9(p. 21), 図 3.10(p. 21) には、それぞれ IPv4 パケット、IPv6 パケットへの IPsec の適用結果を示している。通常は (a) の状態でパケット転送が行われるが、AH によってヘッダが保護され (図中 (b))、ESP によってデータペイロード部が暗号化される。(図中 (c))。

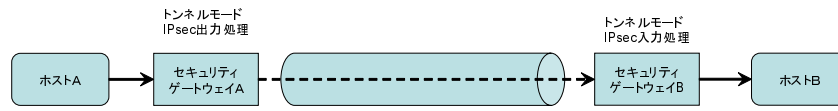


図 3.8: セキュリティゲートウェイを介したトンネルモード IPsec 通信 ([99] をもとに作成)



図 3.9: IPsec による IP データグラムのカプセル化 (IPv4)(出所 : [99])



図 3.10: IPsec による IP データグラムのカプセル化 (IPv6)(出所 : [99])

IPsec におけるパケットの出力処理および入力処理の全体構成を示したものが、図 3.11 および図 3.12 である。IP パケットの出力処理時に、あらかじめ定められた暗号通信ポリシーを SPD(Security Policy Database) から取得し、そのポリシーに対応する IPsec 通信の管理用に保持する SA(Security Association) 情報を構成して、SA に基づいて IPsec 通信を行う (図 3.11, p. 22)。IPsec パケットの受信ホストは、SA 情報から復号アルゴリズムを決定し、暗号パケットの復号を行う (図 3.12,

p. 22)。

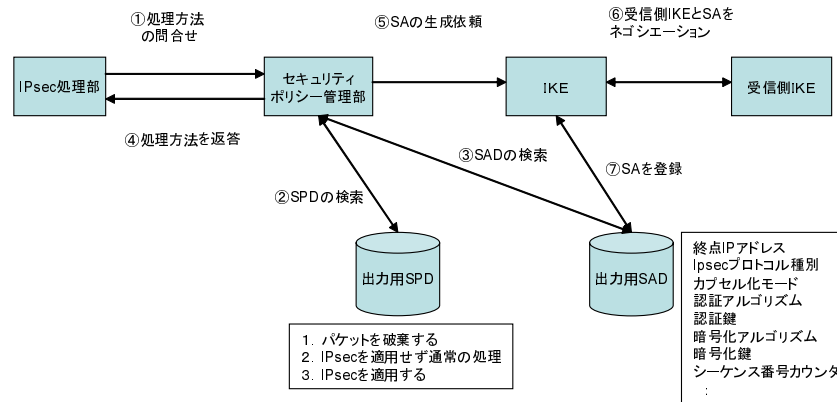


図 3.11: IPsec パケット出力処理 (出所: [99])

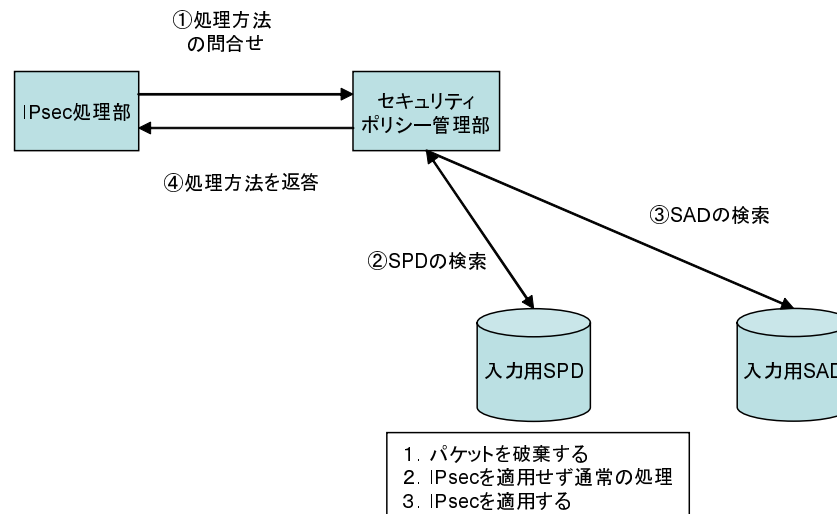


図 3.12: IPsec パケット入力処理 (出所: [99])

### 3.2.2.2 IKE プロトコル

暗号通信では、安全性をより高めるに、利用する暗号鍵を頻繁に変更することが要求される。この暗号鍵の頻繁な変更を全て手動で行うことは現実的な運用とはいえない。そこで IPsec では、自動鍵管理プロトコルについても規定されている。

IPsec の通信では、通信の送信元と送信先の端末間で利用する暗号アルゴリズムや暗号鍵を SA (Security Association) という単一方向の論理コネクションとして保持する。SA は、暗号通信での通信両端の端末間の結合関係を記述したもので、RFC 2401 で規定されている [41]。SA は AH プロトコル、ESP プロトコルが利用する暗号通信の種別を判断するために利用される。SA は暗号通

信の片方向毎に定義され、SPI 値、IPsec プロトコルの種別 (AH や ESP)、宛先 IP アドレス、カプセル化モード (トランスポートモードかトンネルモード)、認証アルゴリズムと認証鍵、暗号化アルゴリズムと暗号化鍵、SA の有効期間などのパラメータを持つ。

SA に対応して SPI (Security Pointer Index) と呼ばれる 32 ビット識別子が用意される。IPsec 通信では、パケット中の AH ヘッダと ESP ヘッダにこの SPI を格納することで、暗号通信の種別を通知する。IPsec ではパケット内に利用している暗号などの情報を直接挿入しないため、SA を合意した通信の当事者以外には利用している暗号方式などがわからないという利点を持つ。

IKE では、SA の確立までにフェーズ 1、フェーズ 2 という二つの状態を持つ。IPsec 通信の送信側と受信側で SA の合意を得るまでがフェーズ 1 である。フェーズ 1 では IPsec SA を確立する前段階として、ISAKMP SA で利用する共有秘密鍵の生成、通信相手の認証を行い、ISAKMP SA を合意する。フェーズ 2 では ISAKMP SA を利用して、IPsec SA のパラメータネゴシエーションと、共有秘密鍵の生成を行う。

IPsec では、個々のパケットに対してフロー単位 (通信元・通信先のアドレスペアや、トランスポート層ポート番号で通信を区別したもの) で通信の扱いをあらかじめ規定しておくことで、通信を制御する。通信の扱いをセキュリティポリシーと呼び、対象となる通信が利用する、暗号方式、認証方式などの情報を規定する。フロー単位でのセキュリティポリシー設定の情報は SPD (Security Policy Database) へ格納される。パケット出力処理には SPD を参照することによって、出力パケットに設定されているセキュリティポリシーを得る。次に、そのポリシーに対応する SA を SAD (Security Association Database) から呼び出す (p. 22, 図 3.11)。ここで SA が SAD に無ければ、IKE によって新しく SA を作成する。また、IPsec パケットの受信側では、受信パケットをあらかじめ設定されている SA に従って処理する。

IKE では SA の合意を外部に漏らさずに行うために自身が暗号通信機能を持っている。暗号通信のためには Diffie-Hellman 方式によって、通信元と通信先で秘密鍵を共有する。共有された秘密鍵によって暗号通信路を確立し、その通信路を使って SA の合意に至る。

図 3.13 (p. 24) は、IKE フェーズ 1 での鍵交換での代表的なメッセージ交換である事前共有秘密鍵認証を利用した鍵交換方式を示している。この方式では、始動者と応答者は、それぞれ事前に秘密鍵を共有しておく (事前共有秘密鍵)、図中の 3,4 の鍵情報交換によって、始動者、応答者は Diffie-Hellman 公開値や、乱数などを共有する。そして、これらの値から共有秘密鍵を生成し、フェーズ 2 での通信の暗号鍵として利用する。

フェーズ 2 では、フェーズ 1 で確立した ISAKMP SA を利用して、IPsec SA を確立する。このときは既に安全な通信路が用意されているため、クイックモードと呼ばれる、始動者から応答者へ SA のパラメータの提案の送信、応答者による選択の返信によって、IPsec SA パラメータのネゴシエーションと、IPsec SA で利用する共通秘密鍵の共有を行う。このようにして得られた SA を用いて、ESP プロトコル、AH プロトコルによる安全な通信が可能になる。

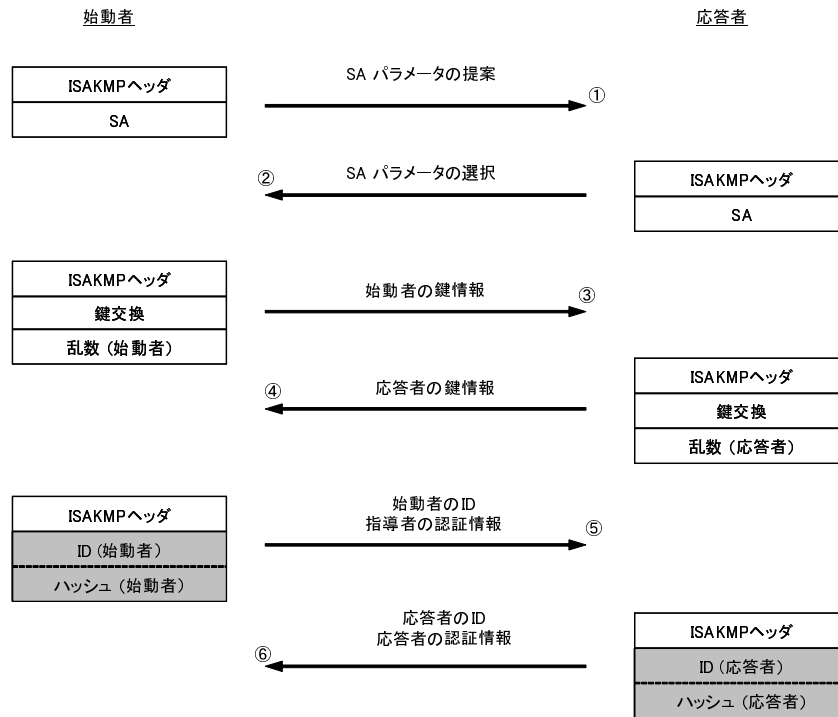


図 3.13: IKE フェーズ 1・事前共有秘密鍵認証方式を利用したメインモード (出所:[99])

### 3.2.2.3 ESP プロトコル

ESP プロトコルは、RFC 2406 に従い、IPsec のうちのパケットの暗号化による保護機能を提供する。ESP が提供するものは以下の機能である。

- データの機密性確保
- コネクションレス完全性の確保
- データ送信元の認証
- リプレイ攻撃への防御
- トラフィック情報の機密性確保

図 3.14 (p. 25) は、ESP によるパケットフォーマットを示している。ESP では IP データグラム中の通信内容を暗号化したものに SPI とデータ順序を表すシーケンス番号、認証データの 3 つをヘッダとして付加する。このヘッダを付加された暗号化データそのものも ESP と呼ばれる。この ESP に IP ヘッダが付加され、通信相手へ送信される。

ESP では、配送データを暗号化することで「データの機密性確保」を保証する。IPsec では ESP として暗号化をする対象を、IP データグラムのペイロード部のみか、IP データグラム全体とで切り替えることで、それぞれトランスポートモード、トンネルモードと呼ばれる 2 つの実行モードを切り替えることができる(それぞれのヘッダ図)。トランスポートモードでは上位層(アプリケーション

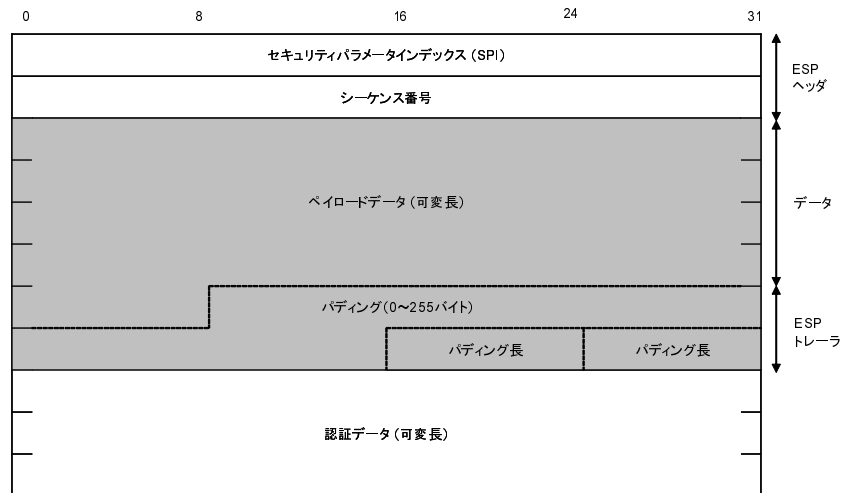


図 3.14: ESP ヘッダ

ンデータ)の保護を、トンネルモードではトンネル内を通過する IP データグラムの保護を目的としている。こうした暗号化処理は、ネットワーク中のエンドホストやネットワーク間のゲートウェイのいずれにも組み込むことができる。

IPsec パケットの受信側では、ESP ヘッダの SPI を参照することで復号のための暗号化アルゴリズムと暗号化鍵を知ることができる。続く ESP ヘッダにはシーケンス番号がある。送信側ではシーケンス番号を 1 ずつ増加し、受信側はそのシーケンス番号が順に到着しているか確認しながら受信する。シーケンス番号によって、通信を傍受した第三者によるリプレイ攻撃への防御を図っている。続いて、認証データは、ESP パケットから認証データを抜いたものから計算された ICV(Integrity Check Value: 整合性確認値)である。ICV の計算に用いる認証関数には、一方向性ハッシュ関数 (MD5, SHA-1 など) や対象鍵暗号化アルゴリズム (DES など) による MAC(Message Authentication Code: メッセージ認証コード) がある。送信側と受信側では、SA によって規定される方法で ESP パケットから認証データを計算し、「データ送信元の認証」と「コネクションレス完全性の確保」を実現する。特に IP データグラム全体を保護するトンネルモードではデータの送信元、宛先、データ内容などを完全に秘匿して「トラフィック情報の機密性確保」を提供する。IPsec では、認証データの作成に用いる認証関数に特定の関数を定めていない。

ESP が利用するデータ暗号化アルゴリズムを以下に記す。ESP では、暗号化アルゴリズムとして DES-CBC と NULL 暗号化アルゴリズムを実装することが必須となっている (RFC 2406)。

- DES-CBC (実装必須)

ESP によるペイロード部の暗号化に DES 暗号 CBC モードが用いられる。DES などのブロック暗号処理を行う暗号化アルゴリズムでは、データに対して単純に暗号化処理を続けるだけでは同じ平文が続いた場合には同じ暗号文を送信することになる。そこで、前段の暗号化データと平文データの論理排他の結果を対象にして暗号化を施す。こうすることで、同じ平文に対しても異なる暗号化データを得ることになる。一番最初のパケットには論理排他の初期値として

IV(Initialization Vector)を持つ。DES-CBCではデータを64ビットのブロック毎に暗号化するため、暗号化対象データは64ビットの整数倍になるようにパディングが付加される。ESPへのDES-CBCの利用はRFC 2405として規定されている [52]。

- その他のCBCモード暗号化アルゴリズム

その他のCBCモード暗号化アルゴリズムには3DES-CBC, RC5-CBC, IDEA-CBC, CAST-128-CBC, Blowfish-CBCが用いられる。

IPsecに標準で実装されているDES-CBCは暗号強度が弱いとされているため、DESの改良版である3DESを利用した3DES-CBCが一般的に利用されている。3DESはDESの変形で、対象となるデータに対して、異なる鍵でDESを3度適用する [93]。RC5暗号化アルゴリズムは、DESの代替となる暗号化アルゴリズムへの要求に応じて、RSA Data Security Inc. のRon Rivestによって開発された。RC5は米国特許として認められている。またRFC 2040として規定されている [8]。IDEA (International Data Encryption Algorithm) はXhejia LaiとJames Masseyによって開発された暗号化アルゴリズムである。IDEAアルゴリズムの特許は、ヨーロッパおよび米国で認可されており、日本では認可申請中である。IDEAを商用目的で利用するためにはライセンスが必要である。CAST-128はCarlisle AdamsとStafford Tavaresによって考案された暗号化アルゴリズムで、RFC 2144として規定されている [5]。これも米国特許として認められているが、フリーで公開されているため広く利用されている。BlowfishはBruce Schneierによって考案された暗号化アルゴリズムである。Blowfishには特許の制約がないため自由に誰でも使うことができる。

これらの暗号化アルゴリズムでも暗号化ブロック長は64ビットとなるため、データ長も64ビット長の整数倍となる。これらのアルゴリズムのESP暗号化アルゴリズムとしての利用はRFC 2451として規定されている [78]。

IPsecで用いられる3DESは、CRYPTREC推奨暗号リストにおける3-key Triple DESに対応する。

- AES-CBC

NISTが次世代の標準暗号化技術として選考したAES(Advanced Encryption Standard)を受け、IPsecでもAESを暗号化プロトコルとして利用するためのRFC 3602が規定されている [30]。

- NULL暗号化アルゴリズム (実装必須)

NULL暗号化アルゴリズムでは暗号化処理を行わない。このアルゴリズムは、ESPで機密性の確保を必要としない場合に利用される。しかし、認証とデータの完全性を保証することは可能である。NULL暗号化アルゴリズムはRFC 2410で規定されている。

ESPでは暗号化するデータの認証機能も持つが、認証アルゴリズムとして利用されるのは次に説明するAHプロトコルと同様のものが使用される。特に、AHと同じくHMAC-MD5-96, HMAC-SHA-1-96の実装が必須とされている。

## 3.2.2.4 AH プロトコル

AH(Authentication Header) プロトコルは、機密性の確保が必要とされない IP データグラムに対して、送信元の認証と完全性を保証する IPsec のサブプロトコルである。機密性を確保する必要がある場合には、ESP と組み合わせて利用される。AH と ESP の違いとして、AH には ESP が保護しない、パケット配送用の IP データグラムヘッダを保護することが挙げられ、AH の保証する完全性の範囲は ESP より広いものとなる。

AH が提供する機能は以下の通りである。

- コネクションレス完全性の確保
- データ送信元の認証
- リプレイ攻撃への防御

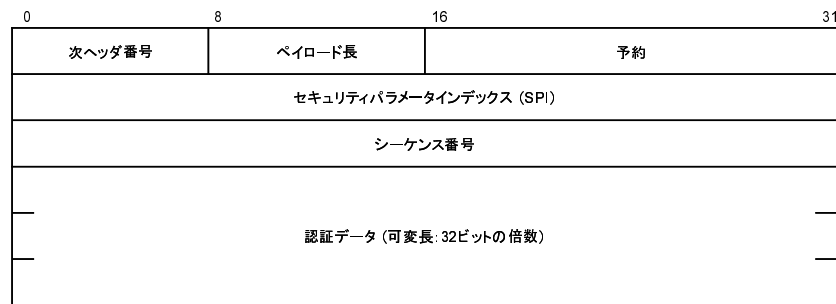


図 3.15: AH ヘッダ

AH のヘッダフォーマットを図 3.15(p. 27) に示す。AH ではデータの暗号化は行わず、SPI、シーケンス番号と認証データを IP ヘッダと IP ペイロード部の間に、AH ヘッダとして挿入する。SPI、シーケンス番号、認証データの役割は ESP と同じである。

AH では認証アルゴリズムとして、HMAC-MD5 と HMAC-SHA-1 を実装することが定められている (RFC 2402)。どのような IPsec 実装であっても、少なくともこの 2 つのアルゴリズムは備えていなければならない。また、これ以外にもいくつかの認証アルゴリズムが提案されている。

- HMAC-MD5-96 (実装必須), HMAC-SHA-1-96 (実装必須), HMAC-RIPEMD-160-96
- HMAC(Hashed Message Authentication Code) はハッシュ関数を利用した MAC 生成アルゴリズムで、RFC 2104 では他のハッシュ関数と組み合わせて認証アルゴリズムとして利用する方法が規定されている。MD5 ハッシュ関数と組み合わせて HMAC-MD5[53], SHA1 ハッシュ関数と組み合わせて HMAC-SHA1[54], RIPEMD-160 と組み合わせて HMAC-RIPEMD-160[42] という認証アルゴリズムとなる。
- SHA-1(Secure Hash Algorithm) は RFC1320 で提案されている MD4 から派生した、160 ビットのメッセージダイジェスト・ハッシュ値を計算するハッシュ関数である。IPsec での SHA 利

用については RFC 2841 として提案されている。SHA1 アルゴリズムそのものは RFC3174 として提案されている。

MD5 (Message Digest) はメッセージ全体からメッセージダイジェストとして 128 ビットのハッシュ値を計算するハッシュ関数である。RFC 1321 として規定されている

- DES-MAC, AES-MAC

DES-MAC では、認証対象のデータを DES-CBC で暗号化した結果の最後の 1 ブロック (64 ビット) を MAC として利用する方式である。DES-CBC の代わりに AES-CBC を利用する、AES-MAC も提案されている [31]。

- Keyed-MD5

Keyed-MD5 は、鍵付き MD5 と呼ばれるハッシュを利用する認証アルゴリズムである [60]。通信の送信元と送信先で鍵を共有し、対象となるデータに鍵を連結して、MD5 によるハッシュ値の計算を行うことで、認証を行う。

### 3.2.2.5 IPcomp プロトコル

IPcomp(IP Payload Compression) は、IP データグラムを単位として圧縮を行うためのプロトコルである。IPcomp は IPsec とは独立したプロトコルとして RFC 3173 に規定されている [87]。データ圧縮ではデータの特徴に注目して情報量を削減するが、IPsec によるデータの暗号化では、データの秩序は失われ、ネットワーク層以下での圧縮効率が悪化する。そこで IPsec に圧縮機能を与えるために、IPsec では IPcomp の利用がサポートされている。

## 3.3 SSH プロトコル

SSH (Secure SHell) は、TCP のポート 22 番を利用するセキュアプロトコルで、クライアント - サーバ間における通信の安全性を確保するために利用されている。ユーザから見れば、OSI 参照モデルのアプリケーション層に位置する。SSH では、ユーザ認証や安全な通信、安全なクライアント - サーバ間通信路の確立、これらのために公開鍵暗号を始め種々の暗号技術を利用する。SSH を利用することで通信内容の秘匿、通信元・通信先の認証、通信内容の保証を行うことができる。SSH は、ポートフォワーディングと呼ばれる、他のアプリケーションに対して暗号通信路を提供する機能を持ち、他のアプリケーションはその通信路を利用することで安全に通信を行うことができる。SSH には商用の実装やフリーの実装が存在し、幾つかの実装が広く利用されている。商用では SSH Communications Security 社の実装、フリーでは OpenSSH が有名である。

### 3.3.1 SSH の策定経緯

Secure Shell は、セキュリティを確保する機能を持っていない遠隔操作系コマンド (rsh、rcp、rlogin、telnet、ftp など) の代わりとして、ユーザー認証と通信内容の秘密を保証するために開発されたセキュアプロトコルである。SSH の Version 1 は 1995 年に Tatu Ylo:nen によって開発された。現在では同氏が設立した SSH Communications Security 社 (フィンランド) が中心となって SSH の開発を続けている。1996 年には SSH version2 が開発され、それを受けて IETF 内に SECSH (Sechure Shell) WG が作られた。現在のところ、SSH プロトコルを定義した RFC は発行されていないが、同 WG から SSH プロトコルのインターネットドラフトが提出され、それを基にして様々な実装が登場している。尚、version2 より前の SSH は SSH1、version2 以降の SSH は SSH2 と呼称されている。本節では主に SSH2 について述べる。

### 3.3.2 SSH プロトコル仕様

SSH 通信のプロトコルは IETF 内の SECSH (Sechure Shell) WG からインターネットドラフトとして提案されている。本節では、SECSH WG によって標準化がすすめられている SSH version2 について説明する。SSH はいくつかのサブプロトコルから構成されておりアーキテクチャ全体について記述しているものが文献 [97] である。図 3.16 はそれらのサブプロトコル群と、その位置づけを表したものである。SSH プロトコルは SSH トランスポート層プロトコル、SSH ユーザー認証プロトコル、SSH コネクションプロトコルの 3 層のサブプロトコルから構成される。

- SSH トランスポート層プロトコル

SSH トランスポート層プロトコルは TCP など信頼性のあるトランスポート層プロトコルの上で動作し、サーバの認証や、データの機密性・完全性を保証しながら通信を行う。また、通信データの圧縮機能も有する。SSH トランスポート層プロトコルはインターネットドラフト [98] として提案されている。

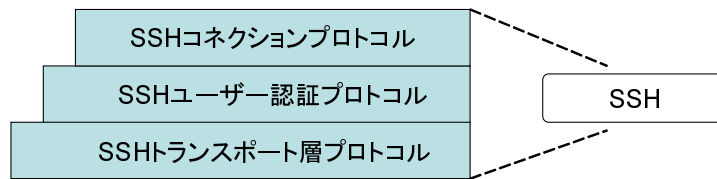


図 3.16: SSH サブプロトコル群

- SSH ユーザー認証プロトコル  
サーバがクライアントのユーザーを認証することを可能にするためのプロトコルである。本プロトコルは SSH トランスポート層プロトコル上で動作する。このプロトコルもインターネットドラフト [95] として提案されている。
- SSH コネクションプロトコル  
SSH コネクションプロトコルは、SSH ユーザー認証プロトコル上で動作し、他のアプリケーションに対して提供する複数の暗号化通信路を論理チャンネルとして多重化することを可能とする。このプロトコルもインターネットドラフト [96] として提案されている。

### 3.3.2.1 プロトコルの全体構成

SSH プロトコルのアーキテクチャ全体を示しているのがインターネットドラフト [97] である。SSH 通信では、まず SSH トランスポート層プロトコルによって安全な通信路をサーバとクライアント間で確立する。次に、サーバは SSH ユーザー認証プロトコルに従って、クライアントの認証を行う。それから、SSH コネクションプロトコルによって暗号通信の論理チャンネルが確立され、ユーザーはそのチャンネルを使って安全な通信を行うことができる。ポートフォワーディング機能も SSH コネクションプロトコルによって実現される。

SSH で使用する、暗号化アルゴリズム名、MAC アルゴリズム名、公開鍵アルゴリズム名、鍵交換アルゴリズム名、プロトコル名などの名前やプロトコル番号は IANA によって定められている [49]。

以下は SSH サブプロトコル群についての説明である。

### 3.3.2.2 SSH トランスポート層プロトコル

SSH トランスポート層プロトコルは、SSH の最下層に位置するプロトコルである。このプロトコルでは、鍵交換のアルゴリズムや、公開鍵暗号のアルゴリズム、ハッシュ関数、共通鍵暗号のアルゴリズム、通信メッセージの MAC のアルゴリズム、これらを、クライアント、サーバ間でネゴシエーションにより統一する機能、暗号通信の機能、ホスト認証の機能、データ完全性の保護の機能などを提供する。このプロトコルでの認証は、通信相手のホストを認証するもので、ユーザー認証ではないことに留意する必要がある。ユーザー認証はより上位のプロトコルで規定される。また、SSH トランスポート層プロトコルでは通信データの圧縮機能も提供される。

SSH トランスポート層プロトコルの提供する機能は、安全な暗号通信路の確立で、そのために暗号鍵の交換を行う。SSH トランスポート層プロトコルは、暗号鍵の交換技術と、それを取り巻く関連技術を含んでいる。まず関連要素から説明する。

- ホスト鍵

SSH サーバはクライアントがアクセス先サーバを認証できるように、ホスト鍵と呼ばれる暗号鍵を持つ。サーバは複数のホスト鍵を持つことができる。SSH は DSA(NIST によって制定された DSS の一部であるデジタル署名アルゴリズムであり、離散対数問題に基づく公開鍵暗号をベースとする) の実装を必須としており、ホストは DSA の公開鍵をホスト鍵として必ず持つ。このホスト鍵は後述の「鍵交換」処理において、鍵交換相手をホスト認証するために利用される。

ホスト認証のためにはあらかじめ認証対象ホストであるサーバとホスト鍵の対応を知っておく必要がある。この対応情報の提供に二つの信頼モデルを使うことができる。一つは、クライアント側で対応データベースを運用する方法で外部インフラなどを必要としないが運用負荷が高い問題がある。もう一つは、対応関係の表を信頼性のある CA(証明局) で認証する方式である。この方式ではクライアントは必要とする対応が正しいものを、CA を利用して検証でき、保守の手間は省けるが、サーバ側では自身のホスト鍵に CA からの証明を受けておく必要がある。現在よく用いられている方式では、最初の通信時はサーバの検証をせず、サーバからホスト鍵を取得し、次回からの通信時に通信相手の同一性を保障するために、そのホスト鍵を利用するという処理を行っている。

- プロトコルバージョンの交換

SSH 通信では、最初にクライアントとサーバ間でプロトコルバージョンの識別文字列を交換する。文字列のフォーマットを表 3.3.2.2 (p.31) に記す。SSH 通信は、現在 version 2 が主流となっているが、このプロトコルバージョンの交換という手続きがあるため、通信の下位互換性が保たれており、新旧のサーバ・クライアントの混在環境であっても SSH 通信は成立する。

表 3.5: SSH のバージョン情報の例

	識別子
フォーマット	"SSH-(プロトコル・バージョン)-ソフトウェアバージョン・コメント", CR/LF
サーバ例	SSH-1.99-OpenSSH_3.4p1 CR+LF
クライアント例	SSH-1.5-TTSSH/1.5.4 win32 CR+LF

- SSH バイナリパケットフォーマット

SSH 通信では通信を暗号化する。暗号化通信には表 3.3.2.2 (p.32) で示すフォーマットのバイナリパケットを利用する。暗号化を施されるのは、パケット長、パディング長、ペイロード、ランダムパディング、これらの部分である。ペイロードの最大長は無圧縮状態で 32,768 バイトまでとされている。

- 圧縮機能

圧縮処理を利用する場合は、ペイロード部のみをネゴシエーションで決められたアルゴリズムによって圧縮する。パケット長と MAC 値は圧縮後のパケットから計算する。暗号化

表 3.6: SSH パケットのバイナリフォーマット

名称	長さ	説明
パケット長	4 バイト	MAC とこのフィールドを除いたパケットのバイト長。このフィールドも暗号化される。
パディング長	1 バイト	パディングのバイト長。
ペイロード	1 バイト	通信データ本体。圧縮機能を利用する場合は圧縮済データとなる。(ペイロード長:1 = パケット長 - パディング長 - 1)
ランダムパディング	m バイト	パケット長を、暗号ブロック長と 8 のいずれか大きいほうの倍数バイトにするために 255 バイトまでの乱数をパディングとしてパケットに付加する。
MAC	n バイト	メッセージ認証用のフィールド。メッセージ認証機能を利用する場合はメッセージ認証コードが入る。

もまた圧縮後に行う。現在のところ圧縮アルゴリズムとして規定されているのは、無圧縮と zlib(LZ77)[22][21] を利用する圧縮アルゴリズムである。圧縮機能の実装は任意で必須ではない。

- サービス要求

SSH トランスポート層プロトコルでネゴシエーションを終了すると、クライアントはサービス名で指定したサービス要求を送信する。現在定義されているサービス名は、「ssh-userauth」と「ssh-connection」である。サーバでそのサービスの利用が許可されると、受託通知がクライアントへ送られる。それぞれのサービスについては、「SSH ユーザー認証プロトコル」と「SSH コネクションプロトコル」として次節以降に説明する、

ここまでは SSH トランスポート層プロトコルの基本技術を説明した。同プロトコルの主要な役割は暗号鍵の交換のサポートである。ここから暗号鍵の交換アルゴリズムと、交換する暗号鍵の暗号アルゴリズムを用途別に説明する。

- 暗号アルゴリズムと暗号鍵の決定

通信路暗号化 (パケットの暗号化) に利用する暗号アルゴリズムと暗号鍵は、鍵交換の際に決定される。暗号アルゴリズムの決定は送受信の方向を考慮し、通信の送受信で異なる暗号アルゴリズムが利用できることが必須となっている。表 3.3.2.2 (p.33) は規定されている暗号アルゴリズムの一覧である。

- メッセージ認証アルゴリズムの決定

MAC(Message Authentication Code:メッセージ認証) 値を計算することで、データの完全性保護を提供する。SSH トランスポート層プロトコルのドラフトで定義されているメッセージ認証アルゴリズムが表 3.3.2.2 (p.33) である。鍵交換の際にメッセージ認証アルゴリズムと鍵を決定し、その認証アルゴリズムに対して、認証するパケット自身 (MAC フィールドを除く) と、

表 3.7: SSH 暗号アルゴリズム一覧

識別名	実装要求	暗号アルゴリズム
3des-cbc	必須	3DES-CBC
blowfish-cbc	任意	128 ビット鍵の Blowfish-CBC
twofish256-cbc	任意	256 ビット鍵の Twofish-CBC
twofish-cbc	任意	Twofish256-cbc の別名 (歴史的経緯により用意)
twofish192-cbc	任意	192 ビット鍵の Twofish
twofish128-cbc	任意	128 ビット鍵の Twofish
aes256-cbc	任意	256 ビット鍵の AES (Rijndael)-CBC
aes192-cbc	任意	192 ビット鍵の AES-CBC
aes128-cbc	推奨	128 ビット鍵の AES-CBC
serpent256-cbc	任意	256 ビット鍵の Serpent-CBC
serpent192-cbc	任意	192 ビット鍵の Serpent-CBC
serpent128-cbc	任意	128 ビット鍵の Serpent-CBC
arcfour	任意	ストリーム暗号 ARCFOUR
idea-cbc	任意	128 ビット鍵の IDEA-CBC
cast128-cbc	任意	128 ビット鍵の CAST-128-CBC (RFC2144)
none	任意	暗号化しない (非推奨)

パケットのシーケンス番号 (0 から  $2^{32} - 1$  のパケット順序番号、送信パケット毎に 1 ずつ増加する)、共有秘密を入力することで MAC 値を計算し、SSH バイナリパケット内の MAC フィールドに挿入する。MAC フィールドは暗号化されずに転送される。MAC フィールド長は選択する認証アルゴリズムによっても異なる。

表 3.8: SSH メッセージ認証アルゴリズム一覧

識別名	実装要求	認証アルゴリズム	MAC 長	鍵長
hmac-sha1	必須	HMAC-SHA1	20	20
hmac-sha1-96	推奨	first 96 bits of HMAC-SHA1	12	20
hmac-md5	任意	HMAC-MD5	16	16
hmac-md5-96	任意	first 96 bits of HMAC-MD5	12	16
none	任意	認証無し (非推奨)		

- 公開鍵アルゴリズムの決定

SSH トランスポート層プロトコルでは公開鍵アルゴリズムも規定している。現在のドラフトで規定されている公開鍵アルゴリズムは表 3.3.2.2 (p.34) である。公開鍵暗号は、SSH ユーザー認証プロトコルにおいて利用される。

表 3.9: SSH 公開鍵アルゴリズム一覧

識別名	実装要求	公開鍵アルゴリズム
ssh-dss	必須	Raw DSS
ssh-rsa	推奨	Raw RSA
x509v3-sign-rsa	任意	X.509 certificates (RSA)
x509v3-sign-dss	任意	X.509 certificates (DSS)
spki-sign-rsa	任意	SPKI certificates (RSA)
spki-sign-dss	任意	SPKI certificates (DSS)
pgp-sign-rsa	任意	OpenPGP certificates (RSA)
pgp-sign-dss	任意	OpenPGP certificates (DSS)

SSH で用いられる RSA 公開鍵暗号は、CRYPTREC 推奨暗号リストにおける RSASSA-PKCS1-v1\_5 に対応する。

- 鍵交換アルゴリズム

SSH 通信では、上述のように、通信路暗号化 (パケットの暗号化) を共通鍵暗号を用いて行うが、その共通鍵をセッション鍵と呼ぶ。セッション鍵の鍵交換として Diffie-Hellman 方式を利用する鍵交換が実装必須として規定されている [48]。鍵交換によって、サーバ・クライアントで one-time セッション鍵を共有する。

- アルゴリズムネゴシエーション

SSH プロトコルは、上述のとおり多様な暗号技術をサポートしており、特定の暗号技術には依存しない。そのため、サーバとクライアントで利用する暗号アルゴリズムを一致させる必要がある。そのためにアルゴリズムネゴシエーションを行う。

サーバ、クライアントの双方は通信相手からアルゴリズムを指定するネゴシエーションパケットが到着する前に、相手が指定するアルゴリズムを推測して、リクエストを送ってもよいことになっている。このとき、推測が外れたときは、通信相手が指定してきたアルゴリズムを使うことを通知する。

鍵交換アルゴリズム/通信路暗号化アルゴリズム/メッセージ認証アルゴリズム/圧縮アルゴリズムのネゴシエーションでは、クライアントが利用を宣言しているアルゴリズムをサーバが利用可能ならば、そのアルゴリズムを利用する。このときサーバ側で利用可能なアルゴリズムが存在しなければ接続は切断される。

アルゴリズムネゴシエーションはネゴシエーションパケットの交換から始まる。ネゴシエーションパケットは、利用できる暗号・認証・圧縮アルゴリズムなどの一覧を、通信の片方向毎にそれぞれ指定して交換する (表 3.3.2.2 (p.35))。ここでそれぞれのアルゴリズムとして IANA で定められた識別名のリストが記述される。

表 3.10: 鍵交換パケット

情報	説明
鍵交換アルゴリズム	利用する鍵交換アルゴリズムを記述する
サーバホスト鍵アルゴリズム	サーバがホスト認証に利用するアルゴリズム
クライアント暗号アルゴリズム	クライアントが通信路暗号化に利用可能な暗号アルゴリズムの一覧
サーバ暗号アルゴリズム	サーバが通信路暗号化に利用可能な暗号アルゴリズムの一覧
クライアント認証アルゴリズム	クライアントが通信の完全性保護に利用可能なメッセージ認証アルゴリズムの一覧
サーバ認証アルゴリズム	サーバが通信の完全性保護に利用可能なメッセージ認証アルゴリズムの一覧
クライアント圧縮アルゴリズム	クライアントが利用可能な圧縮アルゴリズム
サーバ圧縮アルゴリズム	クライアントが利用可能な圧縮アルゴリズム

### 3.3.2.3 SSH ユーザー認証プロトコル

SSH ユーザー認証プロトコルでは、サーバが SSH トランスポート層での接続を確立したクライアントの通信可否をユーザー認証によって判断する。このサブプロトコルでは SSH トランスポート層上でのユーザー認証の方法が定義されている [95]。

SSH ユーザー認証プロトコルで利用できる認証方式には、公開鍵認証、パスワード認証、ホストベース認証の 3 つがある。このうち公開鍵認証のみが、実装必須とされている (表 3.3.2.3 (p.36))。ここで規定されている認証は、クライアントホストではなく、ユーザーの認証であるため、どの認証方法であっても、クライアントはユーザー名を指定してサーバへの接続を試みる。サーバ側はあらかじめ、ユーザー認証のための情報を持っておく必要がある。

- Authentication Request

SSH ユーザー認証プロトコルでは、まずクライアントからの認証要求パケット送信から始まる。この認証要求パケットには、クライアントのユーザー名と利用する認証方式の要求が含まれる。

- Authentication Response

認証要求パケットを受信したサーバは、認証応答を返信する。認証応答によって、クライアントが指定した認証方式での認証を開始する。クライアントが指定した認証方式をサーバがサポートしていない場合は、サーバは切断要求を認証応答として送信する。

- 公開鍵認証

SSH 認証プロトコルでは、公開鍵認証のみが実装必須となっている。この認証方式では、ユーザーが自身の秘密鍵から作成した署名を送信する。サーバはその署名をあらかじめ保持していた

表 3.11: ユーザー認証方式

識別名	方式	説明
publickey	公開鍵認証	サーバはユーザーの自己証明をユーザー自身の秘密鍵を利用した署名によって、検証することで認証とする。この方式は SSH では実装必須
password	パスワード認証	クライアントユーザーしか知り得ないパスワードを共通秘密として認証に利用する
hostbased	ホストベース認証	クライアント端末の秘密鍵を利用してクライアントの自己証明を行う。サーバはクライアントがログイン可能な端末かどうかを判断して、認証とする。

ユーザの公開鍵で検証し、ユーザを識別してユーザ認証とする。

サーバが利用可能な公開鍵暗号アルゴリズムは、すでに SSH トランスポート層プロトコルで交換されて、クライアントにとって既知となっているため、クライアントは利用できる公開鍵暗号アルゴリズムの公開鍵の一覧を送信する。サーバはその公開鍵認証要求を受け入れ可能ならば、その受け入れ可能通知をクライアントへ返信する。するとクライアントは、指定された公開鍵暗号を利用して、署名を作成し、サーバへ送付する。サーバは、あらかじめ保持しているユーザーの公開鍵を利用して、署名を検証し、認証を行う。

公開鍵認証は、クライアントユーザが自身の秘密鍵を安全に保持するという前提がある。ユーザの秘密鍵は、クライアントホストへ暗号化された形で保管され、その利用にはパスワードを必要とするように設定することができる。これによって、秘密鍵の盗難リスクを軽減することができる。

- ホストベース認証

この認証方式は特定のクライアントからのアクセスを許可する場合に利用される。クライアントは自身の秘密鍵を利用した署名をサーバへ送信し、サーバはあらかじめ保持していたクライアントの公開鍵を利用して署名の検証を行う。サーバは、そのクライアントが認証許可を持っている端末であれば、次にクライアントが指定するユーザー名がホストベース認証でログイン可能なアカウントであるかを判断し、認証を行う。

この認証はユーザーでなく、ホストの秘密鍵を利用して行われるため、個別のユーザーがホストの秘密鍵を得ることを出来ないように設定しておく必要がある。また高い安全性を要求するサイトでは、ホストベース認証は利用しないほうがよい。この認証方式の実装は完全に任意とされている。

ホストベース認証では、クライアントは安全であるという前提を必要とし、この前提が失われた場合のリスク軽減の対策は、この方式には用意されていない。

- パスワード認証

サーバはあらかじめ、ユーザー名とパスワードの組を保持する。パスワード認証では、クライ

アントはサーバにユーザー名とパスワードの平文（ただし SSH トランスポート層での暗号通信路上で転送されるため通信内容は秘匿される）の組を送信する。サーバでは受け取った、ユーザー名とパスワードを保持するデータベースから検証し、認証を行う。

パスワード認証は、サーバが安全であるという前提を持つ。もしサーバが安全でないならば、パスワード認証を利用することで、攻撃者にユーザー名とパスワードの組を与える可能性がある。公開鍵認証では、このようなリスクは発生しない。

#### 3.3.2.4 SSH コネクションプロトコル

SSH コネクションプロトコルは、SSH トランスポート層が用意され、ユーザー認証が行われた後に、対話型セッション、リモートコマンドの実行、TCP ポートフォワーディング、X11 コネクションフォワードの4つのサービスをユーザーに対して提供する。これらのサービスは、一つの暗号通信路へ多重化される。

それらのサービスは、チャンネルと呼ばれる論理通信路を利用して行われる。チャンネルは、通信の両端でそれぞれ一意な番号を与えられて区別される。それぞれのサービスを要求するたびに、チャンネルが複数用意される。

対話型セッション、リモートコマンドの実行は、ssh が rsh(remote shell) の代替として動作するための基本的な仕組みである。遠隔端末との間の通信は暗号化されて、安全に遠隔端末を操作することが可能となる。

TCP フォワーディングは、クライアントとサーバ間で、特定の TCP 通信に対して安全な暗号化通信路を用意する。この技術を利用することで、特定のアプリケーションの通信をクライアントとサーバ間で安全に保つことができる。アプリケーションの通信は、サーバへ到着すると、暗号化通信路から出力され、通常通信（暗号化無し）として目的ホストへ転送される。

X11 は、UNIX 系 OS で広範に利用されている GUI(Graphical User Interface) システムの X window のことである。X window では、もともと遠隔操作の機能を持っているが、X11 フォワーディングでは、遠隔操作のための転送技術である。X11 フォワーディングで転送される X11 通信は、SSH の暗号化通信路を利用して転送される。

#### 3.3.3 SSH1 の SSH2 に対する主な暗号利用の相違点

SSH1 では、上述の SSH2 のようにサブプロトコル3つに分かれておらず、鍵交換や通信路暗号化が一つのプロトコルの中に収まっている。ここでは、暗号利用に絞って SSH1 の SSH2 に対する主な相違点・共通点を列挙する。

- SSH1 ではホスト鍵は SSH2 と同様に保持する。
- SSH1 では SSH2 と同様に共通鍵暗号で以って通信路を暗号化する。
- SSH1 では SSH2 のようなセッション鍵の Diffie-Hellman 鍵交換は行わない。サーバがサーバ鍵という公開鍵をホスト鍵とは別に用意し、ホスト鍵と共にクライアントへ送る。セッション

鍵はクライアント側が生成し、ホスト鍵とサーバ鍵で2重に公開鍵暗号化してサーバに送ること  
でセッション鍵を共有するという独特の方法が取られている。

- SSH2 がユーザ認証プロトコルを独立したサブプロトコルとして規定しているのに対して SSH1  
ではユーザ認証を独立したプロトコルとしては規定していない。SSH1 では、SSH2 におけるホ  
ストベース認証によって、ユーザ認証を代替しているとも言える。

## 第4章

# セキュアプロトコルに関する既知の攻撃法

本章では、セキュアプロトコルに関する既知の攻撃法についてまとめる。

### 4.1 攻撃法の調査範囲

プロトコルに対する攻撃法を網羅的に収集するために、図 4.1 (p. 39) に示す手順で調査範囲を設定した。

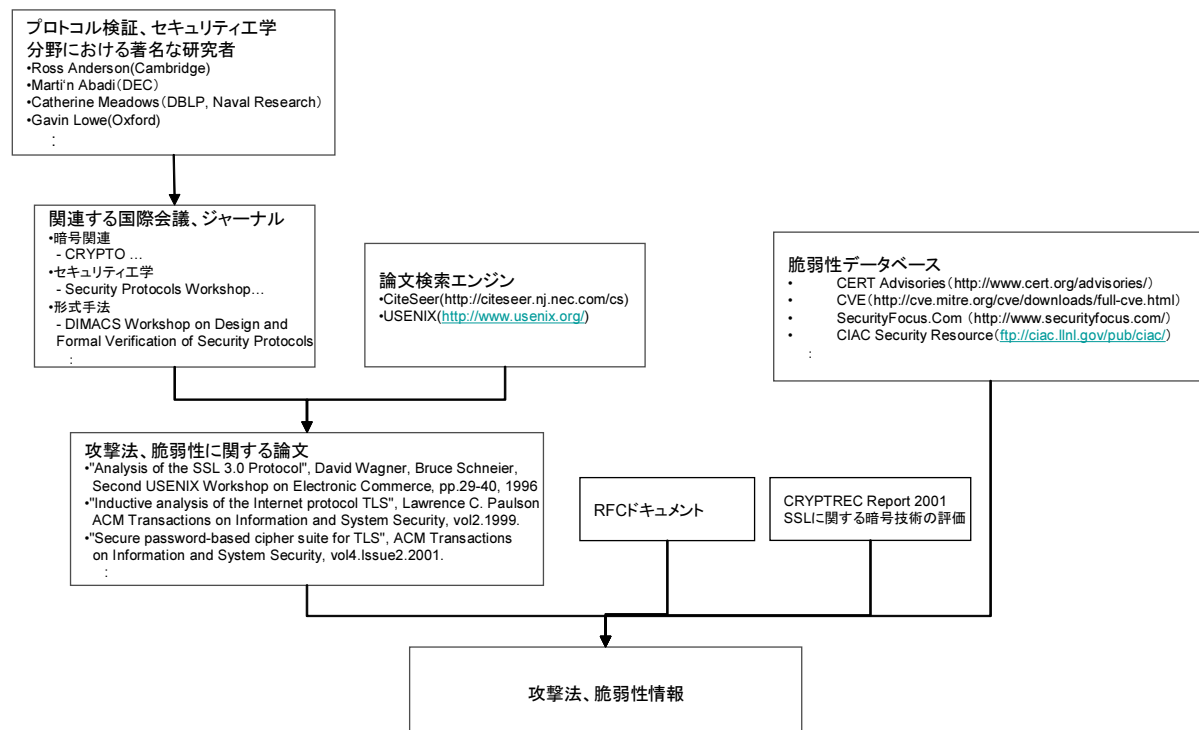


図 4.1: セキュアプロトコルの既知の攻撃法に関する調査手順

プロトコルの検証、セキュリティ工学の分野では、一定の研究者が中心となって、理論的な面から検討を行っているため、これらの分野における主要な研究者を探索した。次に、これらの研究者の論文およびその参考文献から、関係する国際会議、学術論文誌等を網羅的に収集した。これらの情報源から、調査対象とするプロトコルに関連する研究を抽出するとともに、インターネットの google 等の一般の検索エンジン、および CiteSeer 等の論文専門の検索エンジンを用いて、関連文献を抽出した。一方、製品レベルから脆弱性を網羅的に収集するために、CVE 等の脆弱性情報データベースからセキュアプロトコルの脆弱性をキーワード検索により抽出した。具体的な調査情報源を付録 C にまとめた。

また、IETF におけるインターネット標準 RFC、インターネットドラフトおよび CRYPTREC Report 2001 の SSL に関する暗号技術の評価からも攻撃法に関する情報を収集した。

今回調査対象としたセキュアプロトコルの TLS/SSL, IPsec, SSH は、いずれも、インターネット上で極めて広く普及しているプロトコルであり、これらに対する攻撃による被害影響は極めて大きいものとなると言える。したがって、たとえ小さな脆弱性、攻撃法であっても、その影響は大きいと、上記の情報源のいずれかで公開される可能性が極めて高いと言える。そのため、考慮するに値する現実味のある攻撃法であれば、その影響度の大きさから、上記の主要な情報源により網羅的に収集された情報に含まれないことはあまり考えられない。実際、収集した攻撃法の中には、極めて低い確率で可能になる解読法や、極めて初歩的なプロトコルの設定上の問題などを前提にしてはじめて、可能である攻撃法がいくつも見られた。本調査では、このような現実的ではない攻撃法では対象とせず、上記の情報源に含まれる既知の攻撃法を対象として分析を行うこととした。

## 4.2 TLS/SSL

第 4.1 節の手順に従い TLS/SSL の攻撃法を収集し、実装に関する攻撃法を除いたプロトコル仕様に関する攻撃法を抽出し、分析を加えてまとめた。CVE 脆弱性情報データベースから得られた TLS/SSL に関する攻撃法は 48 件であり、そのうちプロトコル仕様に関する攻撃法として 8 件抽出した。CRYPTREC Report 2001 (SSL に関する暗号技術の評価の章) で挙げられている SSL に関する攻撃法の内、本調査においてはプロトコル仕様に関する攻撃法として 2 件抽出した。RFC に挙げられている脆弱性のうち、本調査では、上記の事例と重複するものも含めて 5 件抽出した。残りは学術論文からの収集により全体で、18 件の攻撃法を抽出した。

表 4.2 (p.42) ~ 4.4 (p.44) は、TLS/SSL プロトコルに関する既知の攻撃法、脆弱性について分析結果をまとめたものである。

表中の攻撃箇所は、図 3.3 (p. 11) において示したメッセージ交換のサブプロトコルまたはメッセージ交換ステップ名に対応する。

表 4.2 (p.42) 中の各項目の内容の説明は表 4.1 (p.41) の通りである。

攻撃法分類および形式手法適用可否については、総合分析編でさらに分析を加える。

もっとも多くみられる攻撃法は、暗号適用不備に関するもので、その内、認証不備、応答情報漏洩が多数を占めている。認証不備は、メッセージ認証コードを十分に検証しないなど論理的な認証の不備が含まれ、また、応答情報漏洩には、暗号アルゴリズムの動作において、アルゴリズム自体に

表 4.1: 攻撃法分析表の項目説明

項目名	内容説明
ID	攻撃法を識別するために一意に設定したコード
攻撃法分類	攻撃法の特徴により類別した分類
プロトコル	プロトコル種別、バージョン等
攻撃箇所	攻撃対象サブプロトコル、処理フェーズ
攻撃法・脆弱性	攻撃法・脆弱性の内容
原因	攻撃の原因となる問題点
被害影響	攻撃の結果生じる被害影響等
対策	攻撃に対する対策
形式手法適用可否	形式手法による検証の適用可能性
情報源	攻撃法の情報源

は違反していないものの、応答情報を解析することにより鍵情報の一部が解読されるなど脆弱性が含まれる。プロトコルメカニズムの不備に関するものとしては、Ciphersuite Rollback 攻撃のように、メッセージ交換の順序の厳格性を欠くものや、プロトコル完了処理が不十分であるために、途中のメッセージを削除されるなどの例が挙げられる。

表 4.2: TLS/SSL に対する既知の攻撃法 (1/3)

ID	攻撃法分類	プロトコル	攻撃箇所	攻撃法、脆弱性	原因	被害影響	対策	形式手法 検証可否	情報源
TLS1	暗号適用不備 (応答情報漏洩)	TLS, SSL	ハンドシェイクプロトコルにおけるサーバへのRSA暗号文送信時 (ClientKeyExchange など)	<b>(Bleichenbacher攻撃)</b> サーバに送られた暗号文ブロックを復号化したものが、バイト列00または02で開始しない場合には、RSAの正規のコーディングではないと判断してエラー応答を返すように実装した場合※、正規のコーディングの場合と応答の違いが観測できる。暗号文ブロックを巧妙に作成し、サーバの反応を確認しながら、合成する暗号文を精密かしていくことにより、暗号文ブロックの解読が可能になる。解読に必要な応答数は1024bitRSAの場合2の20乗である。 (注意※:この実装はRSAセキュリティ社より既にPKCS#1v2.0以降警告されている。)	RSA復号処理における振舞いの違いによる情報漏えい	TLSのPremaster secret 漏洩。	1)非正規のPKCS#1メッセージに対する応答を一定にする。 2)ClientKeyExchangeの復号化後のメッセージ長、開始バイトを検証する。	統計的な攻撃のモデル化は不可。	RFC2246, CVE-1999-0007
TLS2	暗号適用不備 (応答情報漏洩)	SSL, TLS (RSAオプションを選択した場合)	ハンドシェイクプロトコル:サーバRSA鍵生成時	<b>(RSAタイミング攻撃)</b> RSA形式の秘密鍵を生成する際の処理時間を計測することで、秘密鍵を解読するタイミング攻撃を受ける問題が存在する。この問題を利用する攻撃により、リモートの攻撃者は標的サーバの秘密鍵を奪取する可能性がある。統計的手法を使用して、ある暗号鍵に対するRSA復号化操作を完了するのに必要な時間を注意深く測定することで、攻撃者はRSAキーの要素の1つ(q)を回復される可能性がある。以後、公開鍵と要素qを用いることで攻撃者に秘密鍵が漏洩する恐れがある。	RSA秘密鍵生成時の振舞いの違いによる情報漏えい	鍵交換用秘密鍵の漏洩	RSA暗号処理の応答を均一化する。	統計的な攻撃のモデル化は不可。	CAN-2003-0147
TLS3	暗号適用不備 (応答情報漏洩)	SSL3.0, TLS1.0	ハンドシェイクプロトコル: ClientKeyExchangeメッセージ送信時	<b>(Klima-Pokorny-Rosa攻撃(修正型Bleichenbacher攻撃))</b> RSA暗号文を含む、サーバに対してPKCS#1 v1.5パディングを用いた数百万の規模のSSL 3.0/TLS 1.0のコネクションを開かせることで、暗号文と平文の対応関係を漏洩する脆弱性があり、サーバは、選択された暗号文に対するサーバのRSA秘密鍵を用いた復号化した平文を漏洩する。	RSA復号化における平文漏洩	Premaster secret 漏洩、RSAキーは漏洩しない。	RSA復号化結果の平文を厳密にチェックする。	統計的な攻撃のモデル化は不可。	CAN-2003-013
TLS4	暗号適用不備 (応答情報漏洩)	SSL 3.0, TLS 1.0 以前	アプリケーションデータプロトコル動作時のレコードプロトコル、または、アラートプロトコル	<b>(Vaudenayタイミング攻撃(Timing-based Attacks on SSL/TLS with CBC Encryption))</b> 攻撃者により置き換えられた暗号文ブロックを含むSSLおよびTLS接続を受信しても、改ざんされた暗号文ブロックに対して拒否や接続の強制終了を行わない。データの復号化の際に、ブロック暗号のパディングによるエラーを発見した後のメッセージ認証コード(Message Authentication Code)の妥当性確認が省略されているため、ブロック暗号のパディングによるエラー、もしくはメッセージ認証コードが妥当でないために生じるエラーの2つの異なるエラーケースを見分けることが可能である。そのため、リモートの攻撃者はエラーに対応する応答時間を測定し、パディング内容に対する総当たり攻撃を試みることで、ブロックの内容を取得することが可能になる。	CBCモード暗号における認証処理における振舞いの違いによる情報漏洩	セッション時暗号文のブロックの内容の盗聴	改ざんされた暗号文ブロックを拒否し、接続を強制終了するようなメッセージ認証	統計的な攻撃のモデル化は不可。	CAN-2003-0078, SecurityFocus bid 6884

表 4.3: TLS/SSL に対する既知の攻撃法 (2/3)

ID	攻撃法分類	プロトコル	攻撃箇所	攻撃法、脆弱性	原因	被害影響	対策	形式手法 検証可否	情報源
TLS5	暗号適用不備 (相手認証不備)	TLS, SSL	ハンドシェイクプロトコル: ServerKeyExchange メッセージ交換	X.509 証明書の Basic Constraints 領域には、中間認証局 (中間 CA) が他の認証局に署名できる権限を持っているかどうかの確認を行う情報が含まれる。この確認を十分に行わないことにより、ルート認証局が信頼できるものであれば、その証明書自体を信頼してしまう。	証明書認証不備	成り済ましによる暗号文の機密性の喪失	Basic Constraints 領域の確認	証明書内部の認証モデル化困難	CAN-2002-0862、CAN-2002-1407
TLS6	オプション強度不備	TLS, SSL	ハンドシェイクプロトコル (ClientHello, ServerHello, ServerKeyExchange などにおける鍵交換認証モード設定時)	(TLSに対する成り済まし攻撃 (Man-in-the-middle攻撃)) SSL, TLSには、オプションにより1)双方認証,2)サーバ認証, 3)匿名の3つのモードがある。匿名モードを選択した場合には、未承認のサーバ証明書に基づくRSAあるいはDiffie-Hellmanによる鍵交換において成り済ましによるpremaster-secretの奪取が可能になる。ただし、受動的な盗聴では、premaster-secretは漏洩しない。	認証なしのオプション	premaster secretの漏洩	サーバ、クライアントの双方認証モードを用いる	認証モデル化による検証可	RFC2246
TLS7	プロトコルメカニズム (認証不備)	SSL3.0	ハンドシェイクプロトコル (ClientHello, ServerHelloにおけるセッション再開時)	(Version rollback攻撃) SSL2.0互換サーバは、セッション再開時に、SSL2.0が設定されたClientHelloメッセージを受け付け、SSL3.0からSSL2.0にダウングレードする脆弱性がある。	セッション再開時のClientHelloメッセージの認証不備	バージョン2.0へのダウングレードにより安全性の低下	SSL3.0のセッション再開時にSSL2.0への移行を制限する。	認証のモデル化による検証可能	RFC2246, Thomas
TLS8	暗号適用不備 (擬似乱数生成不備)	SSL	ハンドシェイクプロトコル (ServerKeyExchangeによるDH鍵交換のバリエーション)	Diffie-Hellman鍵交換において利用されるDSA認証方式で、各々のメッセージに対して乱数である秘密鍵を生成する際に偏りがある。これによりメッセージの完全性に問題が生じる可能性がある。	擬似乱数生成の偏り	メッセージの完全性に問題が生じる可能性	FIPS186-2 Change Notice1に基づく乱数生成機の改良	乱数生成のモデル化不可	CRYPTREC
TLS9	暗号適用不備 (相手認証不備)	SSL, TLS	ハンドシェイク: ServerKeyExchange メッセージ	クライアント、サーバ認証において、証明書の検証は、最新版のCRL (証明書破棄リスト) を用いていない場合、攻撃者は、成り済まし攻撃が可能になる。	証明書の検証が不十分	認証に失敗し成り済まし攻撃による秘密鍵の漏洩など	最新のCRLを用いて証明書を検証する。	CRL確認のモデル化可能	RFC2246
TLS10	暗号適用不備 (メッセージ認証不備)	SSL	ハンドシェイクプロトコル (ServerKeyExchangeの認証)	(Key-exchange algorithm rollback攻撃) ServerKeyExchangeメッセージ内のKeyExchangeAlgorithmが署名対象になっていない点を利用し、サーバにはephemeral DH鍵交換アルゴリズム、クライアントには、RSAを使用するように改竄しても、気づかれない。	ServerKeyExchangeメッセージの認証不備	セッション中の全メッセージを盗聴される。通信者になりすまして改竄したメッセージを送信できる。	ServerKeyExchangeメッセージの長さをチェックする	認証のモデル化による検証可能	Wagner
TLS11	暗号適用不備 (メッセージ認証不備)	SSL	暗号仕様変更プロトコル、ハンドシェイクプロトコル (Finishedにおけるメッセージ認証)	(Dropping the change cipher spec message攻撃) ChangeCipherSpecメッセージが、Finishメッセージでメッセージ認証対象になっていないことを利用し、ChangeCipherSpecメッセージを通信路で削除することにより、通信者が暗号化なしでセッションを開始させる。	Finishメッセージ中で、ChangeCipherSpecメッセージを検証しないため	セッションの全通信が盗聴、改竄できる。	ChangeCipherSpecメッセージを確認してから、Finishedメッセージを送信する。	認証のモデル化による検証可能	Wagner

表 4.4: TLS/SSL に対する既知の攻撃法 (3/3)

ID	攻撃法分類	プロトコル	攻撃箇所	攻撃法、脆弱性	原因	被害影響	対策	形式手法 検証可否	情報源
TLS12	プロトコルメ カニズム (メッセージ 認証不備)	SSL2.0	ハンドシェイクプロト コル(ServerHello, Finishedにおける暗 号スイート平文)	(Ciphersuite Rollback 攻撃) hello message 中に暗号文でかかれたサポートする暗号スイ ートを改竄する。SSL3.0では、finished メッセージのメッセージ認証 により回避。	Finishメッセ ージ検証前に セッション通 信を開始する ため	暗号強度の弱い パラメータを指定 し、その後の通信 の盗聴、改竄を可 能にする。	SSL3.0以降の Finishedメッセ ージ認証を行う。	認証のモ デル化に よる検証 可能	Paulson, CRYPTRE C
TLS13	プロトコルメ カニズム (メッセージ 認証コード の応答漏 洩)	SSL	ハンドシェイクプロト コル	(Finished Message攻撃) 攻撃者が、SSLサーバに対して client hello メッセージにより セッションの再開を要求すると、master_secretのMACを生成し finished メッセージを返す。Finishedメッセージに返答しないこと により、コネクションを開いたままにする。(不正な返答をすると、 エラーによりコネクションが再開できなくなるため。)これらの要求 を繰り返し多数送信することで、平文とmaster_secretにより生成 されたMACを多数入手でき、master_secretを推定する可能性が ある。	メッセージ認 証コードの応 答漏洩	master_secretが 解読される可能 性あり。	TLSのようにハッ シュ関数にHMAC を利用する。	統計的な 攻撃のモ デル化は 不可。	Wagner
TLS14	暗号脆弱性	SSL	レコードプロトコルの 暗号化処理	(トラフィック解析攻撃) ストリーム暗号を用いる場合、暗号メッセージのサイズから平文 メッセージのサイズが分かる。	ストリーム暗号 の脆弱性	セッション中の情 報が盗聴される可 能性がある。	ブロック暗号を用 いる。	統計的な 攻撃のモ デル化は 不可。	Wagner
TLS15	プロトコルメ カニズム (メッセージ 認証不備)	SSL	警告プロトコル (ClosureAlertの処理 時)	(Premature Closure攻撃) 攻撃者がセッション中のメッセージを削除しても、通信者に気づ かれない。	ClosureAlert メッセージの 不備	通信者の一方ま たは双方が一部 の情報しか受け 取れないことによ る、完全性の欠 如。	ClosureAlertメッ セージを正しく処 理する。	メッセ ージ交換 の 手順モ デル化可	Thomas
TLS16	プロトコルメ カニズム(平 文生成不注 意)	TLS, SSL	ハンドシェイクプロト コル(ServerHelloの生 成時)	(Session ID脆弱性) SSL仕様では、SessionID選択に自由度がある。SessionIDは、 ClientHello, ServerHelloメッセージで平文で送信されるため、 サーバによるSessionID選択の際には、秘密情報を使わないよう 注意が必要。	Session ID生 成の不注 意	SessionIDに含ま れる秘密情報の 漏洩。	サーバによる SessionID選択の 際には、秘密情報 を使わないよう注 意が必要。	平文情報 の意味モ デル化不 可	RFC2246

## 4.3 IPsec

IPsec プロトコルに関する攻撃法を分析整理したものが表 4.5 (p.46) ~ 4.7 (p.48) である。CVE 脆弱性情報データベースから得られた攻撃法は 11 件あり、そのうちプロトコル仕様に関する攻撃として 3 件抽出した。RFC から得られた攻撃法は 1 件である。その他、学術論文、インターネット情報から得られたものを合計し 13 件の攻撃法を抽出した。

もっとも多く見られる攻撃法は、暗号適用不備によるもので、そのなかでも相手認証不備、メッセージ認証不備を原因とするものである。特に、ブロック暗号文を含むパケットの一部のデータを他の同一セッション鍵を用いたパケットとの間で切り張りすることによるセッションの乗っ取りなどの認証の不備に対して能動的に攻撃するものや、初期化ベクトルの漏洩による暗号文の改竄に関する攻撃などが特徴的である。また、RSA 暗号に対するタイミング攻撃は、TLS/SSL と同様に、IPsec についても、IKE プロトコルにおいて攻撃対象となっている。また、認証コード不可と暗号化およびフラグメンテーションの適用順序から、生じる脆弱性に対する攻撃が重要なものとして挙げられる。

表 4.5: IPsec に対する既知の攻撃法 (1/3)

ID	攻撃法分類	プロトコル	攻撃箇所	攻撃法、脆弱性	原因	被害影響	対策	形式手法 検証可否	情報源
IPSEC1	暗号適用不備 (応答情報漏洩)	IPsec	IKE: デジタル署名認証方式または公開鍵暗号証明方式の場合	<b>(RSAタイミング攻撃)</b> RSA 形式の秘密鍵を生成する際の処理時間を計測することで、秘密鍵を解読するタイミング攻撃を受ける問題が存在する。鍵交換プロトコルIKEにおいて、この問題を利用する攻撃により、リモートの攻撃者は標的サーバの秘密鍵を奪取する可能性がある。統計的手法を使用して、ある暗号鍵に対する RSA 復号化操作を完了するのに必要な時間を注意深く測定することで、攻撃者は RSA キーの要素の 1 つ (q) を回復することが可能である。以後、公開鍵と要素 q を用いることで攻撃者は秘密鍵の計算が可能になる。	RSA秘密鍵生成時の振舞いの違いによる情報漏洩	鍵交換用秘密鍵の漏洩	RSA暗号処理の応答を均一化する。	統計的な攻撃のモデル化は不可。	CAN-2003-0147
IPSEC2	暗号適用不備 (相手認証不備)	IPsec	IKE: デジタル署名認証方式または公開鍵暗号証明方式の場合	X.509 証明書の Basic Constraints 領域には、中間認証局 (中間 CA) が他の認証局に署名できる権限を持っているかどうかの確認を行う情報が含まれる。鍵交換プロトコルIKEにおいて、この確認を十分に行わない場合、ルート認証局が信頼できるものであれば、その証明書自体を信頼してしまう。	証明書認証不備	成り済ましによる暗号文の機密性の喪失	Basic Constraints 領域の確認	証明書内部の認証モデル化困難	CAN-2002-0862、CAN-2002-1407
IPSEC3	暗号適用不備 (メッセージ認証不備)	IPsec	ESP(IPsec)CBCモード暗号化	<b>(セッションハイジャック (Cut-and-paste攻撃の一種))</b> 不用意に同一のセッション鍵を2つの異なる通信において用いた場合、CBC暗号のself-healing propatyを利用して、正規通信者間の通信のESPメッセージのうち、一部を切り取り、攻撃対象の通信者間のESPメッセージの一部と置き換える。攻撃者は、両者の計算機へのログイン権限を持たなくても攻撃可能である。	異なる通信でセッション鍵を共有したため。	セッションが乗っ取られる。	異なる通信でセッション鍵を共有しない。	CBCモードのブロック特性のモデル化は困難。	Bollevin
IPSEC4	暗号適用不備 (メッセージ認証不備)	IPsec	ESP, AHプロトコル後のフラグメンテーション処理	<b>(フラグメンテーション攻撃)</b> ESP, AH処理が、パケットのフラグメンテーションの後に実行される場合、攻撃者は、ネットワーク上のパケットを取得し、IPヘッダーのフラグメントオフセットを0に書き換えて再送する。受け取った通信者は、正規のパケットとして処理することになる。この脆弱性を利用して、通信者に、フラグメンテーション境界の後のヘッダー内容を改竄した大きなパケットを送信させることができる。	ESP, AH処理が、パケットのフラグメンテーションの後に実行されるため。	IPヘッダーの改竄	AH, ESP処理をフラグメンテーションの前に実行する。または、IPsecトンネルモードを利用する。	フラグメンテーション処理のモデル化は困難	Bollevin
IPSEC5	暗号脆弱性とUDPプロトコルの干渉	IPsec	ESP: ブロック暗号CBCモード利用時	<b>(初期化ベクトル脆弱性)</b> 初期化ベクトル(Initialization Vector(IV))は平文で送信され、復号化ではIVとの排他論理和によって行われることが分かっているため、攻撃者は、最初の暗号ブロックに復号文に生じる変更が予測可能な改竄を行うことができる。通常、UDPパケットには、ヘッダー部全体が最初のブロックに収まるため、パケットを新しい接続の開始に利用することができる。	初期化ベクトルを平文で送信すること。	新しい接続開始に利用される。	初期化ベクトルを安全に共有するか、第一暗号文ブロックのメッセージ認証を十分に行う。	ブロック暗号処理のモデル化困難	RFC1858

表 4.6: IPsec に対する既知の攻撃法 (2/3)

ID	攻撃法分類	プロトコル	攻撃箇所	攻撃法、脆弱性	原因	被害影響	対策	形式手法 検証可否	情報源
IPSEC6	暗号適用不備(暗号化適用順序の不備)	IPsec	AH, ESPプロトコル	(被暗号化ハッシュ関数攻撃) 完全性のチェックは、パケットとそのハッシュ値に暗号化を行っているため、選択平文を用いたCut-and-paste攻撃の対象になる[Bollevin]。	ハッシュ値付加と暗号化の順序と初期化ベクトルに関する初期ブロックの情報漏洩の脆弱性が組合わさるため。	通信文の改竄。	暗号化適用順序をハッシュ付加の前に行う。	ブロック暗号処理のモデル化困難	Bollevin
IPSEC7	プロトコル干渉(応答情報漏洩)	IPsec	トンネルモードESPプロトコル	(Reading short blocks脆弱性) telnetセッションにおいて、トンネルモードESPかTCPタイムスタンプオプションが使用された場合、パケットが1文字のデータ送信をする際に、ESPパケットの7番目のバイトは5(padding)、8番目のバイトは6(TCP)か4(IP-in-IP)と特定の定数であり、1番目のバイトは文字コードのいずれかであることが分かる。TCPチェックサムにより、1番目のバイトが正しければACKパケットが返り、正しくなければ応答がないため、1番目の正しい値を知ることができる。選択平文攻撃を用いて、1番目のバイトの各値について暗号文を取得したパケットを送信し、その応答を観測することで1番目のバイトを知ることができる。	TCPチェックサムとの干渉により、応答情報が漏れる。	暗号通信文の最初のバイトが解読される。	不明	ESPパケットのデータフォーマットモデル化困難	Bollevin
IPSEC8	暗号適用不備(暗号化適用順序の不備)	IPsec	ESP, AHプロトコル	(Security Association脆弱性) マニュアル鍵トランスポートモードAHプロトコルのSAによる通信は、replay攻撃に対する耐性を持たない。この2者が、トランスポートモード暗号化オンリーESPのSAを交換し、AHとESPのSAによる通信を行う。その後しばらくして、新たなESPのSAを、前と同じSPIにより設定するとする。攻撃者は、前のパケットを、後の通信において送信する。受信者は、AHによりパケットが正規であるとみなし、新たなESPのSAで復号化することにより、認証を欺くことができる。	メッセージ認証が不十分である。	AHプロトコルの認証をすり抜ける。	メッセージ認証を暗号化の前に行い、送信メッセージの全ビットに対する認証を行う。	暗号技術適用手順のモデル化難	Ferguson

表 4.7: IPsec に対する既知の攻撃法 (3/3)

ID	攻撃法分類	プロトコル	攻撃箇所	攻撃法、脆弱性	原因	被害影響	対策	形式手法 検証可否	情報源
IPSEC9	暗号適用不備(相手認証不備)	IPsec	ISAKMPプロトコル	(Identity Protection exchangeの脆弱性) Identity Protection Exchangeにおいて、メッセージ交換の開始者(イニシエータ)は、相手を認証する前にアイデンティティを送信する。能動的攻撃者は、応答者に成りすまして、イニシエータが通信を中断するまでアイデンティティを収集することができる。	相手認証をするまにアイデンティティを送信するため。	イニシエータのアイデンティティの漏洩	公開鍵暗号方式を利用することで回避される。	認証のモデル化可能	Ferguson
IPSEC10	暗号適用不備(暗号化の不備)	IPsec	IKE アグレッシブ・モード	(VPN brute force 攻撃) 事前共有鍵方式(パスワード)を用いた IKEアグレッシブ・モードでは、IPSECトンネルを確立したいVPNクライアントの最初のパケットへの応答として、事前共有鍵に基づく認証ハッシュが送付される。このハッシュは暗号化されていないため、スニフアーを使ってこれらのパケットを捕捉し攻撃者が事前共有鍵にオフライン・ディクショナリ(brute force)攻撃をするために必要なすべての情報を手に入れることが可能になる。	事前共有鍵に基づく認証ハッシュの送付に際して、ハッシュが暗号化されていないため	事前共有鍵の解読	リモート・アクセスVPNにハイブリッド・モードを使用するか、アグレッシブ・モードを使用しない。	ハッシュ暗号化のモデル化可能	CheckPoint
IPSEC11	暗号適用不備(相手認証不備)	IPsec	IKEプロトコル:メインモード、事前共有秘密鍵認証鍵方式	(Reflection攻撃) メインモード、事前共有秘密鍵認証鍵方式による鍵交換の場合、応答者に成りすました攻撃者が、イニシエータと同じIDを主張し、認証を通過することができる。	不明	応答者に成り済ます。	メインモード、事前共有秘密鍵認証鍵方式以外を用いる。	相手認証のモデル化可能	Ferguson
IPSEC12	暗号適用不備(メッセージ認証不備)	IPsec	IKEプロトコル	(Proposal攻撃) ハッシュ計算に 応答者からの SA返答が含まれないため、攻撃者は、イニシエータが送信したメッセージのペーロードに含まれる提案するSAの優先順序を改竄することができる。これにより、強度の弱いSAを使用させることができる。	応答者からのSAに関してハッシュ値に含めないため	強度の弱いSAをイニシエータに選択させる。	応答者からのSAをハッシュ値に含める。	ハッシュ対象のモデル化可能	Ferguson
IPSEC13	暗号適用不備(秘密情報の推測漏洩)	IPsec	ESPプロトコル	(Fluhrer攻撃) ESP初期化ベクトル選択法が予測可能な場合、攻撃者は、適応型選択平文攻撃を行うことができる。	ESP初期化ベクトル選択法が予測されるため。	暗号化通信文が盗聴される。	初期化ベクトルを予測されない生成アルゴリズムを用いる。	確率的攻撃のモデル化不可	Nuodpone

## 4.4 SSH

CRYPTO、EUROCRYPTO、USENIX 等の学会・文献サイトにおいては、SSH の中で攻撃可能な暗号等の脆弱性の発表例はあるが、SSH 自体が関係する脆弱性の発表例はない。論文検索サイト Citeseer では SSH の脆弱性について約 30 件の報告があるが、大部分は個別実装に関する脆弱性であり、SSH の仕様上の脆弱性および実装に多く見られる脆弱性は 2 件であった。SSH の脆弱性報告が最も充実しているのは CERT で、約 40 件程の報告があるが、やはり大部分が個別実装に関する脆弱性であり、SSH の仕様上の脆弱性および実装に多く見られる脆弱性は 5 件であった。その他脆弱性報告サイトにおける傾向も同様であり、SSH の仕様上の脆弱性および実装に広範に見られる脆弱性として収集できたのは 3 件であった。なお、SSH の個別実装に関する脆弱性は各ベンダーからも逐次報告されている。

SSH プロトコルに関する攻撃法、脆弱性を分析整理したものが表 4.8 (p.50), 4.9 (p.51) である。

SSH はその固有のプロトコル部分は少ないため、固有のプロトコル部分の仕様が単独で脆弱性を発現しているケースはごく少数である。SSH は多様な暗号アルゴリズムを選択可能としているため、それら暗号の持つ特性が SSH の中で脆弱性となってしまうケースもある。また、耐タンパー性やサイドチャネル攻撃耐性が低いという、最も一般的な実装上の脆弱性が SSH 実装においてもよく見られる。

### 参考文献の対応関係

攻撃法分析表中の情報源に関する記号は以下のものを意味する。

- **CVE**, **CAN** で始る記号 : CVE 脆弱性情報データベース <http://cve.mitre.org/cve/>
- **RFC** で始る記号: IETF インターネット標準 RFC <http://web.mit.edu/network/ietf/sa/>
- **SecurityFocus bid** で始る記号 SecurityFocus 脆弱性情報データベース <http://www.securityfocus.com/>
- **VU#** で始る記号 脆弱性情報データベース CERT Advisory <http://www.cert.org/advisories/>
- **JPSJ CSEC** 情報処理学会 コンピュータセキュリティ研究会 <http://www.sdl.hitachi.co.jp/csec/>
- **CRYPTREC**: CRYPTREC 暗号技術評価報告書 (2001 年度版)
- **Wagner**: David Wagner and Bruce Schneier. Analysis of the ssl 3.0 protocol. In Second USENIX Workshop on Electronic Commerce, pages 29–40, 1996 (1997 revised).
- **Thomas**: Stephen Thomas, SSL and TLS Essentials, Wiley Computer Publisher, New York, 2000.
- **Paulson**: Lawrence C. Paulson. Inductive analysis of the internet protocol TLS. Technical report, Computer Laboratory, University of Cambridge, 1997. Technical Report440.
- **Bellovin**: Steven Bellovin. Problem areas for the ip security protocols. In Proceedings of

表 4.8: SSH に対する攻撃法 (1/2)

ID	攻撃法分類	プロトコル	攻撃箇所	攻撃法、脆弱性	原因	被害影響	対策	形式手法 検証可否	情報源
SSH1	暗号適用不備(メッセージ認証不備)	SSH1	CFBモードIDEAの復号時	暗号化にCFBモードのブロック暗号を使った場合、ブロックの改竄は復号時に検出できるが、ブロック暗号がIDEAの場合、最終ブロックではこの検出が無効になる。	ブロック暗号において復号方式の差異が考慮されていないため	セッション通信におけるブロックの改竄	SSH1においてIDEA使用しない	暗号ブロックのモデル化困難	VU#315308
SSH2	暗号適用不備(メッセージ認証不備)	SSH1	CFBモードブロック暗号の復号時	ブロック暗号に対する選択平文攻撃が可能でありかつセッション傍受可能な攻撃者に対しては、VU#315308と同様の脆弱性がIDEA以外のブロック暗号使用時にも存在する。	ブロック暗号の選択平文攻撃に対する耐性不備	セッション通信におけるブロック改竄	攻撃検知のコード	暗号ブロックのモデル化困難	VU#13877
SSH3	プロトコルメカニズム(その他)	SSH1	RC4におけるパケットの巡回冗長検査時	SSH1では、RC4で暗号化されて送られてきたパケットを統合する際の巡回冗長検査(CRC)において使われるchecksumが変更可能である。攻撃者は改竄したパケットに整合するようにchecksumを変えることによってCRCをパスする可能性があり、その場合にはパケットの改竄が発見されない。	パケット検査欠陥	パケット改竄(パケット落ちを含む)	SSH2を使用する。	チェックサムデータのモデル化困難	VU#25309
SSH4	暗号適用不備(その他)SSH1実装とPKCS#1の相乗	SSH1	セッション確立時	SSH1側に暗号化時の内部ステータスが外部から取得可能という実装上の問題があった場合、PKCS#1.1.5の公開鍵暗号でBleichenbacherの脆弱性と組み合わせることで秘密鍵無しに暗号文から平文が復号できる。	SSH1実装の不備とPKCS#1の不備の相乗	セッション鍵の漏洩	SSH2を使用する。	確率的攻撃のモデル化不可	VU#161576
SSH5	暗号適用不備(その他)パスワード認証と鍵生成の不整合	SSH1	パスワード認証時	RC4を使いつつパスワード認証を行う場合の脆弱性である。RC4を使う場合、サーバはセッションIDを暗号化に使うがクライアントは使わない。さらにパスワード認証では、サーバ側もセッションIDは使わない。攻撃者がRC4を使いつつreplay connectionを求めてもサーバは拒否しない。	パスワード認証と鍵生成の不整合	一定時間のセッション漏洩	SSH2を使用する。	パスワード処理不正後のモデル化不可	VU#665372

表 4.9: SSH に対する攻撃法 (2/2)

ID	攻撃法分類	プロトコル	攻撃箇所	攻撃法、脆弱性	原因	被害影響	対策	形式手法 検証可否	情報源
SSH6	プロトコルメ カニズム(メッ セージ認証 不備)	SSH	公開鍵ユーザ認証時	「Version Rollback攻撃」によりSSH2の利用者であっても、SSH1の利用を強制させ、さらには、最も脆弱な暗号化アルゴリズム、かつ、パスワードを用いたユーザ認識方式を強制することができる。	バージョンの情報のメッセージ認証不備	中間攻撃と併用してパスワードを奪取される。	バージョン情報のメッセージ認証を行う。	メッセージ認証のモデル化可能	JPSJ,CSEC No.022- 014
SSH7	暗号適用不 備(相手認 証不備)	SSH1	公開鍵ユーザ認証時	公開鍵ユーザ認証におけるメッセージのやり取りにおいて、メッセージの正当性を検証するための情報が不十分(メッセージに格納されていない)ため、鍵交換時に成りすましが可能になる。	相手認証の不備	成済ましによる通信内容の漏洩。	SSH2を使用する。	相手認証のモデル化可能	pa.bell- labs.com/ ~abadi/ Papers/ secspec
SSH8	暗号適用不 備(応答情 報漏洩)	SSH1, SSH2	SSH USER Authenticationプロ トコルのRSA鍵生成時	RSA形式の秘密鍵を生成する際の処理時間を計測することで、秘密鍵を解読するタイミング攻撃を受ける問題が存在する。この問題を利用する攻撃により、リモートの攻撃者は標的サーバの秘密鍵を奪取する可能性がある。統計的手法を使用して、ある暗号鍵に対するRSA復号化操作を完了するのに必要な時間を注意深く測定することで、攻撃者はRSAキーの要素の1つ(q)を回復することが可能である。以後、公開鍵と要素qを用いることで攻撃者は秘密鍵の計算が可能になる。	秘密鍵生成時の応答情報漏洩	鍵が漏洩する。	RSA秘密鍵生成の応答を均一化する	確率的攻撃のモデル化不可	Security Focus 7101
SSH9	機能設計不 備	SSH1, SSH2	サーバ側のコマンド 処理	シングルオーテーション(*)で括ったコマンド名を与えることによりリモートからのコマンドの実行が許可されている。このためクライアントは直接ログインせずにプログラム実行可能である。	コマンド処理のチェック不備	ログイン無しでプログラム実行される。	コマンド処理の改善	機能仕様上の検証不可	Security Focus 4547
SSH10	機能設計不 備	SSH1, SSH2	ログイン時	loginプロセスにおいてサーバの環境変数にクライアント側の環境変数を渡して設定することができてしまう。	環境変数設定チェック不備	最悪の場合、root権限を奪取される。	環境変数設定チェック	機能仕様上の検証不可	Security Focus 3914

the Sixth USENIX UNIX Security Symposium, 1996.

- **Ferguson:** Niels Ferguson and Bruce Schneir. A cryptographic evaluation of IPsec. Technical report, <http://citeseer.nj.nec.com/ferguson00cryptographic.html>, 2000.
- **CheckPoint:** <http://www.checkpoint.co.jp/securitycenter/advisories/2003/-cpsa-2003-04.html>
- **Nuopponen:** <http://www.hut.fi/~svaarala/publications/espiv/index.html>

## 第II部

# プロトコル検証編



## 第5章

# フォーマルメソッドによるプロトコル検証の動向

セキュアプロトコルに対するフォーマルメソッドを用いた検証には、さまざまな方法が用いられている。主な検証手法は、大まかに以下の4種類の分類することができる。

- 状態探索手法
- モデル検査法
- 信念論理推論
- 定理証明・帰納法

図 5.1 (p. 55) の左表は、これら手法の分類の特徴および欠点と、具体的な検証システムをまとめている。図に示した通りこれらの手法にはそれぞれ長所と欠点があり、どれかひとつの手法で万能という方法はない。

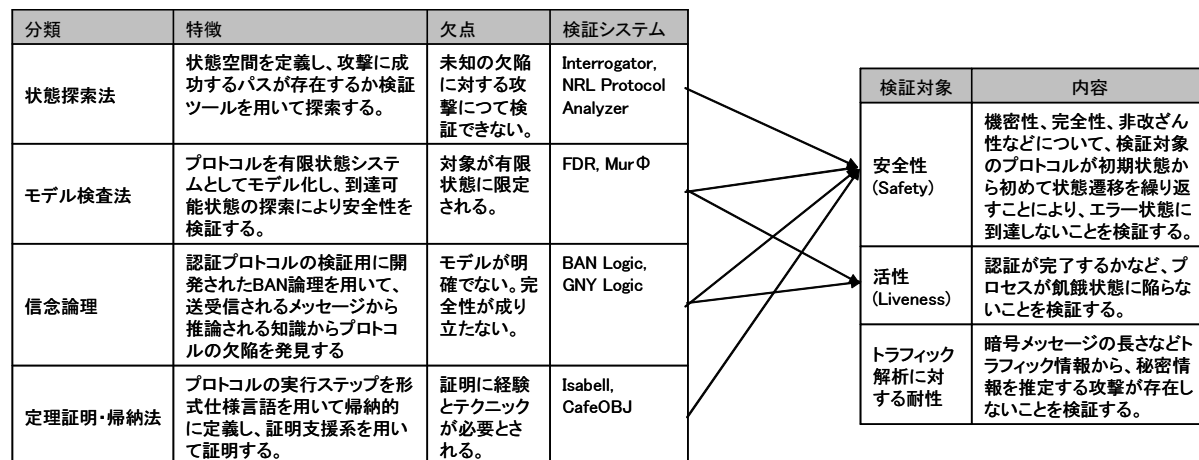


図 5.1: 主な形式手法と検証対象の関係

一方、セキュアプロトコルに対して検証すべき主な性質を形式手法の観点から分類すると図 5.1 の右表にあるとおり、以下の3種類の性質に大きく分類することができる。

- 安全性
- 活性
- トラフィック解析に対する耐性

図 5.1 の各表の間を結ぶ矢印は、各検証手法が、おもにどの性質の検証に適しているかを示したものである。フォーマルメソッドは、トラフィック解析に対する耐性など確率的な事象に対する検証に向かないなど、右表にあげた検証すべき性質に対してすべてをカバーできる訳ではないが、特定の性質に関しては、プロトコルが満たすべき性質について理論的な保証を与えることができる点で極めて有用であるとみなされている。

図 5.2 (p. 56) は、各検証システムを検証手法ごとに分類し、それらを用いてプロトコル検証を行った研究の初期の事例を年推移軸上に示したものである。

	1978	1980	1990	2000	年
状態探索			Interrogator (1984) Ina Test (1987) NRL Protocol Analyzer (1989)		
モデル検査				FDR (1996) NurΦ (1997)	
信念論理			BAN Logic (1989) GNY Logic (1990)		
定理証明・帰納法				Isabell (1997) PVS (1999) CafeOBJ (2000)	
その他手法			LOTOS (1989)	Z (1994) CSP (1995)	
形式手法に係わる動向	Needham&Schroeder (1978) 形式手法の適用可能性に言及	Dolev-Yao (1983) プロトコルモデル化		Composability (2001) Open-Ended Protocol (1997)	

図 5.2: 検証システムを用いたプロトコル検証の分類と事例変遷

形式手法に係わる最初の動きとして、形式手法を用いた検証の可能性についてはじめて言及した Needham と Schroeder の研究 [69] や形式手法によるプロトコルのモデル化をはじめて行った Dolev-Yao の研究 [26] などを挙げるができる。

状態探索法では、攻撃者を自動化して、有限状態を探索し攻撃を発見する Introgator を用いた手法 [61]、Naval Research Laboratory において行われた Dolev-Yao のモデルを完全に形式化した NRL を用いた研究 [57] などが挙げられる。

信念論理 (Belief Logic) の分野では、Burrows, Abadi, Needham が、BAN 論理を構築し、認証プロトコルの検証を行った [13]。BAN 論理では、暗号の性質に関する直感的推論を信念のルールとして形式化し、プロトコルの動作に関する性質の推論を行っている。

モデル検査法の分野では、FDR を用いた Needham-Schroeder プロトコル検証した研究 [50] が代表的である。定理証明、帰納的定義に基づく証明による方法では、Paulson が、高階論理に基づく Isabell を用いて TLS の検証を行っている [77]。

---

近年、ISAKMP-IKE に代表されるように複数のサブプロトコルを組合わせたプロトコルスイートとして全体の機能を実現するメタプロトコル（プロトコルフレームワーク）が増え、プロトコル間の相互作用に関する検証ニーズが高まっている [58]。また、プロトコルの規模が増大すると、検証規模は組合わせ的に増大する場合が多く、大きな単一のプロトコルの検証は、非常に困難となる。そのために、サブプロトコルを組合わせにより全体のプロトコルに害を与えない安全な組み上げ法 (Universal Composability) に関する研究が注目されている [16, 35, 33]。



## 第 6 章

# 前提となる脅威モデルと検証の限界

フォーマルメソッドによるプロトコル検証においては、前提となる脅威モデルと、その脅威モデルのもとで保証される性質を明確にすることが重要である。

TLS プロトコルの検証においては以下のことを前提条件とする。

- [プロトコルの独立性]

検証対象のプロトコルは、同時に進行する他のプロトコルとの干渉は無いものとする。したがって、他のプロトコルの脆弱性による秘密情報等の漏洩はない。一般に、安全なプロトコルを組合せたものは安全である (Universal Composability [59]) との保証はないため、プロトコルの非干渉性を仮定することで、この問題を回避する。

- [暗号技術の安全性]

メッセージ暗号化、署名、認証、疑似乱数生成等の暗号技術は安全であると仮定する。

- [通信者サイト内部での処理の安全性]

信頼できる通信者の内部処理において、秘密情報の漏洩等が発生せず、安全であると過程する。

- [攻撃者の能力]

プロトコルに対する攻撃としては、Dolev と Yao によって提案された一般化された攻撃者のモデル [27] に基づき、以下のような能力を仮定すれば十分であると考えられる。

- (盗聴)

攻撃者は、通信者が送受信するネットワーク上を流れるメッセージをすべて取得できる。

- (復号化)

攻撃者は、ネットワーク上を流れる暗号化されたメッセージに対して、ネットワーク上から取得できる鍵または自分が生成する鍵を用いて暗号化されたメッセージを復号化することができる。

- (改竄)

ネットワーク上を流れるメッセージを、攻撃者が取得できるメッセージをもとに任意の改竄を施し、任意の相手に送信することができる。

以上のような攻撃を任意に組み合わせることにより、正規の送信者のメッセージに必要な改竄を施し（必要な鍵など取得して改竄が可能な場合）、送信先に送付することを繰り返すことで行われる成り済み攻撃などもモデルに含まれることになる。一般的にこのような攻撃能力を任意に組み合わせにより、可能な攻撃をカバーすることが可能である。

以上の仮定によるモデル化によるフォーマルメソッドによる検証可能な限界は以下の通りである。

- (確率的攻撃)  
サーバの応答形態や応答メッセージの情報の違いを利用した確率的な推定に基づく攻撃のモデル化は不可能である。一般に、サイドチャネル攻撃、タイミング攻撃と呼ばれる攻撃は、この種の確率的な性質を用いた攻撃に含まれ、これらを検証することができない。
- (暗号技術自体の脆弱性)  
暗号技術自体のモデル化は困難であるため、暗号技術の安全性を仮定する。したがって、暗号の脆弱性に基づく攻撃は検証の対象としない。
- (通信者サイトの内部状態)  
通信者サイト内の処理をモデル化は規模的な面で困難である。
- (検証ゴールの記述力)  
モデル化に基づき検証できる性質（ゴール）は命題の形で記述することが必要である。命題の記述は、プロトコルのモデル化と形式仕様記述言語の記述力に依存する。
- (証明可能性)  
検証ゴールが記述されても、実際の証明あるいは反証が可能である保証はない。証明された場合には、その正しさは保証されるが（健全性）、証明が失敗したからといってその性質が成り立たない（完全性）ことを保証することはできない。

## 第7章

# 検証対象

### 7.1 対象プロトコル

本調査研究では、フォーマルメソッドを用いた検証の対象として、調査対象としたセキュアプロトコルの SSL/TLS, IPsec, SSH の内、TLS を選択する。プロトコル調査編で示したとおり、TLS は、IPsec, SSH に比べ、プロトコル全体のうちメッセージ交換によるセッション確立の割合がもっとも多いプロトコルである。検証の前提で示したとおり、通常、プロトコル検証では、暗号化などの技術の安全性は仮定するため、IPsec などデータのフォーマットとその暗号化、認証コード生成の比重の大きいプロトコルはあまり検証に向かない。今回、検証するのは TLS プロトコルのみであるが、他のプロトコルに関しても、本検証手法により検証可能であると考えられる。

特に、実際の検証においては、TLS プロトコルのうち、セッション確立のためのメッセージ交換が中心となる TLS ハンドシェイクプロトコルを対象とする。

### 7.2 検証すべき性質

検証対象とする TLS ハンドシェイクプロトコルに対して検証すべき性質は、この TLS ハンドシェイクプロトコルが提供する機能によって定まる。TLS ハンドシェイクプロトコルの目的は、安全な通信を確立するための暗号仕様と秘密鍵暗号のもとになる Master secret の安全な共有にある。これらを実現するために、TLS ハンドシェイクプロトコルは以下の性質を保証する [25] もであり、それ以上でもそれ以下でもない。

- 性質 1 (通信相手の認証)  
TLS プロトコルの通信者 (サーバー、クライアント) の双方が互いに相手を認証する。
- 性質 2 (秘密鍵の安全な共有)  
ネゴシエーションにより共有する秘密 (共有秘密鍵のもとになる Premaster Secret) が通信者以外に漏洩しない。
- 性質 3 (通信者間の合意内容の真正性)  
ネゴシエーションにより合意した暗号スイートが、両者の意図したものと一致している。

これらの性質は、当然のことながら、セキュリティに関するすべての性質を満たしている訳ではない。例えば、TLS によって確立されたセッションによって、実際に行われる通信においては、共通鍵暗号を用いるため、本質的に否認防止の要件を満たすことはできない。また、一般的に、可用性 (Availability)(サービスが常に提供されていること) をセキュアプロトコルで保証することも困難であると言える。これらは、TLS が保証する性質に、設計上最初から含まれていないものである。

CafeOBJ を用いた検証では、TLS が保証する性質に対して以下のような性質が満たされることを示す。

- 性質 4 (Premaster Secret の機密性)

プロトコルの実行において、Premaster Secret が通信者以外に漏洩しない。セッション確立後の暗号通信は、Master Secret を用いて、サーバ、クライアントの各サイトで、内部的に生成された共通鍵を用いる。

- 性質 5 (ServerFinish メッセージの認証)

信頼できるクライアントが、ServerFinish メッセージをサーバと思われる送信者から、プロトコルに則ったメッセージを受信した場合、ServerFinish メッセージは、確かにサーバから送信され、途中で改竄されていない。

- 性質 6 (ServerHello, Certificate メッセージの認証)

信頼できるクライアントが、ServerHello, Certificate, ServerFinish メッセージをサーバと思われる送信者から、プロトコルに則ったメッセージを受信した場合、ServerHello, Certificate メッセージは、確かにサーバから送信され、途中で改竄されていない。

図 7.1 (p.62) は、TLS が提供する機能に必要な性質全体と今回検証を行う性質の関係を示したものである。

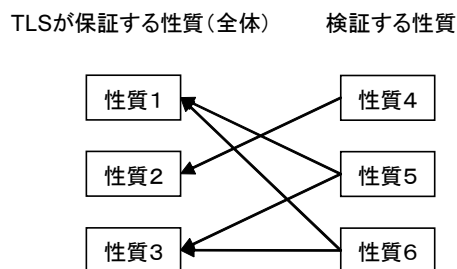


図 7.1: TLS の性質と検証性質の関係

性質 4 は、性質 2 に一致する。なぜならば、Master Secret は、Premaster Secret を用いて、サーバ、クライアントの各サイトで、内部的に生成される。検証においては、サイト内部処理の安全性を仮定しているため、Premaster Secret がサーバ、クライアント以外に漏洩しなければ、共通鍵の機密性が保証される。

性質 1 は、通信者の相手認証に関するもので、これは、ServerFinished, ServerHello, Certificate の送信者を認証する性質 5 および性質 6 の組み合わせによって保証される。

性質 3 は、合意内容の完全性に関するもので、これは、ServerFinished, ServerHello, Certificate メッセージが改竄されず確にサーバから送信されていることを保証する性質 5 および性質 6 の組合わせによって保証される。



## 第 8 章

# 代数仕様言語 CafeOBJ によるプロトコル検証

### 8.1 代数仕様言語 CafeOBJ の概要

CafeOBJ[23] は、始代数 (initial algebra) と隠蔽代数 (hidden algebra) と呼ばれるモデルによって意味を規定された形式仕様記述言語である。始代数は自然数等の抽象データ型の記述に、隠蔽代数 [24, 32] は抽象機械の記述に用いる。したがって CafeOBJ では、抽象データ型と抽象機械を統一の枠組、つまり代数で記述することが可能になる。

図 8.1 は、CafeOBJ による仕様記述のうちプロトコル検証において用いられる基本的な構文を示している。

CafeOBJ には、2 種類のソート (プログラミング言語の型に相当する概念) がある。可視ソート

```

mod! <モジュール名> {
  pr(<被入力モジュール名>)
  * [ <隠蔽ソート名> < <隠蔽ソート名> ] *
  [ <可視ソート名> . . . . . ]
  op <演算名> : -> <ソート名> .

  bop <作用演算名> : <隠蔽ソート名> <可視ソート名> . . .
                    -> <隠蔽ソート名> .

  eq <等式左辺> = <等式右辺> .
  ceq <等式左辺> = <等式右辺> if <Bool ソート項> .
}

```

図 8.1: CafeOBJ 仕様定義の一般形

(visible sort) と隠蔽ソート (hidden sort) である。前者は抽象データ型を、後者は抽象機械の状態空間を表すのに用いる。図に示した通り、可視ソートは '[' と ']' でかこって宣言し、隠蔽ソートは '\*[' と ']\*' でかこって宣言する。

抽象機械の状態空間を表す隠蔽ソートに対して 2 つの演算を定義できる。作用演算 (action operation) と観測演算 (observation operation) である。前者は抽象機械の状態を変化させるのに、後者は抽象機械の状態を観察するのに用いる。作用演算は、抽象機械の状態と 0 個以上のデータを引数にとり、抽象機械の (変化後の) 状態を返す。観測演算は、抽象機械の状態と 0 個以上のデータを引数にとり、その状態に関連する値を返す。作用演算は、基本的に、抽象機械の状態に適用する前後で各観測演算の返戻値がどのように変化するかを、等式を用いて定義する。作用演算と観測演算の宣言は 'bop' で始め、その他の演算は 'op' で始める。'bop' あるいは 'op' の後、演算子を記述し、つづいて ':' と (引数の) ソート列を書き、最後に '-' と (結果の) ソートを書く。引数のソート列および結果のソートが同じ演算を複数同時に宣言する場合は、'bop' と 'op' の代わりに、それぞれ 'bops' と 'ops' を用いる。

等式宣言は、可視ソート上のデータに関する等しさを定義する。等式は 'eq' で開始し、条件付等式の宣言は 'ceq' で開始する。'eq' の後、 '=' で連結された 2 つの項を記述し、最後に '.' を付ける。'ceq' の後、 '=' で連結された 2 つの項を記述し、つづいて 'if' および条件を表す項を書き、最後に '.' を付ける。

CafeOBJ の実装として CafeOBJ 処理系 [68] がある。CafeOBJ 処理系は、記述された等式を左から右への書き換え規則として使い、与えられた項を書き換える (簡約する) ことができる。この実行可能性により、記述したシステムの模擬実験を行ったりシステムがある性質を有することを検証したりできる。

CafeOBJ での記述の単位はモジュールである。モジュールは他のモジュールを部品としてパラメタ化することができ、要素に依存しない汎用のリスト等を記述できる。演算の属性として、結合律や交換律を宣言でき、集合や多重集合を簡便に記述できる。

## 8.2 検証法の概要

### 8.2.1 検証フロー

本分析では、図 5.1 に示した検証手法のうち定理証明・帰納法に分類される代数モデルに基づく形式仕様記述検証システム CafeOBJ を用いた検証を行う。CafeOBJ では、対象システムの状態遷移を、状態の外部から観察することでモデル化する観測遷移機械 (以下、観測遷移システムとも呼ぶ) として記述することが可能である [75]。本分析における検証手法の流れを示したものが図 8.2 (p. 67) である。

まず、プロトコル仕様書 (RFC) をもとに、セキュアプロトコルを観測遷移システムとしてモデル化し、プロトコルステップを状態の遷移規則として帰納的に定義する。実際には、検証規模の問題に対処するため、プロトコルの本質を残しつつメッセージ交換を抽象化した、プロトコルの抽象仕様に対して、モデル化を行う。証明すべき性質は、述語論理式の形式で表される (図 8.2”証明の記

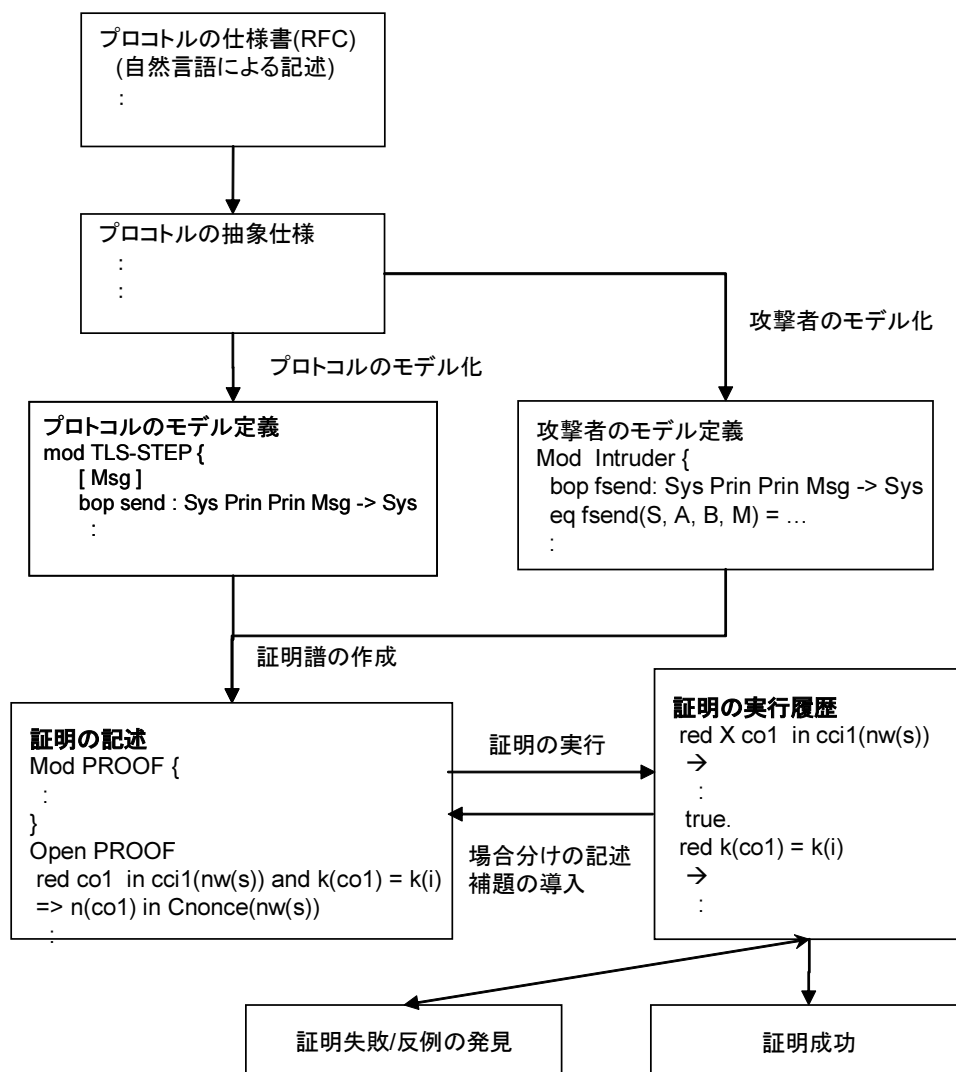


図 8.2: 代数仕様における帰納的定義に基づくプロトコル検証フロー

述”に例示)。プロトコルの各ステップにおいて、次の状態に遷移させるすべての遷移規則について、帰納法に基づき項書き換え系により証明するための記述である証明譜 (Proof Score) を作成し、その証明譜を機械的に実行 (等式推論) することによりプロトコルの安全性が保証されることを検証する。安全性の検証に失敗した場合には、項書き換えの履歴を追跡することで、プロトコルの脆弱性について分析し、反例の発見に役立てることができる。項書き換え系による検証は、高階証明支援系などによる他の手法に比べ証明過程が明瞭で、理解しやすいという特徴があるため、プロトコルの潜在的リスクの分析にも向いている。以上により、得られる形式的な記述には、図 8.2 にあるようにプロトコルの形式仕様、証明譜、証明実行履歴がある。

## 8.2.2 観測遷移機械 (観測遷移システム)

### 8.2.2.1 モデル

プロトコル検証において、対象システムの状態空間を包含する状態空間  $\Upsilon$  の存在を仮定し、状態空間の各要素を状態と呼ぶことにする。対象システムに関連する値およびそれらが状態遷移によりどのように変化するかを状態の外部から観察することでシステムのモデルを作成する。このように作成するシステムのモデルを観測遷移機械 (observational transition systems)[72, 73, 74] (または、観測遷移システム) と呼ぶ。

観測状態機械  $S$  は 3 組  $\langle \mathcal{O}, \mathcal{I}, \mathcal{T} \rangle$  で定義され、各要素は以下のとおりである：

- $\mathcal{O}$ : 観測の集合。各観測  $o \in \mathcal{O}$  は、状態空間  $\Upsilon$  から自然数等のデータ型  $D$  への関数  $o : \Upsilon \rightarrow D$  である (観測ごとに値域  $D$  は異なってもよい)。観測の返戻値を観測値と呼ぶ。  
 $S$  および 2 つの状態  $v_1, v_2 \in \Upsilon$  が与えられた場合、2 つの状態の  $S$  に関する等価性 ( $v_1 =_S v_2$ ) を以下のように定義する：

$$v_1 =_S v_2 \text{ iff } \forall o \in \mathcal{O}. o(v_1) = o(v_2),$$

ただし、 $o(v_1) = o(v_2)$  における '=' は各  $o \in \mathcal{O}$  の値域ごとに定義されているとする。

- $\mathcal{I}$ : 初期状態の集合。  $\mathcal{I} \subset \Upsilon$  である。
- $\mathcal{T}$ : 条件付遷移規則の集合。各遷移規則  $\tau \in \mathcal{T}$  は、 $=_S$  で分類される  $\Upsilon$  の商集合上の関数  $\tau : \Upsilon / =_S \rightarrow \Upsilon / =_S$  である。各  $v \in \Upsilon$  に対し、 $\tau(v)$  を同値類  $\tau([v])$  の代表元を表すものとし、 $v$  の  $\tau$  に関する事後状態と呼ぶ。  
各  $\tau \in \mathcal{T}$  に付随する条件  $c_\tau : \Upsilon \rightarrow \{\text{true}, \text{false}\}$  を効力条件と呼ぶ。  $c_\tau$  は以下の条件を満たすことを仮定する：

$$\forall v \in \Upsilon. c_\tau(v) \text{ iff } v \neq_S \tau(v).$$

$c_\tau$  が真となる状態では、 $\tau$  は効力があるという。

複数の類似した観測や遷移規則は添字を付けて表現する。つまり、観測や遷移規則は、一般に、 $o_{i_1, \dots, i_m} \in \mathcal{O}$  および  $\tau_{j_1, \dots, j_n} \in \mathcal{T}$  で表す。ただし、 $m, n \geq 0$  で、 $k \in D_k$  ( $k = i_1, \dots, i_m, j_1, \dots, j_n$ ) であるデータ型  $D_k$  の存在を仮定する。たとえば、プロセス  $p$  の配列  $a$  に対応する観測は  $a_p$  で、プロセス  $p$  が (整数型の) 配列  $a$  の  $i$  番目の要素を 1 増加することに対応する遷移規則は  $inc\text{-}a_{p,i}$  で表すことができる。

$S$  の実行は、初期状態からはじまり、次のとおりに永続的に遷移規則を適用することにより得られる状態の無限列である；実行の各段階で、遷移規則の 1 つが非決定的に選択され適用される (事後状態が決定される)。ただし、この非決定的選択は次の公平性 (fairness) を満たすものとする；すべての遷移規則は際限なく選択される。 $S$  が与えられると、このような状態の無限列の集合 ( $S$  の実行) が得られる。より正確には、 $S$  の実行は、以下の条件を満たす状態の無限列  $v_0, v_1, \dots$  である：

- 開始性： $v_0 \in \mathcal{I}$  である。
- 連続性：各  $i \in \{0, 1, \dots\}$  に対し、 $v_{i+1} =_S \tau(v_i)$  を満たす  $\tau \in \mathcal{T}$  が存在する。
- 公平性：各  $\tau \in \mathcal{T}$  に対し、 $v_{i+1} =_S \tau(v_i)$  を満たす  $i \in \{0, 1, \dots\}$  が無限に存在する。

状態が  $S$  の実行に現れるとき、その状態は  $S$  に関して到達可能であるという。

### 8.2.2.2 検証法

観測遷移機械  $S$  にとって重要な性質は、安全性 (safety property) と活性 (liveness property) に分類できる。本検証で議論する性質は安全性のみであるので、安全性のみについて記述する。 $S$  がある安全性を有するとは、 $S$  のあらゆる実行において、その安全性を侵す危険な状態に陥らないことである。より正確には次のとおりである。述語  $p : \Upsilon \rightarrow \{\text{true}, \text{false}\}$  が  $S$  に関する安全性である、あるいは  $S$  が安全性  $p$  を有するとは、 $S$  の任意の到達可能状態  $v \in \Upsilon$  において、 $p(v)$  が真である、ことである。

$S$  が安全性  $p$  を有することを検証する場合、主に以下の遷移規則の適用回数に関する帰納法を用いる：

- 基底：任意の初期状態  $v \in \mathcal{I}$  において、 $p(v)$  が成り立つことを確認する。
- 帰納段階： $p(v)$  が成り立つ任意の到達可能状態において、すべての遷移規則  $\tau \in \mathcal{T}$  に対し、 $p(\tau(v))$  が成り立つことを確認する。

## 8.3 プロトコル検証の構成と検証におけるリスク要因

プロトコル検証を構成する要素の関係と潜在するリスク要因を分析したものが図 8.3 (p. 70) である。

プロトコルおよび攻撃者は、いくつかの検証の前提のもとで、振舞い等式論理と呼ばれる論理に基づく代数仕様記述言語 CafeOBJ を用いて、観測遷移システム (8.2.2 章) としてモデル化される。検証すべき性質は論理式として形式的に記述される (検証ゴールとよぶ)。検証ゴールは、プロトコルおよび攻撃者のモデル、モデル化においてネットワークの状態に関する性質を記述する述語定義などに関する推論規則を用いて、等式推論および構造帰納法と呼ばれる証明体系を用いて証明される。

このような検証の構成において、検証結果に対するリスク要因として、図 8.3 (p. 70) にも示した通り以下のようなものにまとめられる。

1. 前提の妥当性 (前提となる暗号技術の安全性などが現実的であるか。)
2. プロトコル定義の完全性 (プロトコル仕様が正しく定義されているか。)
3. 攻撃モデルの妥当性 (攻撃者の能力の定義が妥当か。)
4. 検証ゴール完全性 (検証すべき性質が、論理式として正しく定義されているか。)
5. 証明の完全性 (証明に不備がないか。)

これらのリスク要因をひとつひとつ評価することにより検証結果に対する信頼性を保証すること

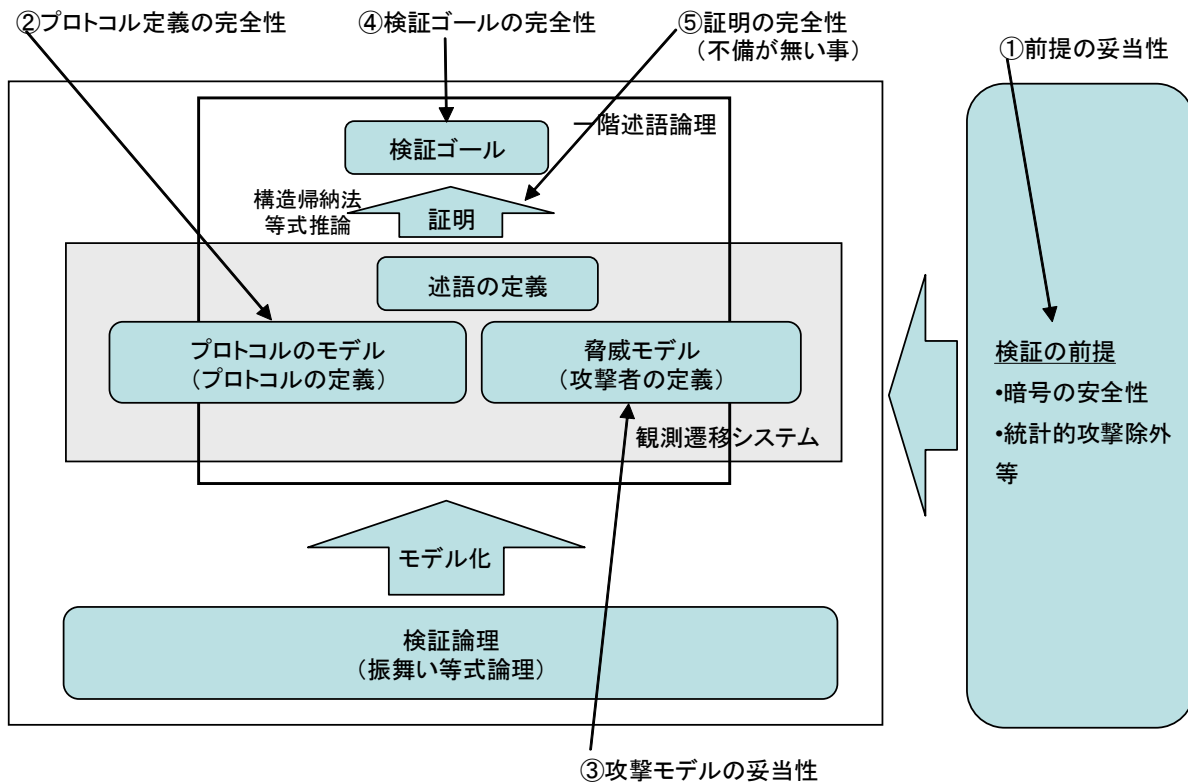


図 8.3: プロトコル検証の構成と検証におけるリスク要因

ができる。

## 8.4 TLS ハンドシェイクプロトコルのモデル化と検証

### 8.4.1 プロトコルの抽象化

検証において不要な複雑化を避けるために、TLS プロトコルが実現する安全性に関わる本質を残しつつ抽象化する。

図 8.4 (p. 71) は、検証対象とする抽象化したプロトコルを示している。A はクライアント、B はサーバを示す。最初の 6 つのメッセージ交換は、フルネゴシエーションのためのメッセージ交換を示し、残りの 4 つのメッセージは、既に確立したセッションの再開のためのメッセージ交換を示している。

検証を現実的にするために上記のように行ったプロトコルの抽象化の考え方は以下の通りである。

- オプションのサーバー証明書送信は、常に行う。
- オプションの ServerKeyExchange, CertificateRequest は、両方実行しない。
- ServerHelloDone メッセージは、Certificate メッセージとともに実行したものと見なし、個別に実行しない。
- クライアントは、オプションの CertificateVerify メッセージは送信しない。

ClientHello	$A \rightarrow B$	: $Rnad_A, ListOfChoices$
ServerHello	$B \rightarrow A$	: $Rand_B, SID, Choice$
Certificate	$B \rightarrow A$	: $Cert_B$
ClientKeyExchange	$A \rightarrow B$	: $\mathcal{E}_{K_B}(PMS)$
ClientFinish	$A \rightarrow B$	: $\mathcal{E}_{ClientKey}(ClientFinish)$
ServerFinish	$B \rightarrow A$	: $\mathcal{E}_{ServerKey}(ServerFinish)$
ClientHello2	$A \rightarrow B$	: $Rnad_A, SID$
ServerHello2	$B \rightarrow A$	: $Rnad_B, SID, Choice$
ServerFinish2	$B \rightarrow A$	: $\mathcal{E}_{ServerKey}(ServerFinish2)$
ClientFinish2	$A \rightarrow B$	: $\mathcal{E}_{ClientKey}(ClientFinish2)$

図 8.4: 検証対象とする TLS ハンドシェイクプロトコル

- ChangeCipherSpec メッセージは暗示的に実行されるものとする。
- 証明書認証局 (CA) は、唯一信頼できるもののみ存在するものと仮定する。
- Pre-master secret の交換のための暗号化には、RSA のみ利用できる。
- Finished メッセージは、それまでに交換されたハンドシェイクメッセージのハッシュ値ではなく、Client.Random, Server.Random, premaster-secret 等のハッシュ値とする。
- Finished メッセージのハッシュ値はクリアテキストで送信されるものとする。

これらの抽象化によりプロトコルの本質が保たれることが確認できる。ただし、本検証においては、検証対象とするプロトコルは、サーバ認証のみを行うモードでかつ Premaster Secret の交換には、RSA 暗号の利用に限定したモデルとしている。

### 8.4.2 モデルの構成

プロトコルは、メッセージ交換フローに関する観測遷移システムとして定義する。正規の通信者が実行するメッセージ交換および攻撃者が可能なメッセージの偽造送信等を、状態遷移規則として定義する。

このようにして定義した観測遷移システムにおいて、プロトコルのメッセージ交換および攻撃者によるメッセージ偽造のあらゆる遷移系列は、図 8.5 (p. 72) のよう理解することができる。

つまり、状態遷移空間全体について、初期状態  $init$  が存在し、任意の状態遷移演算を適用した結果、次々と新しい状態に遷移するものである。

検証したい性質は、この状態遷移図において、初期状態から開始して到達可能なすべての状態において論理式により定義された命題が満たされること示す。そのためには、初期状態で成立すること、および任意の状態で性質が成り立つと仮定した場合に、すべての状態遷移規則適用の結果もその性質が維持されることを示す必要がある。

図 8.6 (p. 73) は、CafeOBJ による TLS のモデルおよび検証推論のための定義を形式的に記述す

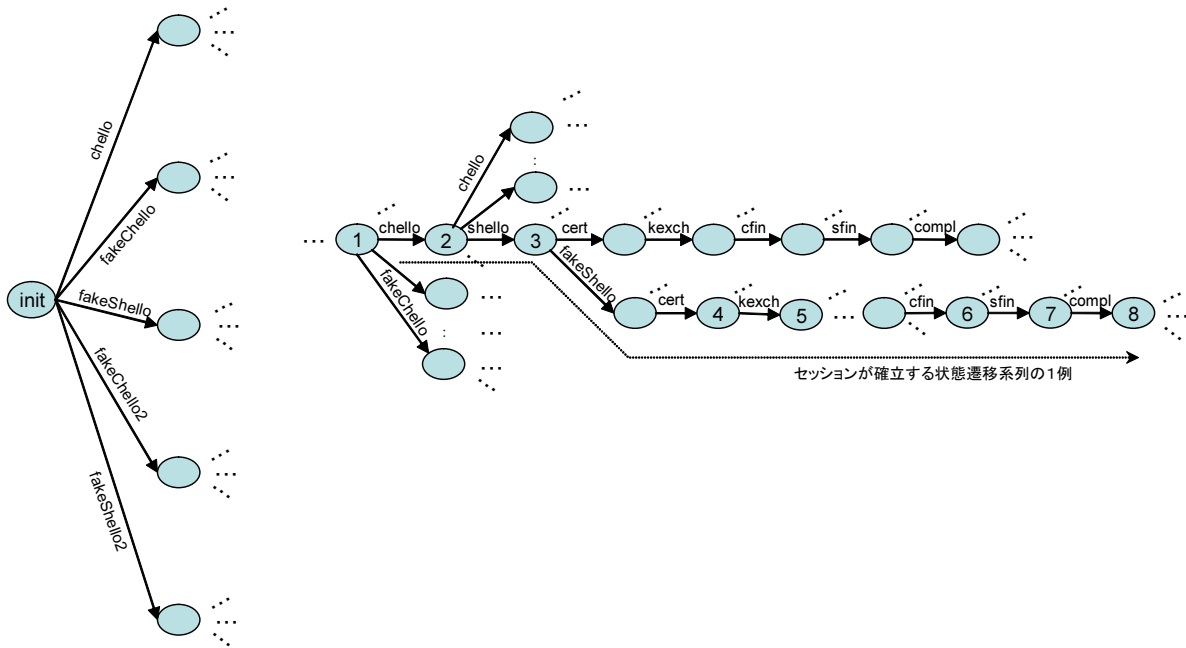


図 8.5: 観測遷移システムにおける状態遷移の様子

るためのモジュール構成を示している。

図中のモジュール TLS は、プロトコルのモデルを記述し、NETWORK は、ネットワークの状態等の定義を記述している。もっとも基礎的なモジュールには、通信参加者を定義する PRINCIPAL、公開鍵を定義する PUBKEY、セッション ID を定義する SID などがある。これらの基礎的なモジュールを、上位概念を示すモジュールから読み込み、プロトコルの定義で利用する。

### 8.4.3 メッセージのモデル化

プロトコルのメッセージ交換において送受信されるメッセージ自体を定義するためには、図 8.4(p. 71) に示した TLS のメッセージ交換に対応した 10 種類のメッセージを示す 10 種類のオペレータを宣言する (図 8.7 (p. 74))。

メッセージデータはこれらの演算を用いて構成される。ソート Msg は、メッセージを示す可視ソートである。Msg を構成するための引き数はそれぞれ以下のデータを意味する。

- Prin: 通信参加者 (攻撃者を含む)。引数の 3 つの Prin はそれぞれ、真の送信者を示すメタ情報で、見掛けの送信者 (パケットのヘッダにかかれるもの)、受信者を表す。真の送信者を表す Prin は、プロトコル外部の観測者と、真の送信者のみアクセスできるもので、攻撃者が改竄することはできない。
- Rand: 乱数
- ListOfChoice: 暗号スイートのリスト
- Sid: セッション ID

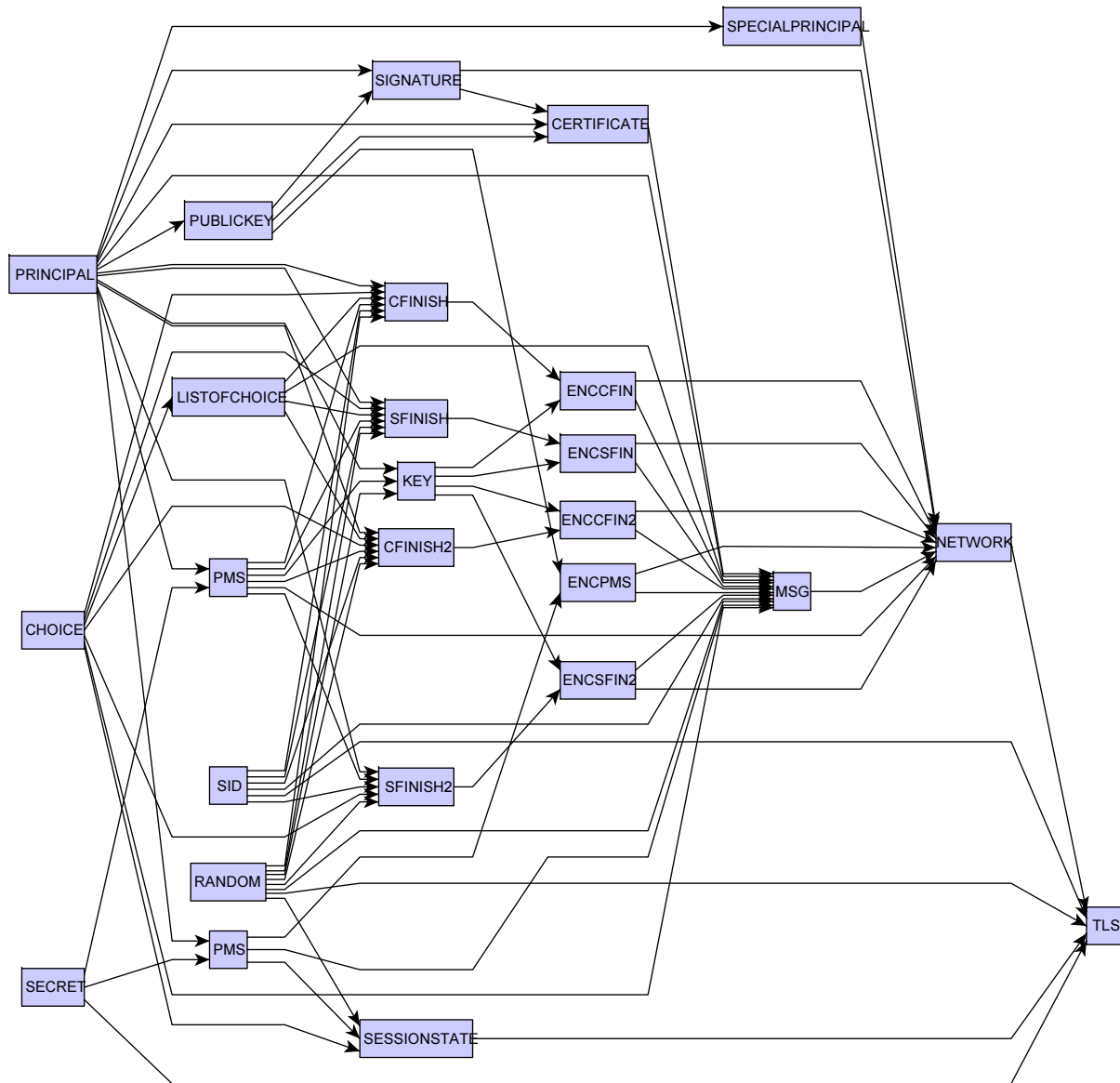


図 8.6: TLS 検証モデルのモジュール構成

- Choice: 暗号スイート
- Cert: 証明書
- EncPms: PMS(Premaster Secret) を暗号化したもの
- EncCFin: ClientFinish メッセージを暗号化したもの
- EncSFin: ServerFinish メッセージを暗号化したもの
- EncCFin2: セッション再開における ClientFinish メッセージを暗号化したもの
- EncSFin2:セッション再開における ServerFinish メッセージを暗号化したもの

これらのデータの性質は、すべてモジュールにおいて形式的に定義される。

op ch	:	Prin Prin Prin Rand ListOfChoice	-> Msg
op sh	:	Prin Prin Prin Rand Sid Choice	-> Msg
op ct	:	Prin Prin Prin Cert	-> Msg
op kx	:	Prin Prin Prin EncPms	-> Msg
op cf	:	Prin Prin Prin EncCFin	-> Msg
op sf	:	Prin Prin Prin EncSFin	-> Msg
op ch2	:	Prin Prin Prin Rand Sid	-> Msg
op sh2	:	Prin Prin Prin Rand Sid Choice	-> Msg
op cf2	:	Prin Prin Prin EncCFin2	-> Msg
op sf2	:	Prin Prin Prin EncSFin2	-> Msg

図 8.7: メッセージデータ構成のための演算子の宣言

#### 8.4.4 ネットワークのモデル化

ネットワークは、送受信によってネットワーク上を流れたメッセージの多重集合としてモデル化する。これは、攻撃者が利用できるメッセージの全体および、正規の送信者のメッセージ送信履歴として利用される。

観測遷移システムとしては、ネットワークは、プロトコルの状態を観測した結果得られるそれまでにネットワーク上を流れたメッセージの多重集合を示している。

攻撃者は、ネットワークから Pre-master secret(PMS), デジタル署名、および 5 種類の暗号文を収集することができる。

TLS プロトコルの乱数 ClientRandom, ServerRandom, およびセッション ID, 暗号スイート、暗号スイートのリスト、公開鍵は、クリアテキストで送信されるため、攻撃者は簡単に取得できるものとする。共通鍵 (ClientKey, ServerKey) は、メッセージ交換に含まれないため、pre-master secret が漏洩しないかぎり、攻撃者に取得されることはない。

ネットワークは、可視ソート Network として宣言される。ネットワークから収集され攻撃者が利用可能な情報の集合は図 8.8 (p. 75) のオペレータにより定義される。

#### 8.4.5 プロトコルのモデル化

プロトコルの状態遷移は、プロトコルの外部から観測可能な値によって定義される。観測可能な値とは、第 8.4.4 節で示したネットワーク、2 人の通信者のセッション状態、使用された乱数の集合、使用されたセッション ID の集合、そして、使用された秘密の集合である。これらは図 8.9 (p. 75) の観測オペレータにより定義される。

Protocol は、状態空間を表す隠蔽ソート。URand, USid, USecret は、乱数集合、セッション ID

op cpms	:	Network	-> ColPms	Premaster Secret(PMS) の集合
op csig	:	Network	-> ColSig	署名の集合
op cepms	:	Network	-> ColEncPms	暗号化された PMS の集合
op cecfin	:	Network	-> ColEncCFin	暗号化された ClientFinish の集合
op cesfin	:	Network	-> ColEncSFin	暗号化された ServerFinish の集合
op cecfin2	:	Network	-> ColEncCFin2	暗号化されたセッション再開 ClientFinish の集合
op cesfin2	:	Network	-> ColEncSFin2	暗号化されたセッション再開 ServerFinish の集合

図 8.8: 攻撃者が利用可能なデータを構成する演算子の宣言

bop nw	:	Protocol	-> Network
bop ss	:	Protocol Prin Prin Sid	-> Session
bop ur	:	Protocol	-> URand
bop ui	:	Protocol	-> USid
bop us	:	Protocol	-> USecret

図 8.9: プロトコルに対する観測を行う演算子

の集合、秘密の集合を表す隠蔽ソートである。

状態遷移は、メッセージ交換の各ステップに対応する状態遷移オペレータによって定義される。図 8.4 の 10 種類のメッセージ交換に対応して 図 8.10 (p. 76) の通り状態遷移オペレータにより定義される。

Secret は、Premaster Secret(PMS) がグローバルに一意にするための秘密情報である。その他のソートはすでに説明済みである。

状態遷移は、これらの状態遷移オペレータをプロトコルの状態を表す隠蔽ソート P に適用し、その結果に対して観測オペレータ (図 8.9) による観測結果を定義することにより規定される。

例えば、サーバからクライアントに証明書を送信するメッセージ交換を表す kexch は、図 8.11 (p. 76) のように定義される。

定義中の P は、ソート Protocol の変数、B, A2 はソート Princial の変数、M1, M2 は、ソート Message の変数を示す。

第 1, 2 行目は、状態遷移が生じる作用条件を定義している。第 2 行目の定義は、ネットワーク上に、ClientHello, ServerHello メッセージに則った M1, M2 が存在し、M1 の送信先は B、M2 の署名者は B、M2 の送信元は B、M1 の送信元は、M2 の送信先に一致し、M2 に含まれる選択された暗号スイートは、M1 の暗号スイートリストに含まれることを意味している。

第 3 の条件式等式は、作用条件が成り立つならば、サーバ証明書の送信によるプロトコルの状態

bop chello	:	Protocol Prin Prin Rand ListOfChoice	-> Protocol
bop shello	:	Protocol Prin Prin Rand Sid Choice Msg	-> Protocol
bop cert	:	Protocol Prin Msg Msg	-> Protocol
bop kexch	:	Protocol Prin Secret Msg Msg Msg	-> Protocol
bop cfin	:	Protocol Prin Secret Msg Msg Msg Msg	-> Protocol
bop sfin	:	Protocol Prin Msg Msg Msg Msg	-> Protocol
bop compl	:	Protocol Prin Secret Msg Msg Msg Msg Msg	-> Protocol
bop chello2	:	Protocol Prin Prin Rand ListOfChoice	-> Protocol
bop shello2	:	Protocol Prin Prin Rand Msg	-> Protocol
bop sfin2	:	Protocol Prin Msg Msg	-> Protocol
bop cfin2	:	Protocol Prin Secret Msg Msg Msg	-> Protocol
bop compl2	:	Protocol Prin Msg Msg Msg Msg	-> Protocol

図 8.10: プロトコル状態遷移を表す状態遷移オペレータ

op	c-cert	:	Protocol Prin Msg Msg	->	Bool
eq	c-cert(P,B,M1,M2)	=	(M1 \in nw(P) and M2 \in nw(P) and ch?(M1) and sh?(M2) and dst(M1) = B and crt(M2) = B and src(M2) = B and src(M1) = dst(M2) and choice(M2) \in list(M1)) .		
ceq	nw(cert(P,B,M1,M2))	=	ct(B,B,dst(M2),cert(B,k(B),sig(ca,B,k(B)))) , nw(P) if c-cert(P,B,M1,M2) .		
eq	ss(cert(P,B,M1,M2),A2,B2,I2)	=	ss(P,A2,B2,I2) .		
eq	ur(cert(P,B,M1,M2))	=	ur(P) .		
eq	ui(cert(P,B,M1,M2))	=	ui(P) .		
eq	us(cert(P,B,M1,M2))	=	us(P) .		
ceq	cert(P,B,M1,M2)	=	P if not c-cert(P,B,M1,M2) .		

図 8.11: ServerKeyExchange メッセージに対応する状態遷移演算 kexch の定義

遷移が発生し、遷移後のネットワーク状態を観測すると、B からの証明書が追加されていることを示している。

第 4, 5, 6, 7 の等式は、それぞれ、証明書の送信によりセッション状態が変化しないこと、使用された乱数の集合が変化しないこと、使用されたセッション ID の集合が変化しないこと、使用された Premaster Secret の集合が変化しないことを意味している。第 8 の条件付き等式は、作用条件が満たされない時は状態遷移が発生しないことを示す。

図 8.10 (p. 76) における他の状態遷移の定義も同様に、観測演算を用いて定義される。

### 8.4.6 攻撃者のモデル化

攻撃者のモデルは、ネットワーク上から収集できる情報の定義と、それらを用いて、任意のメッセージを生成あるいは、正規のメッセージの改竄し、任意の通信者に送信できることを意味する定義から構成される。実際には、正規のメッセージ送信による状態遷移規則に対して、任意のメッセージパターンを生成するための状態遷移規則 15 個を用いて定義される。

具体例として、ClientKeyExchange メッセージを偽造する 2 種類の状態遷移規則を図 8.12 (p. 77) に示す。

```

bop fakeKexch1 : Protocol Prin Prin EncPms -> Protocol
bop fakeKexch2 : Protocol Prin Prin PubKey Pms -> Protocol
-- fakeKexch1
op c-fakeKexch1 : Protocol Prin Prin EncPms -> Bool
eq c-fakeKexch1(P,A,B,EPMS) = EPMS \in cepms(nw(P)) .
--
ceq nw(fakeKexch1(P,A,B,EPMS)) = kx(intruder,A,B,EPMS) , nw(P)
    if c-fakeKexch1(P,A,B,EPMS) .
eq ss(fakeKexch1(P,A,B,EPMS),A2,B2,I2) = ss(P,A2,B2,I2) .
eq ur(fakeKexch1(P,A,B,EPMS)) = ur(P) .
eq ui(fakeKexch1(P,A,B,EPMS)) = ui(P) .
eq us(fakeKexch1(P,A,B,EPMS)) = us(P) .
ceq fakeKexch1(P,A,B,EPMS) = P if not c-fakeKexch1(P,A,B,EPMS) .
    fakeKexch2
op c-fakeKexch2 : Protocol Prin Prin PubKey Pms -> Bool
eq c-fakeKexch2(P,A,B,PK,PMS) = PMS \in cpms(nw(P)) .
--
ceq nw(fakeKexch2(P,A,B,PK,PMS)) = kx(intruder,A,B,epms(PK,PMS))
    , nw(P) if c-fakeKexch2(P,A,B,PK,PMS) .
eq ss(fakeKexch2(P,A,B,PK,PMS),A2,B2,I2) = ss(P,A2,B2,I2) .
eq ur(fakeKexch2(P,A,B,PK,PMS)) = ur(P) .
eq ui(fakeKexch2(P,A,B,PK,PMS)) = ui(P) .
eq us(fakeKexch2(P,A,B,PK,PMS)) = us(P) .
ceq fakeKexch2(P,A,B,PK,PMS) = P if not c-fakeKexch2(P,A,B,PK,PMS) .

```

図 8.12: ClientKeyExchange メッセージ偽造に対応する状態遷移規則

P は、ソート Protocol の変数、A、B はソート Princial の変数、PMS は、Premaster Secret の変数、EPMS は、暗号化した PMS を示す変数、intruder は攻撃者の定数をしめす。

fakeKexch1 は、ネットワーク上に流れた暗号化された Premaster Secret を利用して、サーバに ClientKeyExchange メッセージを送信する状態遷移規則を定義する。fakeKexch2 は、ネットワーク上に流れた平文の Premaster Secret を利用して、任意の鍵を用いて暗号化した ClientKeyExchange メッセージをサーバに送信する状態遷移規則を定義する。

本検証における攻撃者のもでるは、第 6 章に示した前提のもとで、既知の攻撃および未知の攻撃などあらゆる攻撃を包含する一般性のある攻撃をモデル化している。

#### 8.4.7 検証ゴール

検証すべき性質は、第 7.2 章で示した。本章では、実際に検証する性質を示し、それらが、検証すべき性質に対応していることを示す。

検証する性質を表現すると以下ようになる。

<p>性質 7 (機密性)</p> <p>Premaster Secret が正規の通信者 (サーバ、クライアント) 以外に漏洩しない。</p>
<p>性質 8 (ServerFinished メッセージの認証)</p> <p>正規のクライアントがプロトコルに従った ServerFinished メッセージを、サーバと思われる者から受信したならば、そのメッセージは、確にサーバから送信されたものである。</p> <p>性質 9 (セッション再開 ServerFinised メッセージの認証)</p> <p>正規のクライアントがプロトコルに従った ServerFinished2 メッセージを、サーバと思われる者から受信したならば、そのメッセージは、確にサーバから送信されたものである。</p>
<p>性質 10 (ServerHello,Certificate メッセージの認証)</p> <p>正規のクライアントがプロトコルに従った ServerHello, Certificate, ServerFinished メッセージをサーバから受信したと思われるならば、ServeHello, Certificate メッセージは、確に正規のサーバから送信されたものである。</p> <p>性質 11 (セッション再開 ServerHello メッセージの認証)</p> <p>正規のクライアントがプロトコルに従った ServerHello2, ServerFinished2 メッセージをサーバから受信したと思われるならば、ServeHello2 メッセージは、確に正規のサーバから送信されたものである。</p>

すべてのセキュリティパラメータは、Premaster Secret をもとに生成されるため、Premaster Secret は、プロトコルにおいてもっとも機密性を必要とされる情報である。

上記の性質 8 は、クライアントが対応するサーバから TLS ハンドシェイクプロトコルに基づき暗号スイートとセキュリティパラメータを取得した場合、サーバは確にその内容に合意していることを保証する。

性質 9 は、クライアントが対応するサーバからセッション再開プロトコルにより、暗号スイートとセキュリティパラメータを取得した場合、サーバは確にその内容に合意していることを意味する。

性質 10 は、クライアントが対応するサーバから TLS ハンドシェイクプロトコルに基づき暗号ス

イトとセキュリティパラメータを取得した場合、暗号スイートといくつかのセキュリティパラメータは、サーバにより提示されたものである。

性質 11 は、クライアントが対応するサーバからセッション再開プロトコルにより、暗号スイートとセキュリティパラメータを取得した場合、暗号スイートといくつかのセキュリティパラメータは、サーバにより提示されたものである。

これらの性質は、CafeOBJ によるモデル化において以下のように表現される。

性質 7 を CafeOBJ で形式的に記述したものが性質 12 である。

性質 12 (機密性)

```
eq inv1000(P,PMS) =
  (PMS \in cpms(nw(P)) implies
   (client(PMS) = intruder or server(PMS) = intruder)) .
```

既を示した説明したとおり、 $cpms(N)$  は、ネットワーク上を流れたメッセージの集合  $N$  に対して、攻撃者が利用可能な PMS の集合を表す。 $nw(P)$  は、プロトコル状態  $P$  に対して、それまでにネットワーク上を流れたメッセージの集合を表す。また、 $client(PMS)$  は、PMS を作成した者、 $server(PMS)$  は、PMS の送付先を求める演算である。また、 $PMS \in CPMS$  は、PMS が、集合  $CPMS$  に含まれることを示す述語である。この時、性質 12 は、任意のプロトコル状態  $P$  と PMS について、PMS が攻撃者が利用可能な場合、その PMS は攻撃者が、作成したものか、攻撃者に送信されたものであることを言明している。これを数学的な記法で表現すると以下ようになる。

$$\begin{aligned} \forall P : Protocol, pms : PMS. (psm \in f_{cpms}(f_{nw}(P)) \\ \Rightarrow client(pms) = I \wedge server(pms) = I) \end{aligned} \quad (8.1)$$

ただし、 $P$  はプロトコル状態空間 Protocol、 $pms$  は、Premaster Secret 集合  $PMS$  空間の全称変数を示し、 $f_{nw}(P)$  は、プロトコル状態  $P$  からネットワーク上に流れたメッセージの (多重) 集合を返す関数、 $f_{cpms}(S)$  は、メッセージ集合  $S$  から攻撃者が利用可能な PMS の (多重) 集合を返す関数である。 $I$  は攻撃者を示す。

同様に、性質 8 ~ 性質 11 に対応する CafeOBJ による記述は、性質 13 ~ 性質 16 である。

性質 13 (ServerFinished メッセージの認証)

```
eq inv1020(P,A,B,B1,R1,R2,L,C,I,S) =
  (not(A = intruder) and
   sf(B1,B,A,esfin(k(B,pms(A,B,S),R1,R2),sfin(A,B,I,L,C,R1,R2,
   pms(A,B,S)))) \in nw(P)
   implies
   sf(B,B,A,esfin(k(B,pms(A,B,S),R1,R2),sfin(A,B,I,L,C,R1,R2,
   pms(A,B,S)))) \in nw(P)) .
```

## 性質 14 (セッション再開 ServerFinised メッセージの認証)

```

eq inv1150(P,A,B,B1,B2,B3,R1,R2,L,C,I,S,K)
  = (not(A = intruder) and
     sh(B1,B,A,R2,I,C) \in nw(P) and
     ct(B2,B,A,cert(B,K,sig(ca,B,K))) \in nw(P) and
     sf(B3,B,A,esfin(k(B,pms(A,B,S),R1,R2),
                    sfin(A,B,I,L,C,R1,R2,pms(A,B,S)))) \in nw(P)
     implies
     sh(B,B,A,R2,I,C) \in nw(P) and
     ct(B,B,A,cert(B,K,sig(ca,B,K))) \in nw(P)) .

```

## 性質 15 (ServerHello,Certificate メッセージの認証)

```

eq inv1040(P,A,B,B1,R1,R2,C,I,S)
  = (not(A = intruder) and
     sf2(B1,B,A,esfin2(k(B,pms(A,B,S),R1,R2),
                      sfin2(A,B,I,C,R1,R2,pms(A,B,S)))) \in nw(P)
     implies
     sf2(B,B,A,esfin2(k(B,pms(A,B,S),R1,R2),
                      sfin2(A,B,I,C,R1,R2,pms(A,B,S)))) \in nw(P)) .

```

## 性質 16 (セッション再開 ServerHello メッセージの認証)

```

eq inv1160(P,A,B,B1,B2,R1,R2,C,I,S)
  = (not(A = intruder) and
     sh2(B1,B,A,R2,I,C) \in nw(P) and
     sf2(B2,B,A,esfin2(k(B,pms(A,B,S),R1,R2),
                      sfin2(A,B,I,C,R1,R2,pms(A,B,S)))) \in nw(P)
     implies
     sh2(B,B,A,R2,I,C) \in nw(P)) .

```

これまでに挙げた検証性質の関係を示したものが図 8.13 (p. 81) である。

性質 1 から性質 6 は、図 7.1 (p. 62) で示したとおりである。性質 7 から性質 11 は、性質 4 から性質 6 を実際に検証するにあたって、性質を分割したものである。性質 13 から性質 16 は、性質 7 から性質 11 を、CafeOBJ における一階述語論理式の形式で表現したものである。

## 8.4.8 証明譜

CafeOBJ による検証は、項書換えにより Bool ソートのオペレータにより定義された述語が true に簡約されることを確認することによって行われる。観測状態遷移システムにおいては、述語を用いて表現した隠蔽ソートを引数とする論理式が初期状態から到達可能なすべての状態において成り

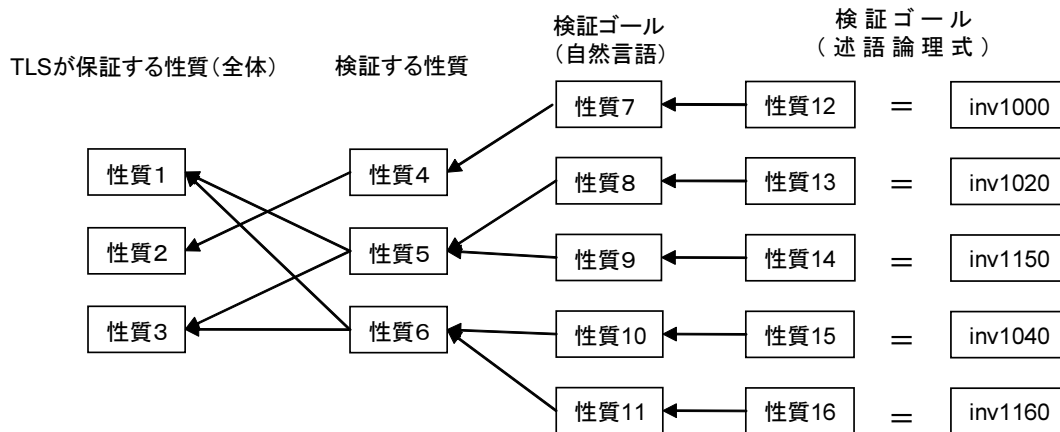


図 8.13: 検証性質に関する関係

立つことを示す。状態遷移は、init から開始し、構造的帰納法によりすべての場合について性質をしめす。そのためには、定義したすべての状態遷移規則に関して、状態遷移前に成り立つならば、状態遷移後も性質が成り立つことを証明する。このような証明のための記述を証明譜 (Proof Score) と呼ぶ。

目的とする性質を帰納法で証明するためには、それに必要な補題をあらかじめ証明する必要がある場合がある。本検証では、図 8.14 (p. 82) に定理を順に証明しながら、それらを補題として用いて、目的とする性質を証明する。

#### 8.4.8.1 述語の定義

証明すべき性質は、Bool ソートとして定義するオペレータによる述語と、論理演算子 (and, or, not, implies) を用いた論理式として表現する。論理演算子は、CafeOBJ 組込みの演算子を用いる。検証する論理式は、通信者 (攻撃者を含む) 証明書、乱数、Premaster Secret、メッセージなどの一貫性を示す等号とネットワーク上に過去にあるメッセージが送信されたか否かを示す述語を用いる。

図 8.15 (p. 82) は、証明書の等しさを示す述語の定義である。

証明書は、cert オペレータにより、通信者を表す Prin, 公開鍵を表す PubKey, 署名を表す Sig を引数として構成される。Cert ソートを 2 つ引数にとる Bool ソートオペレータ  $\_=\_$  は、等式を 2 つ用いて述語を定義する。最初の等式は、構文的に等しい証明書は等しいことを定義している。第 2 の等式は、cert オペレータで構成された証明書の等しさは、cert の引数の等しさに帰着させることにより定義されることを意味している。cert の引数は、通信者、公開鍵、署名の 3 つの要素から構成され、証明書の等しさは、これらの要素がすべて等しいことで定義される。

図 8.16 (p. 82) は、暗号化された ClientFinish メッセージが、過去にネットワークに送信されたかどうかを定義する述語である。

第 1 の等式は、ネットワークの初期状態を示す void には、暗号化された ClientFinish メッセージが含まれないことを定義している。第 2 の条件付き等式は、ネットワーク上のメッセージの最初

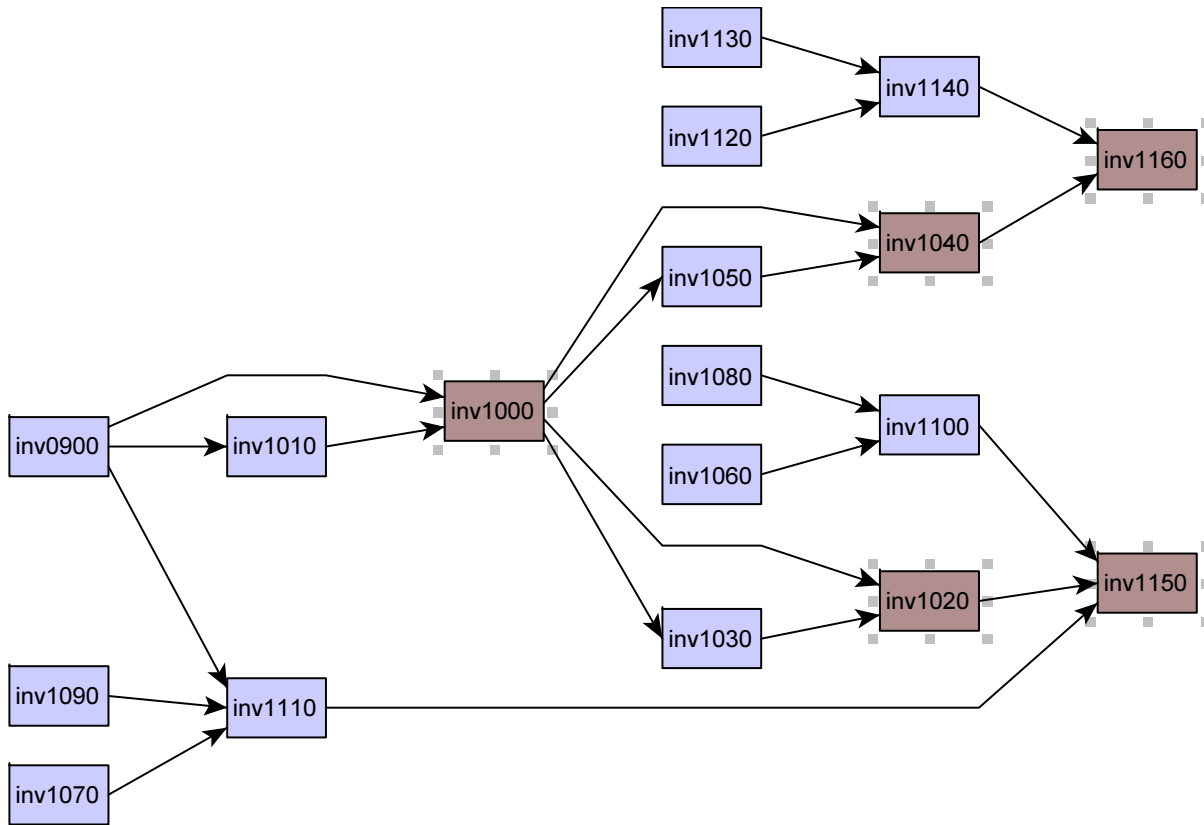


図 8.14: 証明した性質 (定理) の依存関係

```

op cert : Prin PubKey Sig -> Cert
op _=_ : Cert Cert -> Bool {comm}
eq (C = C) = true .
eq (cert(A1,PK1,G1) = cert(A2,PK2,G2))
  = (A1 = A2 and PK1 = PK2 and G1 = G2) .

```

図 8.15: 証明書の等しさを示す述語の定義

```

eq ECFIN \in cecfin(void) = false .
ceq ECFIN \in cecfin(M,NW) = true if (cf?(M) and ECFIN = ecfin(M)) .
ceq ECFIN \in cecfin(M,NW) = ECFIN \in cecfin(NW)
  if not(cf?(M) and ECFIN = ecfin(M)) .

```

図 8.16: 暗号化された ClientFinish メッセージが、過去にネットワークに送信されたか判定する述語の定義

の要素が ClientFinished メッセージの送信を意味する

図 8.17 (p. 83) は、過去にネットワーク上に流れたメッセージから、攻撃者が利用可能な Premaster secret の全集合に、ある Premaster secret が含まれるかを判定する述語の定義である。

第 1 の等式定義は、PMS の作成者が攻撃者なら、その PMS を攻撃者が利用できることを表す。

```

eq PMS \in cpms(void) = (client(PMS) = intruder) .
ceq PMS \in cpms(M,NW) = true if (kx?(M) and owner(k(epms(M)))
                                = intruder and
                                PMS = pms(epms(M))) .
ceq PMS \in cpms(M,NW) = PMS \in cpms(NW)
  if not(kx?(M) and owner(k(epms(M))) = intruder and
        PMS = pms(epms(M))) .

```

図 8.17: Premaster secret の存在を判定する述語の定義

第 2 の条件付き等式は、ネットワーク上を流れたメッセージ列の最初のメッセージ中に攻撃者の公開鍵で暗号化された PMS が含まれる場合、攻撃者はそれを利用できることを表す。第 3 の条件付き等式は、ネットワーク上を流れたメッセージ列の最初のメッセージ中に攻撃者の公開鍵で暗号化された PMS が含まれない場合には、メッセージ列の残りについて、帰納的に、攻撃者が利用可能な PMS を検査することを表す。

図 8.18 (p. 84) は、プロトコル通信状態に対して、ネットワーク上を流れるメッセージを観測する演算  $nw$  により得られたデータに関して、PMS の存在判定を行う述語  $\text{\in}$  の関係を示す図である。

ノードは、状態、データを表す集合を表し、矢印は、集合から集合への演算を表している。

#### 8.4.8.2 構造帰納法

ある性質が任意の状態でも成り立つことを証明するために、構造的帰納法に基づき以下の 2 つを証明する。

- 初期状態  $init$  において性質  $P$  が成り立つ
- ある状態  $s$  で性質  $P$  が成り立つとき、任意の状態遷移 ( $t = chello, \dots, fakeSfin22$ ) 後の状態 ( $s' = t(s, \dots)$ ) においても性質  $P$  が成り立つ

$$\frac{P(init) \quad \forall s, t. P(s) \Rightarrow P(t(s, \dots))}{\forall s. P(s)} \quad (8.2)$$

各状態遷移に関する帰納ステップの証明においては、場合分けによる状態空間の分割および、既に証明した性質（不変論理式）を補題として用いる必要がある場合もある。

例として、性質 1 に関する証明譜について説明する。帰納法の基底ケースの証明譜は、下記の通りプロトコル状態が  $init$  の場合について、任意の Premaster Secret を示すコンスタント  $pms$  を用いて以下のように記述される。

```

open INV
  red inv1000(init, pms) .
close

```

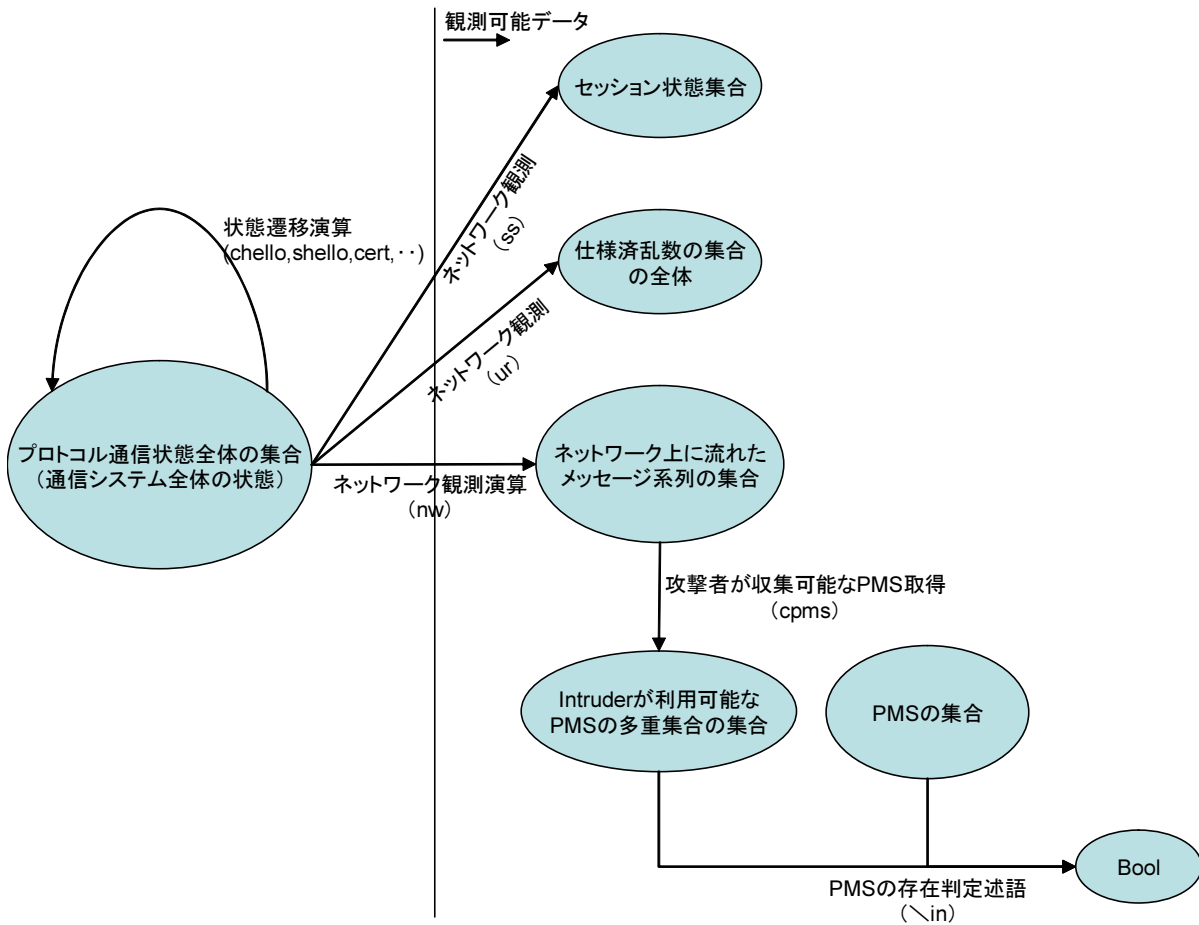


図 8.18: 状態集合と状態観測に対する述語の関係

次に、帰納ステップは、12 の正規通信者によるメッセージ交換に対応した観測状態遷移規則および攻撃者の状態遷移規則 15 個のすべての場合について、帰納法の仮定部が成り立つ場合に、帰納法の帰結部が成り立つことを証明する。各状態遷移規則に関して、実際には、帰納法の前提だけでは書き換えによる証明に成功しない場合がある。そのような場合には、状態空間を分割する条件式による場合分けを導入し、それによって利用可能となる条件式も用いることで、帰納法の帰結部を true に導くことを目指す。下記は、状態遷移規則 chello に関して、帰納法の前提部論理式および、場合分け  $c\text{-chello}(p,a10,b10,r10,110) = \text{true}$  あるいは、 $\text{not}(c\text{-chello}(p,a10,b10,r10,110) = \text{true})$  を用いた証明譜を示している。

```
--> 1) chello(p,a10,b10,r10,l10)
-- 1.1) c-chello(p,a10,b10,r10,l10)
open ISTEP
-- arbitrary objects
ops a10 b10 : -> Prin .
op r10 : -> Rand .
op l10 : -> ListOfChoices .
-- assumptions
-- eq c-chello(p,a10,b10,r10,l10) = true .
eq r10 \in ur(p) = false .
-- successor state
eq p' = chello(p,a10,b10,r10,l10) .
-- check if the predicate is true.
red istep1000(pms) .
close
--
-- 1.2) not c-chello(p,a10,b10,r10,l10)
open ISTEP
-- arbitrary objects
ops a10 b10 : -> Prin .
op r10 : -> Rand .
op l10 : -> ListOfChoices .
-- assumptions
eq c-chello(p,a10,b10,r10,l10) = false .
-- successor state
eq p' = chello(p,a10,b10,r10,l10) .
-- check if the predicate is true.
red istep1000(pms) .
close
```

### 8.4.9 証明実行結果

証明実行は、以下の順に、形式仕様および証明譜を CafeOBJ 処理系にロードすることにより行われる。

1. TLS 形式仕様、述語オペレータ定義の読み込み
2. 検証ゴール定義（不変論理式）の読み込み
3. 証明譜の実行

各処理実行時のメッセージは以下のようになる。

```
CafeOBJ> processing input : ./tls.mod
-- defining module* PRINCIPAL...._* done.
-- defining module! SPECIALPRINCIPAL..._* done.
-- defining module* RANDOM...._* done.
-- defining module* CHOICE...._* done.
-- defining module* LISTOFCHOICES....._* done.
:
(省略)
:
-- defining module* TLS.,,....._* done.
```

検証ゴール定義の読み込みは以下の通り。

```
CafeOBJ> processing input : ./invariants.mod
-- defining module INV....._
....._* done.
-- defining module ISTEP....._* done.
```

証明譜実行時のメッセージは以下の通り。

```

CafeOBJ> processing input : ./proof0900.mod
-- opening module INV.. done.
-- reduce in %INV : inv0900(init,a,k)
true : Bool
(0.000 sec for parse, 7 rewrites(0.000 sec), 12 matches)
--> 1) chello(p,a10,b10,r10,l10)
-- opening module ISTEP.. done._*
-- reduce in %ISTEP : istep0900(a,k)
true : Bool
(0.000 sec for parse, 34 rewrites(0.000 sec), 255 matches)
-- opening module ISTEP.. done._*
-- reduce in %ISTEP : istep0900(a,k)
true : Bool
:
(省略)
:
CafeOBJ> processing input : ./proof1150.mod
-- opening module INV.. done._*
-- reduce in %INV : inv1020(p,a,b,b3,r1,r2,l,c,i,s)
                    and inv1100(p,a,b,b1,r1,r2,l,c,i,s) and
                    inv1110(p,a,b,b2,r1,r2,l,c,i,s,k)
                    implies inv1150(p,a,b,b1,b2,b3,r1,r2,l,c,i,s,k)
true : Bool
(0.000 sec for parse, 2312 rewrites(0.940 sec), 21027 matches)
CafeOBJ> processing input : ./proof1160.mod
-- opening module INV.. done._*
-- reduce in %INV : inv1040(p,a,b,b2,r1,r2,c,i,s)
                    and inv1140(p,a,b,b1,r1,r2,c,i,s)
                    implies inv1160(p,a,b,b1,b2,r1,r2,c,i,s)
true : Bool
(0.000 sec for parse, 443 rewrites(0.170 sec), 4360 matches)

```

証明譜実行時のメッセージにおいて、項書換えにより不変論理式がすべての場合について true に簡約されることを確認する。上記のメッセージにおいて簡約によって得られた項 true の次の行は、実行した項書換えの情報を示している。例えば、最後の true の次の行は、定義した等式とのマッチングは 4360 回行われ、443 回の項書換え（等式推論）が、0.17 秒で実行されたことを示している。

本検証における形式仕様の読み込みと検証実行に要する時間は、PentiumIV 1.7GHz CPU で約 10 分程度である。

## 8.5 検証結果の分析

### 8.5.1 本モデルで保証されない安全性

本モデルでは、プロトコルのモデル化の章で示した通り、TLS プロトコルにおけるオプション的なメッセージ交換の一部を省略することで、サーバの認証は行わすが、クライアントの認証は行わない(サーバ認証モード)。

本モデルにおいて、1)Premaster secret の機密性 (性質 7) と 2) クライアントにとってのサーバからのメッセージの認証 (性質 8,9,10,9) について検証した。

2) に対応するサーバ側の安全性として、サーバにとってのクライアントメッセージの認証は以下のとおりである。

#### 性質 17 (ServerFinished メッセージの認証)

信頼できるサーバがプロトコルに則った ClientFinished メッセージをクライアントと思われる通信者から受信した場合には、そのメッセージは確かにクライアントから送信されたものである。

#### 性質 18 (セッション再開 ServerFinished メッセージの認証)

信頼できるサーバがプロトコルに則った ClientFinished2 メッセージをクライアントと思われる通信者から受信した場合には、そのメッセージは確かにクライアントから送信されたものである。

本性質の証明は成功しない。代わりに、項書換え系を用いた検証プロセスで以下の反例を発見することができ、これらの性質は偽であることが示された。

(性質 17 に対する反例となるメッセージ交換列)

1.  $\text{ch}(a', a', b', r_{a'}, l)$
2.  $\text{sh}(b, b, a, r_b, \text{sid}, c)$
3.  $\text{ct}(b, b, a, \text{cert}(b, k_b, \text{sig}(ca, b, k_b)))$
4.  $\text{kx}(b, b, a, (\text{epms}(k_b, \text{pms}')))$
5.  $\text{cf}(a', a, b, \text{ecfin}(k(a, \text{pms}', r_{a'}, r_b), \text{cfin}(a, b, \text{sid}, l, c, r_{a'}, r_b, \text{pms}')))$

この反例は、サーバ  $b$  がクライアント  $a$  と思われるものから ClientFinished メッセージを受信したとしても、クライアント  $a'$  から送信される場合があることを示している。

このようにして、サーバ  $b$  は、クライアント  $a'$  をクライアント  $a$  と信じてセッションを確立した場合、以下のようにセッション再開に関して性質 18 の反例が存在する。

(性質 18 に対する反例となるメッセージ交換列)

1.  $\text{ch2}(a', a', b', r_{a'}, \text{sid})$
2.  $\text{sh}(b, b, a, r_b, \text{sid}, c)$
3.  $\text{sf}(b, b, a, \text{esfin2}(k(b, \text{pms}', r_{a'}, r_b), \text{sfin2}(a, b, \text{sid}, l, c, r_{a'}, r_b, \text{pms}')))$
4.  $\text{cf}(a', a, b, \text{ecfin2}(k(a, \text{pms}', r_{a'}, r_b), \text{cfin2}(a, b, \text{sid}, l, c, r_{a'}, r_b, \text{pms}')))$

本モデルでは、クライアント認証が行われていないため、性質 17 および 18 が保証されないと理解することができる。TLS の仕様書を読んでいるだけでは、このような性質が成り立たないことを理解するに至らない場合がある。

この例からも分かる通り、プロトコルの形式的な仕様記述および検証を通じて、プロトコルの潜在的な問題点、リスクの発見に役立つことが示されたと言える。

### 8.5.2 モデルの拡張

本検証におけるモデル化では、TLS ハンドシェイクプロトコルのオプションなメッセージ交換の有無を固定した。これにより、条件分岐による検証の複雑化を回避した。一部実行を省略したこれらのオプションなメッセージ交換を含めたプロトコルの検証のためには、以下のようなメッセージ交換のモデルに拡張するとよい。

ClientHello	$A \rightarrow B$	: $Rn_{ad_A}, ListOfChoices$
ServerHello	$B \rightarrow A$	: $Rn_{ad_B}, SID, Choice$
Certificate	$B \rightarrow A$	: $Cert_B$
ServerKeyExchange	$B \rightarrow A$	: $KeyExchParams$
CertificateRequest	$B \rightarrow A$	: $CAType, CAList$
ClientCertificate	$A \rightarrow B$	: $Cert_A$
ClientKeyExchange	$A \rightarrow B$	: $\mathcal{E}_{K_B}(PMS)$
CertificateVerify	$A \rightarrow B$	: $\mathcal{E}_{K_A}(PMS)$
ClientFinish	$A \rightarrow B$	: $\mathcal{E}_{ClientKey}(ClientFinish)$
ServerFinish	$B \rightarrow A$	: $\mathcal{E}_{ServerKey}(ServerFinish)$
ClientHello2	$A \rightarrow B$	: $Rn_{ad_A}, SID$
ServerHello2	$B \rightarrow A$	: $Rn_{ad_B}, SID, Choice$
ServerFinish2	$B \rightarrow A$	: $\mathcal{E}_{ServerKey}(ServerFinish2)$
ClientFinish2	$A \rightarrow B$	: $\mathcal{E}_{ClientKey}(ClientFinish2)$

図 8.19: オプションなメッセージ交換をすべて含む TLS ハンドシェイクプロトコルのモデル

また、今回鍵交換方式として、RSA に限定したモデル化を行ったが、DH 鍵交換を加え条件分岐を用いたモデル化を考えることができる。

### 8.5.3 メッセージ送受信観測のモデル化

TLS の本モデル化においては、観測状態遷移システムにおけるプロトコル状態に関する観測演算として、プロトコルの 1 ステップに対して、1 観測演算を定義している。これに対して、1 プロトコルステップに対して、メッセージ送信の観測およびメッセージ受信の観測に対応した 2 種類の観測

演算を定義することで、単一のメッセージ送受信における改竄等の性質を記述する際により的確な論理式の記述が可能になると考えられる。図 8.20 (p. 90) は、ServerCertificate メッセージに対応する状態遷移演算  $\text{cert}$  に関して、状態遷移が生じた場合に、サーバ B がこれまでに送信したメッセージの集合  $\text{send}(\text{cert}(P,B,M1,M2),B)$  およびサーバ B がこれまでに受信したメッセージの集合  $\text{recieve}(\text{cert}(P,B,M1,M2),B)$  を観測した結果を定義するものである。

```

ceq send(cert(P,B,M1,M2),B) = ct(B,B,dst(M2),cert(B,k(B),
                                sig(ca,B,k(B))))
                                , send(P,B)
    if c-cert(P,B,M1,M2) .

ceq recieve(cert(P,B,M1,M2),B) = sh(B,B,src(M),R,I,C) , send(P,B)
    if c-cert(P,B,M1,M2) .

```

図 8.20: メッセージ送受信観測のモデル

ただし、その場合には、状態遷移観測数が増えることによる検証の規模増大による困難が生じることが予想される。

#### 8.5.4 課題

本検証では、すべての状態遷移演算に対して、構造帰納法に基づく証明により、任意の到達可能な状態に関する性質を検証している。これらの状態遷移演算に関する証明および各状態遷移演算に於ける証明の場合分けの網羅性を自動的に検査する方法を考案することが課題として挙げられる。網羅性の自動検査により、第3者が検証結果を確認するために有益であると考えられる。

さらに、状態遷移演算に対する構造帰納法による証明譜の構造は共通する場合が多いため、状態遷移演算の場合分けに関する証明譜の自動生成により、検証の手間を軽減することができると思われる。

## 第III部

# 総合分析編



## 第9章

# 攻撃法の分類とフォーマルメソッドによる検証可能領域

### 9.1 攻撃法の分類

第4章においてセキュアプロトコルに対する既知の攻撃法について原因、影響、対策等について分析した。これらをもとに、攻撃法を分類し対応する事例を示したものが図9.1 (p. 93) である。攻撃法事例は、表4.2～表4.9で示したもので、それらの表中で攻撃法名を明記していないものには括弧中に攻撃法IDを示した。

攻撃法分類				攻撃事例	
プロトコル脆弱性	実装エラー	バッファオーバーフロー			
		運用設定不備			
		タンパー攻撃			
		その他			
	仕様脆弱性	暗号の脆弱性			トラフィック解析攻撃(TLS14)
		プロトコルメカニズム	暗号適用不備	メッセージ認証不備(完全性)	Key-exchange algorithm rollback攻撃
				相手認証不備(認証)	Version rollback攻撃、Ciphersuite Rollback攻撃
				秘密漏洩(機密性)	VPN Brute force攻撃
				その他(応答情報漏洩など)	Bleichenbacher攻撃、RSAタイミング攻撃
		明文秘密漏洩			セッションID攻撃(TLS16)
		オプション強度不完全			成済まし攻撃(TLS6)
		プロトコル干渉により生じる脆弱性			初期化ベクトル攻撃(IPSEC5)
その他			シングルクォーターション攻撃(SSH9)		

図9.1: プロトコルの脆弱性分類と攻撃法の対応関係

攻撃法は、原因となる脆弱性に基づき図に示す通り系統的に分類することができる。もっとも大きな分類は、実装に関する脆弱性と仕様に関する脆弱性による分類である。本調査では、実装に関する攻撃法は調査対象外とした。仕様の脆弱性に係わる攻撃法には、暗号自体の脆弱性によるもの

と、プロトコルのメカニズムの脆弱性によるものに分類される。さらに、プロトコルのメカニズムに関する攻撃法の内、暗号技術の適用不備が原因となるものが多数みられた。これらはさらに、メッセージ認証の不備によるもの、相手認証の不備によるもの、メッセージ暗号化の不備によるものに分類される。さらにその他の分類では、サイドチャネル攻撃等の応答情報漏洩が原因となる攻撃法が多数見られた。暗号適用不備によるもの意外では、秘密に係わる情報を平文で不用意に送信している場合、プロトコルの暗号オプションにおいて、十分な強度がえられないもの、他のプロトコルとの干渉による情報漏洩等がおもなものである。

## 9.2 フォーマルメソッドによる検証可能領域

図 9.2 (p. 94) は、プロトコルの種類と攻撃法の分類空間における攻撃法事例を示した攻撃法分類マップである。

		攻撃法の主な分類項目							
		暗号の脆弱性	メッセージ認証	相手認証	メッセージ暗号化	オプション強度	応答情報漏洩	プロトコル干渉	その他
TLS/SSL	レコードプロトコル	TLS14 (トラフィック解析)					TLS4(Vaudenary)		
	ハンドシェイクプロトコル		TLS7(Version Rollback) TLS10(KeyExchangeRollback), TLS11(Dropping-the-cipher), TLS12(Ciphersuite) TLS13(Finished)	TLS5 (証明書 Basic Constraints) TLS9.		TLS6(成済み)	TLS1~TLS4 (Bleichenbacker攻撃) TLS8		TLS16(セッションID)
	暗号仕様変更プロトコル		TLS11(Dropping-the-change-cipher)						
	警告プロトコル		TLS15(Premature)				TLS4(Vaudenary)		
	アプリケーションデータプロトコル						TLS4(Vaudenary)		
IPsec	AHプロトコル		IPSEC4(フラグメンテーション)						IPSEC6(順序) IPSEC8(SA)
	ESPプロトコル		IPSEC3 (Cut-and-Paste) IPSEC4 (フラグメンテーション)					IPSEC5(IV) IPSEC7	IPSEC6(順序) IPSEC8(SA)
	IPCompプロトコル								
	ISAKAMPプロトコル			IPSEC9(Identity)					
	IKEプロトコル		IPSEC2 IPSEC12(Proposal)	IPSEC11(Reflection)	IPSEC10(BruteForce) IPSEC13(Fluhrer)		IPSEC1 (RSAタイミング)		
SSH	SSH-1		SSH1,SSH2	SSH7			SSH8(RSAタイミング)	SSH4	SSH3,SSH5 SSH9,SSH10
	SSHトランスポート層プロトコル						SSH8(RSAタイミング)		SSH9,SSH10
	SSHユーザー認証プロトコル		SSH6(VersionRollback)						
	SSHコネクションプロトコル								

凡例: 本研究で検証  
検証可能な領域  
一部検証可能な領域  
検証不可または不明

図 9.2: 攻撃法の分類におけるフォーマルメソッドによる検証可能領域

縦軸には、セキュアプロトコルを構成するサブプロトコルをプロトコルレイヤー順に示し、横軸

には、図 9.1 における攻撃法の主な分類を示している。これらの軸によって構成される分類空間上に、第 4 章で抽出した具体的な攻撃法を分布させたものである。

全体を俯瞰して多く見られるメッセージ認証に関する攻撃法は、検証手法により検証されたもの、あるいは検証可能な領域に位置していることが分かる。暗号適用不備に該当する応答情報漏洩に関する攻撃法については、その事例数は多いが、本検証手法では検証は難しいと考えられる。

メッセージ暗号化の不備に関する攻撃は、プロトコル仕様に関する比較的初歩的な欠陥であると考えられ、その脆弱性に基づく攻撃法はほとんど抽出されなかった。

凡例に示す“本研究で検証”と示された影領域は、プロトコル検証編で示した検証法により実際に検証を行った領域である。ただし、第 8.5.1 章で示した通り、これらの領域の攻撃法のうち、一部は、検証の前提からその耐性を検証することはできないものもある。

“検証可能な領域”とは、本検証手法による検証に適した領域を示しており、ここに示すプロトコル以外の同様のプロトコルにおいても応用により検証することが可能であると考えられる。

“一部検証可能な領域”とは本検証手法により一部検証可能であるが、データフォーマット等に関するモデル化が必要なものについては、検証困難である領域を示している。

“検証不可または不明の領域”は、本検証手法による検証が不可能であるかまたは、その可能性が不明である領域を示している。

## 9.3 プロトコル仕様に係わる一般的な留意点

以上の分析を通して、セキュアプロトコル TLS/SSL, IPsec, SSH の仕様に係わる留意点を一般化してまとめる。

### 9.3.1 攻撃の対象となるプロトコル脆弱性の傾向

- 暗号に関する脆弱性

プロトコルに対する攻撃法の原因となる脆弱性には、暗号自体の脆弱性と暗号技術の適用不備によるものがほとんどであった。暗号自体の脆弱性の中には、トラフィック解析による情報漏洩、十分な鍵長による強度を持たないなどの暗号技術の危殆化によるものがある。しかし、セキュアプロトコルは暗号技術を選択オプションとして外部化しているため、一部のオプション選択においてその脆弱性が生じる場合があるが、数は比較的少ない。一方、暗号技術の適用不備によるものはプロトコルの脆弱性の中で多数みられた。その中でもっとも多いものは、メッセージ認証の不備である。また、暗号技術自体の脆弱性とも関連するが、応答情報を用いた暗号解読法も、暗号技術の適用不備に含めることができる。メッセージ認証の不備は、メッセージ交換列の一部で、暗号スイート等の合意の際に、その内容が改竄されるなどの攻撃への耐性を損なうものである。

- メッセージ認証不備

メッセージ認証の不備の中には、メッセージ交換列において、初期に送受信した平文メッセージの真正性を検証するために、メッセージ交換の後期にメッセージ認証を行うプロトコルがー

一般的であるが、そのような場合に仕様中にメッセージ認証を明記しないために、メッセージ認証の欠落を生じる事例がいくつか見られた。また、パケットレベル、暗号ブロックレベル、フラグメントレベルのデータ改竄による認証のすり抜けを行う攻撃法も見られるが、後者の場合には、フォーマルメソッドによる検証を行うためには、データフォーマットレベルでのモデル化が必要となり、検証の規模的な面で困難が伴う。

- 応答情報漏洩

暗号技術の適用不備の中で多いものの1つに応答情報漏洩が挙げられる。これは、暗号アルゴリズムにしたがって処理をしているにも係わらず、サーバ等へのリクエストに対する応答の違いを多数観測することにより、暗号解読を行うもので、サイドチャネル攻撃の一種である。これは暗号アルゴリズムに深く関連するものであるため、プロトコル仕様から問題点の検知は難しいが、RSA タイミング攻撃等の主な暗号技術の脆弱性を明確化することで、防止することができる。これに関しては、本調査の範囲を越えるものであるが、実際のセキュアプロトコルに対する攻撃法の中でいくつか明らかにすることができた。

- オプションの強度

セキュアプロトコルは、実装の柔軟性を高めるために多数の選択オプションを用意している。これらのオプションの中には暗号強度が十分でない暗号を使うものがあるため、オプションで利用する暗号の強度については個別に評価が必要である。

一方、TLS の認証モードのように、はじめからサーバ、クライアントの一方の認証を放棄するオプションもあり、この場合には、一方の相手認証、メッセージ認証等の安全性は保証されない。プロトコルの利用にあたっては、このようなオプションの選択は採用すべきではない。

### 9.3.2 プロトコル仕様の記述範囲の境界

本調査では、プロトコルの問題点と、実装の問題点との境界について微妙な攻撃が多数見られた。例えば、プロトコルに記載していないある応答のために情報が漏れる場合や、証明書破棄リストの確認など認証処理について明記しないために処理を行わないために生じる脆弱性などがそうである。

プロトコルの問題点には、下記の通り仕様に記載することによる問題と未記載による問題が考えられる。

- 仕様の未記載による問題点

- 不要な動作の未記載

実装において、秘密情報を不用意に送信するなど、明らかに問題となることは、仕様中に禁止しなくても、常識の範囲内で防止できると考えられる。しかし、TLS のセッション ID 送信などにおいては、何らかの秘密情報をもとに作成したセッション ID をネットワーク上に流すことは、秘密情報漏洩の原因となりうる。実装者にとって明確ではないこのような動作の禁止を明記することは必要であると考えられ、したがって、この例のような場合、不要な動作の未記載による脆弱性と考えることができる。

- 必要な動作の未記載

認証処理において証明書破棄リストの確認、メッセージ交換列の前後離れたステップにおけるメッセージ認証処理は、一般に実装時に欠陥を招く原因になりやすい。事実、第4章に示したとおり、これらの不備による攻撃がいくつも知られている。これらの処理は、認証処理で必須のものであるとも言えるが、プロトコル仕様に明記することで、実装時の欠陥を防止することが可能である。このような事例を必要な動作の未記載による脆弱性と考えることができる。

- 仕様の記載による問題点

プロトコル仕様に記載してある動作を実行することにより生じる問題点は、明らかにプロトコル仕様の欠陥である。本調査では、既知の攻撃法のうち、これに該当する事例はほとんど見られなかった。実際、このような問題点は、発見されやすく、広く実用に供されているプロトコルにおいてこのような問題点が残っていることは余り無い。

### 9.3.3 プロトコルの前提と安全性の選択

セキュアプロトコルには、それぞれが提供する機能の目的と保証する安全性が設計時に決まっている。例えば、TLS ハンドシェイクプロトコルに関しては、セッション鍵の安全な共有と暗号スイートの合意である。この場合、これらを用いて行われる通信は、通信情報の機密性や相手認証は保証されても、通信内容の否認防止は実現されない。セキュアプロトコルを利用する場合には、このようにプロトコルが提供する機能と保証する安全性を明確にした上で、通信に必要とされる安全性をもとにセキュアプロトコルを選択することが必要である。

### 9.3.4 フォーマルメソッドを用いたプロトコル検証の効果

- 安全性に関する論理的な保証

第II編において、代数的手法を用いて TLS ハンドシェイクプロトコルのセッション鍵の安全な共有および暗号スイートの合意内容の真正性について検証した。人手による十分な検証分析を経たプロトコルであっても、実用化後、長い年月を経てプロトコルメカニズム上の致命的な欠陥が見つかる事例が存在する [51]。フォーマルメソッドを用いたプロトコル検証は、前提条件のもとで、いかなる攻撃の組み合わせに対しても安全であることを論理的に保証することができる。一度論理的な保証を与えることで将来的に検証に対する労力は必要としなくなる。フォーマルメソッドによる検証には、大きな負荷とコストを要する場合もあるが、高い安全性を求められる広く普及したプロトコルに対して論理的な保証を与えることは、大きなコストに十分見合う効用であると考えられる。

- 仕様上の潜在的な脆弱性の明確化

第II編のプロトコル検証において、サーバ認証モードによるプロトコルのモデル化では、サーバに対する相手認証等の保証が得られないことは、プロトコル検証前の時点では認識していなかった。この例は、プロトコルのモデル化、検証の過程を経ることにより、プロトコルの仕様に関する潜在的な脆弱性を明確化することに役立つことを実証するものである。

- 仕様の曖昧性の排除

第9.3.2章でも指摘したとおり、フォーマルメソッドによるプロトコルの検証においては、仕様の曖昧性を取り除く必要がある。これにより、プロトコルの問題点を未然に防ぐことに役立つ。

- 前提やリスク要因の明確化

プロトコルの設計および検証は、ある前提の下で行うことが現実的である。フォーマルメソッドを用いた検証においては、前提条件を明示的に定義することが必要になる。また、検証結果に対する評価を行うことで、検証に対するリスク要因を明確化することができる。これらのプロセスを経ることで、プロトコルの安全性を強固なものにすることができる。

## 第 10 章

# まとめ

本調査研究では、セキュアプロトコル TLS/SSL, IPsec, SSH に関する既知の攻撃法を網羅的に調査し、TLS に関してフォーマルメソッドを用いた検証を行った。これらに基づき、攻撃法の傾向分類、およびフォーマルメソッドを用いた検証可能な領域について分析した。

第 II 編の代数的手法に基づく検証結果により TLS ハンドシェイクプロトコルのセッション鍵の安全な共有および暗号スイートの合意内容について論理的にその安全性が保証できることを示した。ただし、ここでは、第 6 章に示す前提条件においてのみ保証されることに注意が必要である。これにより第 1 章でしめした以下の仮説を検証することができた。

フォーマルメソッドを用いたセキュアプロトコルの検証により、特定の安全性に関する論理的な保証を確立すると共に、プロトコル仕様上の潜在的なリスクを明らかにすることができる。

一方、第 III 編における表 9.2 の分析においてフォーマルメソッドによる検証が、TLS, IPsec, SSH の内、ハンドシェイクプロトコル (TLS), IKE プロトコル (IPsec), コネクションプロトコル (SSH) などメッセージ交換等を規定したプロトコル仕様の論理的な面について、メッセージ認証、相手 認証、メッセージ暗号化などに関する攻撃法に対して、通信情報の機密性、合意内容の真正性の検証を行うために応用することが可能であることを示した。これにより第 1 章でしめした以下の仮説を検証することができた。

フォーマルメソッドによる検証は、TLS のみならず、IPsec, SSH などの類似するプロトコルにおけるメッセージ交換等の仕様の論理的な側面について、通信情報の機密性、合意内容の真正性等の検証を行うために応用することが可能である。



## 第 IV 部

### 付録



## 付録 A

# セキュアプロトコルに関連する用語の整理

セキュリティ・アソシエーション (SA) セキュリティを確保するためのサービスやメカニズムを完全に特定するためのプロトコルごとに特有のパラメータの集合である。

保護スイート セキュリティプロトコルに適用されるセキュリティサービスのリストである。例えば、保護スイートは、IP ESP のための DES, IP AH のためのキー MD5 などから構成されるリストである。

OSI 参照モデル/OSI 階層モデル/OSI 基本参照モデル 国際標準化機構 (ISO) によって制定された、通信ネットワーク構造の基本設計である OSI に基づいて、通信機能に求められる機能を階層構造化したモデルである。ここでは通信に求められる機能別に、通信を 7 つの階層に分けて考えている。インターネット通信もこの OSI 参照モデルに基づいて設計されている。

HTTP HTTP(HyperText Transfer Protocol) は、WEB サーバと WEB クライアント (WEB ブラウザ) がデータを転送するために利用されるアプリケーション層プロトコルである。クライアントは URL(Universal Resource Location) で指定したデータの送信要求をサーバへ送信し、サーバはその要求に応じて URL で示されるデータを返信する。HTTP/1.0 が RFC 1945、HTTP/1.1 が RFC 2616 として規定されている [10] [29]。通常の HTTP 通信では通信内容は平文で転送される。

POP POP(Post Office Protocol) は、メールをメールサーバから取得するのに利用されるアプリケーション層プロトコルである。現在 POP3 として RFC で規定されている [84]。POP 通信では通信内容は平文で転送される。

SMTP SMTP(Simple Mail Transfer Protocol) はメールの送信をメールサーバに以来する際に利用されるアプリケーション層プロトコルである。RFC 821 で規定されている [80]。SMTP 通信では通信内容は平文で転送される。

IMAP IMAP(Internet Message Access Protocol) はインターネット上で、メールをメールサーバから取得するのに利用されるプロトコルである。現在は IMAP4rev1 が RFC 2060 として規定されている [18]。IMAP では POP とは異なり、メール本文をサーバへ置いたまま処理することができる。通常の IMAP 通信では通信内容は平文で転送される。

ネゴシエーション通信を行う 2 つのホスト間で、通信方式などの通信に必要な情報を交換し、通信設定を決定する処理を指す。暗号通信であっても、暗号方式や暗号鍵の選択にネゴシエーションを必要とし、各セキュアプロトコルはその機能を備えている。

ハッシュ関数 与えられたメッセージから、固定長のハッシュ値を計算するために利用される関数。ハッシュ関数は不可逆な一方関数を含むため、特定のハッシュ値に合わせてメッセージを改変することは困難である。そのためメッセージとハッシュ値を組にして送信することで、受信者はメッセージの改変の有無を確認することが可能となる。

リプレイ攻撃/再送攻撃 セキュアプロトコルなどの認証技術に対して、第三者が有効な認証過程を盗聴し、それと同じ過程を認証パケットなどを再送し、再現することで認証を得ようとする攻撃。セキュアプロトコルでは、再送攻撃を防ぐために、認証パケットなどは順序番号も含める形で利用され、単純に過去の認証パケットの再送では認証が成功しないように作られている。

公開鍵暗号 公開鍵暗号は、暗号化と復号化で利用する暗号鍵が異なる暗号方式である。通常は 2 対の鍵が生成され、一つで暗号化されたものは別の一方でないと復号できない。この性質を利用して、鍵の一つを公開鍵として公開し、一方を秘密鍵として秘匿して保管する。共通の暗号鍵を利用する、秘密鍵暗号方式と異なり、暗号鍵を秘密に通信相手に送る必要がないため鍵管理が容易で、インターネットのような不特定多数との暗号通信に適している。この暗号方式によって、自己証明も行うことができる。

秘密鍵暗号 暗号化と復号化で利用する暗号鍵が同じ暗号方式である。暗号データの送信とは別に、安全な通信路を使って、暗号鍵を共有する必要がある。公開鍵暗号が考案されるまでは、秘密鍵暗号が暗号方式として一般的なものであった。

暗号スイート 暗号スイートとは、安全な通信のために利用される暗号方式をまとめたものである。ユーザーは暗号スイートを選択することで、暗号方式を一括して選択する。暗号スイートに含まれるアルゴリズムは、TLS の場合であると、鍵交換アルゴリズム、暗号アルゴリズム、認証アルゴリズムである。

X.509 公開鍵暗号での鍵や証明書フォーマットを規定する、公開鍵証明書の規格の一つが X.509 で、ITU (International Telecommunication Union) や ISO (International Organization for Standardization) で標準化されている。その他の公開鍵証明書フォーマットとしては SPKI (Simple Public Key Infrastructure) や SDSI (Simple Distributed Security Infrastructure) などが存在するが、TLS や S/MIME などの多くのセキュアプロトコルが X.509 を利用しているため、X.509 は公開鍵証明書の規格の中でも一般的なものとなっている。

X.509 は RFC 2459[38] を始めとする RFC によって規定されており、IETF の PKIX WG によって X.509 公開鍵証明書を用いた公開鍵インフラの標準化が進められている。

メッセージ認証コード メッセージ認証コード (Message Authentication Code:MAC) は、通信メッセージの改変を検出するために利用される。MAC では、ハッシュ関数のようなメッセージ全体からなんらかの要約値 (ダイジェスト値) を出すダイジェスト関数を用いてメッセージの認証を行う。あらかじめ送信元と、送信先の間で、利用するダイジェスト関数の種類や、ダイジェスト関数で用いる引数について取り決めをしておく。その後の通信では、ダイジェスト関数などを用いて通信メッセージのダイジェスト値を計算し、通信メッセージと合わせて送信する。通

信途中でメッセージになんらかの改変が加えられた場合には、送信先ホストでダイジェスト値を計算することでメッセージの改変を検出可能となる。

インターネットドラフト インターネットドラフトとはインターネット技術の提案文書である。インターネットドラフトは誰もが自由に、IETF に対して提案することができる。インターネットドラフトは投稿されてから 6 ヶ月間 IETF で保管され、6 カ月後に Expire(失効)される。インターネットドラフトは IETF での議論を経て、RFC として正式に公開される。

認証局 (Certificate Authority) インターネット上でのセキュアプロトコルでは、公開鍵暗号の仕組みを利用したものが多数存在する。公開鍵暗号では、暗号化や署名の検証を行うために、あらかじめ対象となる公開鍵を取得しておく必要がある。そこでインターネット上で公開鍵の配布を行うために、信用のおける第三者機関による公開鍵の配布というアイデアが考えられた。この第三者機関を認証局 (Certificate Authority) と呼ぶ。認証局は配布する公開鍵を、自身の秘密鍵で署名した「公開鍵証明書」として公開する。こうして公開鍵の正当性を保証しつつ配布することが可能になる。

ポートフォワーディング ここでいうポートとは、トランスポート層プロトコル TCP での識別子であるポートを指す。SSH では、クライアントは任意の TCP ポートを SSH サーバの指定 TCP ポートへの転送や、逆にサーバの任意 TCP ポートをクライアントの任意ポートへの転送機能も持つ。この転送路は SSH 暗号通信路を利用するため、安全な通信を他の TCP 通信を行うアプリケーションに対しても提供できる。

ペイロード 通信パケットから、ヘッダ部を除いた残り、転送するデータそのものをペイロードと呼ぶ。

IP データグラム OSI 参照モデルのネットワーク層での、データ通信の基本単位。ネットワーク層では、通信データは IP データグラムとして転送される。

トンネルモード ここでのトンネルモードとは、IPsec におけるトンネルモードについて説明する。IPsec 通信では、IP データグラムを対象として暗号化を施す。IP データグラムを暗号化した際に、暗号化 IP データグラムをペイロードとする、新たな IP データグラムを作ることができる。IP データグラムの中に IP データグラムを含む、これを IP カプセル化と呼び、このとき内部 IP データグラムと外部 IP データグラムは完全に独立したパラメータとなる。これは暗号化 IP データグラムは、IP データグラムの転送ネットワークから完全に独立することを意味する。そのため、IPsec 通信では、暗号化～復号化の転送区間では IP データグラムはあたかもトンネルを通過しているように見える。これを IPsec のトンネルモード転送と呼ぶ。

トランスポートモード ここでのトランスポートモードとは、IPsec のトランスポートモードについて説明する。IPsec 通信では、IP データグラムを対象として暗号化を施すが、トンネルモードのように暗号化 IP データグラムを再度、新たな IP データグラムとして構成する方法に対して、トランスポートモードでは、暗号化対象 IP データグラムのペイロード部のみを暗号化し、元の IP ヘッダのまま転送を行う。この場合は、データ部分だけが暗号化されたパケットが転送されているように見える。これを IPsec 通信でのトランスポートモード転送と呼ぶ。

フラグメント処理 インターネットの通信は、OSI 参照モデルに基づく階層構造を持っている。通信の上位層から下位層に転送データの受け渡しの際に、それぞれの層で最大データサイズが異なる

る場合がある。上位層より下位層の最大データサイズが小さい場合は、データの分割が発生する。これをフラグメントと呼び、この分割、結合処理をフラグメント処理と呼ぶ。

パディングデータサイズを、暗号処理のブロック単位長に合わせるために挿入する無意味なデータをパディングと呼ぶ。

SPI SPI(Security Parameter Index) は、IPsec ESP ヘッダ内に挿入される 32 ビットの固定長の値で、SA を識別するために利用される。IPsec 通信で、SA のネゴシエーションが完了すると、IPsec 通信の送信元と、受信先で SA と SPI 値の対応が決定される。この対応関係は、IPsec 通信の当事者間でのみ有効となる。

観測遷移機械 (観測遷移システム) 対象システムに関連する値およびそれらが状態遷移によりどのように変化するのかを状態の外部から観察することでモデル化したシステム。

IDEA IDEA (International Data Encryption Algorithm) は Xhejia Lai と James Massey によって開発された商用暗号化アルゴリズムである。IDEA の商用目的利用にはライセンスの購入が必要である。

RC2 RSA データセキュリティ社 (米) が開発した鍵長可変のブロック型暗号である。暗号アルゴリズムは RFC2268 として公開されている [83]。TLS では、鍵長 40bits の RC2 が利用される。

RC4 RSA データセキュリティ社 (米) が開発した鍵長可変のストリーム型暗号である。暗号アルゴリズムは基本的には非公開である。TLS では、鍵長 40bits, 128bits の RC4 暗号が利用される。

DES DES(Data Encryption Standard) は、1977 年に NIST(National Institute of Standards and Technology:米国標準技術協会) によって標準化された暗号アルゴリズムである [71]。TLS では鍵長 56bits のものが利用される。計算機の計算能力の増加により、解読の危険性が指摘され、近年では Triple DES などの改良型アルゴリズムが提案されている。

Triple-DES DES の改良版として Triple DES が提案された [93]。Triple DES は DES の変形で、暗号対象データに対して、異なる鍵で DES を 3 度適用することで、DES に比べてより安全度を高めている。

MD5 MD5 (Message Digest) はメッセージ全体からメッセージ要約値として 128 ビットのハッシュ値を計算するハッシュ関数である。RFC 1321 として規定されている [82]。

SHA-1 SHA-1(Secure Hash Algorithm) は RFC3174 として規定されているハッシュ関数である [20]。SHA-1 では 160 ビットのメッセージ要約値を計算する。

## 付録 B

### 関連する RFC

表 B.1: IPsec プロトコルに関連する主要 RFC

RFC#	タイトル	発行
2412	The OAKLEY Key Determination Protocol.	Nov,1998
2411	IP Security Document Roadmap.	Nov,1998
2410	The NULL Encryption Algorithm and Its Use With IPsec.	Nov,1998
2409	The Internet Key Exchange (IKE).	Nov,1998
2408	Internet Security Association and Key Management Protocol(ISAKMP).	Nov,1998
2407	The Internet IP Security Domain of Interpretation for ISAKMP.	Nov,1998
2406	IP Encapsulatio Security Payload(ESP).	Nov,1998
2405	The ESP DES-CBC Cipher Algorithm With Explicit IV.	Nov,1998
2403	The Use of HMAC-MD5-96 within ESP and AH.	Nov,1998
2402	IP Authentication Header.	Nov,1998
2401	Security Architecture for the Internet Protocol.	Nov,1998

表 B.2: TLS プロトコルに関連する主要 RFC

RFC	タイトル	
2830	Lightweight Directory Access Protocol(v3): Extension for Transport Layer Security.	May 2000
2818	HTTP over TLS.	May 2000
2817	Upgrading to TLS Within HTTP/1.1.	May 2000.
2716	PPP EAS TLS Authentication Protocol.	Oct 1999
2712	Addition of Kerberos Cipher Suites to Transport Layer Security (TLS).	Oct 1999
2595	Using TLS with IMAP, POP3 and ACAP.	Jun 1999
2487	SMTP Service Extension for Secure SMTP over TLS.	Jan 1999
2246	The TLS Protocol Version 1.0.	Jan 1999

表 B.3: SSH プロトコルに関連するインターネットドラフト

タイトル	発行ドラフト
SSH Transport Layer Protocol	draft-ietf-secsh-transport-17.txt
SSH Authentication Protocol	draft-ietf-secsh-userauth-18.txt
SSH Connection Protocol	draft-ietf-secsh-connect-18.txt
SSH Protocol Architecture	draft-ietf-secsh-architecture-15.txt
Generic Message Exchange Authentication For SSH	draft-ietf-secsh-auth-kbdinteract-05.txt
GSSAPI Authentication and Key Exchange for the Secure Shell Protocol	draft-ietf-secsh-gsskeyex-07.txt
SECSH Public Key File Format	draft-ietf-secsh-publickeyfile-04.txt
Diffie-Hellman Group Exchange	
Diffie-Hellman Group Exchange for the SSH Transport Layer Protocol	draft-ietf-secsh-dh-group-exchange-04.txt
SSH Protocol Assigned Numbers	draft-ietf-secsh-assignednumbers-05.txt
Using DNS to Securely Publish SSH Key Fingerprints	draft-ietf-secsh-dns-05.txt
SSH Transport Layer Encryption Modes	draft-ietf-secsh-newmodes-01.txt
Session Channel Break Extension	draft-ietf-secsh-break-01.txt
SCP/SFTP/SSH URI Format	draft-ietf-secsh-scp-sftp-ssh-uri-00.txt
Secure Shell Public-Key Subsystem	draft-ietf-secsh-publickey-subsystem-00.txt

## 付録 C

# 攻撃法の情報源

### C.1 プロトコル検証・セキュリティ工学分野における主な研究者

表 C.1 (p.110) は、プロトコル検証およびセキュリティ工学分野における主な研究者を示したものである。

### C.2 プロトコル検証に関連する国際会議・論文誌

プロトコル検証に関連する国際会議および論文誌を分野ごとに示す。

#### C.2.1 暗号関連

- CRYPTO <http://www.iacr.org/conferences/crypto2003/>
- ACM Conference on Computer and Communication Security <http://portal.acm.org/toc.cfm?id=SERIES320&idx=SERIES320&type=series&coll=portal&dl=ACM&part=-series&WantType=Proceedings&title=CCS&CFID=175995&CFTOKEN=98267204>
- Journal of Computer Security <http://www.csl.sri.com/programs/security/jcs/>
- 電子情報通信学会 暗号と情報セキュリティシンポジウム SCIS <http://scis2003.crypt.ss-titech.ac.jp/SCIS2003Program.html>
- International Conference on Information Security and Cryptography <http://www.icisc.org/>

#### C.2.2 セキュリティ工学関連

- Security Protocols Workshop <http://www.informatik.uni-trier.de/ley/db/conf/spw/>
- International Workshop on Security Protocols, Cambridge, England, April 2003 <http://homepages.feis.herts.ac.uk/comqjam/SPW/next-2003/Programme-v01.html>
- USENIX Security Symposium <http://www.usenix.org/>
- ACM Transactions on Information and System Security <http://www.acm.org/tissec/>
- IEEE Computer Security Foundations Workshop <http://www.csl.sri.com/programs/>

表 C.1: プロトコル検証・セキュリティ工学に関する主要な研究者

研究者名	主な研究内容と URL 等
Ross Anderson (Cambridge 大)	ピア・ツー・ピアシステム、暗号プロトコルの頑健性。暗号プロトコルの解析。セキュリティ工学全般。 <a href="http://www.cl.cam.ac.uk/users/rja14/">http://www.cl.cam.ac.uk/users/rja14/</a>
Marti'n Abadi (DEC)	仕様検証、セキュリティ工学、プログラミング言語 <a href="http://www.cse.ucsc.edu/~abadi/home.html">http://www.cse.ucsc.edu/~abadi/home.html</a>
Catherine Meadows (Naval Research Laboratory)	プロトコル検証システム NRL Protocol Analyzer を用いたプロトコル検証 <a href="http://chacs.nrl.navy.mil/projects/crypto.html">http://chacs.nrl.navy.mil/projects/crypto.html</a>
Lawrence C. Paulson (Cambridge 大)	高階論理証明系 Isabelle を用いた検証。TLS, SET の検証。 <a href="http://www.cl.cam.ac.uk/users/lcp/">http://www.cl.cam.ac.uk/users/lcp/</a>
Gavin Lowe (Oxford 大)	並列プロセス代数 CSP, コンピュータセキュリティの形式モデル化 <a href="http://web.comlab.ox.ac.uk/oucl/work/gavin.lowe/">http://web.comlab.ox.ac.uk/oucl/work/gavin.lowe/</a>
Roger Needham (Cambridge 大、Microsoft Research)	Burrows, Abadi と共に BAN 論理によるプロトコル検証。公開鍵暗号の頑健性、モバイルセキュリティ。 <a href="http://research.microsoft.com/users/needham/default.aspx">http://research.microsoft.com/users/needham/default.aspx</a>
Steven Bellovin (AT&T)	IETF Security Area の共同議長。ネットワークセキュリティ。 <a href="http://www.research.att.com/~smb/">http://www.research.att.com/~smb/</a>
Michael Burrows (DEC)	Abadi, Needham とともに BAN 論理によるプロトコル検証。 <a href="http://citeseer.nj.nec.com/burrows90logic.html">http://citeseer.nj.nec.com/burrows90logic.html</a>
Paul Syverson (Naval Research Laboratory)	セキュアシステム設計、セキュリティ解析、フォーマルメソッド。 <a href="http://www.syverson.org/">http://www.syverson.org/</a>
David Basin (Freibourg 大)	セキュアシステム構築のためのフォーマルメソッドのツール開発。 <a href="http://www.informatik.uni-freiburg.de/~basin/">http://www.informatik.uni-freiburg.de/~basin/</a>
David Wagner (UC Berkeley)	暗号理論。コンピュータセキュリティ。共通鍵暗号の設計と解析。 <a href="http://www.cs.berkeley.edu/~daw/">http://www.cs.berkeley.edu/~daw/</a>

[security/csfw/](http://www.cs.fsw.edu/~security/)

- IEEE Symposium on Security and Privacy <http://www.ieee-security.org/TC/SP->

Index.html

- SEC ( International Conference on Information Security ) 2003 <http://www.sec2003.org/>
- 情報処理学会 コンピュータセキュリティ研究会 Computer Security Symposium (CSS) <http://css2003.is.env.kitakyu-u.ac.jp/05pro/1031.html#8B>

### C.2.3 フォーマルメソッド関連

- Workshop on Formal Methods and Security Protocols <http://www.cs.bell-labs.com/who/-nch/fmsp99/program.html>
- DIMACS Workshop on Design and Formal Verification of Security Protocols <http://dimacs.rutgers.edu/Workshops/Security/>
- FM: Formal Methods <http://www.informatik.uni-trier.de/ley/db/conf/fm/>

## C.3 脆弱性情報データベース

以下は、攻撃法調査の情報源とした脆弱性情報データベースである。CVE は、多くの脆弱性情報データベースの脆弱性情報を網羅的にインデックス化し、脆弱性を特定する共通の基盤となるものである。

- CERT Advisories ( <http://www.cert.org/advisories/> )
- CVE ( <http://cve.mitre.org/cve/downloads/full-cve.html> )
- SecurityFocus.Com ( <http://www.securityfocus.com/> )
- CIAC Security Resource ( <ftp://ciac.llnl.gov/pub/ciac/> )



## 参考文献

- [1] Security considerations for ip fragment filtering. IETF RFC1858.
- [2] Security vulnerabilities in ssl/tls. ISO/CS393/CS682.
- [3] Vulnerability: attacking predictable ipsec esp initialization vectors. <http://www.hut.fi/svaarala/publications/espiv/report.html>.
- [4] *Internet Security Protocol*. Prentice Hall, New Jersey, 2000.
- [5] C. Adams. RFC 2144: The CAST-128 encryption algorithm, May 1997. Status: INFORMATIONAL.
- [6] George Apostolopoulos, Vinod Peris, and Debanjan Saha. Transport layer security: How much does it really cost? In *INFOCOM: The Conference on Computer Communications, joint conference of the IEEE Computer and Communications Societies*.
- [7] R. Atkinson. RFC 1825: Security architecture for the Internet Protocol, August 1995. Obsoleted by RFC2401 [41]. Status: PROPOSED STANDARD.
- [8] R. Baldwin and R. Rivest. RFC 2040: The RC5, RC5-CBC, RC5-CBC-pad, and RC5-CTS algorithms, October 1996. Status: INFORMATIONAL.
- [9] Steven Bellovin. Problem areas for the ip security protocols. In *Proceedings of the Sixth USENIX UNIX Security Symposium*, 1996.
- [10] T. Berners-Lee, R. Fielding, and H. Frystyk. RFC 1945: Hypertext Transfer Protocol — HTTP/1.0, May 1996. Status: INFORMATIONAL.
- [11] Daniel Bleichenbacher. Chosen ciphertext attacks against protocols based on the rsa encryption standard pkcs1. In *Lecture Notes in Computer Science*. 1998.
- [12] P. Buhler, T. Eirich, M. Steiner, and M. Waidner. Secure password-based cipher suite for tls. In *Proceedings of Network and Distributed Systems Security Symposium*, February 2000.
- [13] M. Burrows, M. Abadi, and R. Needham. A logic of authentication. *ACM Transactions in Computer Systems*, 8(1):18–36, 1990.
- [14] R. Canetti and H. Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In *Proceedings of Eurocrypt LNCS*, May 2001.
- [15] R. Canetti and H. Krawczyk. Security analysis of ike’s signature-based key-exchange protocol. In *Proc. of the Crypto conference*, August 2002.

- [16] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Annual Symposium on Foundations of Computer Science (FOCS)*, pages 136–145, 2001.
- [17] M. Crispin. RFC 1730: INTERNET MESSAGE ACCESS PROTOCOL — VERSION 4, December 1994. Obsoleted by RFC2060, RFC2061 [18, 19]. Status: PROPOSED STANDARD.
- [18] M. Crispin. RFC 2060: INTERNET MESSAGE ACCESS PROTOCOL — VERSION 4rev1, December 1996. Obsoletes RFC1730 [17]. Status: PROPOSED STANDARD.
- [19] M. Crispin. RFC 2061: IMAP4 COMPATIBILITY WITH IMAP2BIS, December 1996. Obsoletes RFC1730 [17]. Status: INFORMATIONAL.
- [20] 3rd D. Eastlake and P. Jones. RFC 3174: us secure hash algorithm 1 (sha1), September 2001.
- [21] L. P. Deutsch. RFC 1951: DEFLATE compressed data format specification version 1.3, May 1996. Status: INFORMATIONAL.
- [22] L. P. Deutsch and J-L. Gailly. RFC 1950: ZLIB compressed data format specification version 3.3, May 1996. Status: INFORMATIONAL.
- [23] R. Diaconescu and K. Futatsugi. *CafeOBJ Report*, volume 6 of *AMAST Series in Computing*. World Scientific, Singapore, 1998.
- [24] Razvan Diaconescu and Kokichi Futatsugi. Behavioural coherence in object-oriented algebraic specification. *Journal of Universal Computer Science*, 6:74–96, 2000.
- [25] T. Dierks and C. Allen. RFC 2246: The TLS protocol version 1, January 1999. Status: PROPOSED STANDARD.
- [26] D. Dolev and A. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983.
- [27] D. Dolev and A. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983.
- [28] Niels Ferguson and Bruce Schneier. A cryptographic evaluation of ipsec. Technical report, <http://citeseer.nj.nec.com/ferguson00cryptographic.html>, 2000.
- [29] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. RFC 2616: hypertext transfer protocol – http/1.1, June 1999.
- [30] S. Frankel, R. Glenn, and S. Kelly. RFC 3602: the aes-cbc cipher algorithm and its use with ipsec, September 2003.
- [31] S. Frankel and H. Herbert. RFC 3566: the aes-xcbc-mac-96 algorithm and its use with ipsec, September 2003.
- [32] J. Goguen and G. Malcolm. A hidden agenda. *Theoretical Computer Science*, 245:55–101, 2000.
- [33] Li Gong and Paul Syverson. Fail-stop protocols: An approach to designing secure protocols. *IEEE Computer Society*, pages 79–100, 1998.

- [34] D. Harkins and D. Carrel. RFC 2409: The Internet Key Exchange (IKE), November 1998. Status: PROPOSED STANDARD.
- [35] N. Heintze and J. D. Tygar. A model for secure protocols and thier composition. *IEEE Transactions on Software Engineering*, 22(1):16–30, 1996.
- [36] Adam Hess, Jared Jacobson, Hyrum Mills, Ryan Wamsley, Kent E. Seamons, and Bryan Smith. Advanced client/server authentication in tls. In *Network and Distributed System Security Symposium Conference Proceedings*, 2002.
- [37] P. Hoffman. RFC 2487: SMTP service extension for secure SMTP over TLS, January 1999. Status: PROPOSED STANDARD.
- [38] R. Housley, W. Ford, W. Polk, and D. Solo. RFC 2459: Internet X.509 public key infrastructure certificate and CRL profile, January 1999. Status: PROPOSED STANDARD.
- [39] John Ioannidis. How come we still don't have ipsec, dammit? <http://www.tla.org/talks/ipsec-dammit.pdf>. AT&T Labs–Research.
- [40] Jakob Jonsson and Burton S. Kaliski Jr. On the security of rsa encryption in tls. In *CRYPTO*, 2002.
- [41] S. Kent and R. Atkinson. RFC 2401: Security architecture for the Internet Protocol, November 1998. Obsoletes RFC1825 [7]. Status: PROPOSED STANDARD.
- [42] A. Keromytis and N. Provos. RFC 2857: the use of hmac-ripemd-160-96 within esp and ah, June 2000.
- [43] R. Khare and S. Lawrence. RFC 2817: upgrading to tls within http/1.1, May 2000.
- [44] J. Klensin, N. Freed, M. Rose, E. Stefferud, and D. Crocker. RFC 1651: SMTP service extensions, July 1994. Obsoleted by RFC1869, STD0010 [45, 81]. Obsoletes RFC1425 [46]. Status: DRAFT STANDARD.
- [45] J. Klensin, N. Freed, M. Rose, E. Stefferud, and D. Crocker. RFC 1869: SMTP service extensions, November 1995. See also STD0010 [81]. Obsoletes RFC1651 [44]. Status: STANDARD.
- [46] J. Klensin, WG Chair, N. Freed, M. Rose, E. Stefferud, and D. Crocker. RFC 1425: SMTP service extensions, February 1993. Obsoleted by RFC1651 [44]. Status: PROPOSED STANDARD.
- [47] Kli'ma, Pokorny, and Rosa. Attacking rsa-based sessions in ssl/tls. In *CRYPTO*, 1998.
- [48] S. Lehtinen and D. Moffat. ID: diffie-hellman group exchange for the ssh transport layer protocol, July 2003.
- [49] S. Lehtinen and D. Moffat. ID: ssh protocol assigned numbers, October 2003.
- [50] G. Lowe. Breaking and fixing the needham-schroeder public-key protocol using fdr. *Software Concept and Tools*, 17:93–102, 1996.
- [51] Gavin Lowe. An attack on the Needham-Schroeder public-key authentication protocol. *Information Processing Letters*, 56:131–133, 1995.
- [52] C. Madson and N. Doraswamy. RFC 2405: The ESP DES-CBC cipher algorithm with

- explicit IV, November 1998. Status: PROPOSED STANDARD.
- [53] C. Madson and R. Glenn. RFC 2403: The use of HMAC-MD5-96 within ESP and AH, November 1998. Status: PROPOSED STANDARD.
- [54] C. Madson and R. Glenn. RFC 2404: The use of HMAC-SHA-1-96 within ESP and AH, November 1998. Status: PROPOSED STANDARD.
- [55] D. Maughan, M. Schertler, M. Schneider, and J. Turner. RFC 2408: Internet Security Association and Key Management Protocol (ISAKMP), November 1998. Status: PROPOSED STANDARD.
- [56] Meadows. Analysis of the internet key exchange protocol using the nrl protocol analyzer. In *Proc. the Symp. on Security and Privacy, IEEE*, 1999.
- [57] C. Meadows. Applying formal methods to the analysis of a key management protocol. *Journal of Computer Security*, pages 5–53, 1992.
- [58] Catherine Meadows. Open issues in formal methods for cryptographic protocol analysis. In *Proceedings of the DARPA Information Survivability Conference and Exposition*, volume I,II, 2000.
- [59] Catherine Meadows. Formal methods for cryptographic protocol analysis: Emerging issues and trends. *IEEE Journal on Selected Areas in Communication*, 21(1):44–54, 2003.
- [60] P. Metzger and W. Simpson. RFC 1828: IP authentication using keyed MD5, August 1995. Status: PROPOSED STANDARD.
- [61] J. K. Millen, S. C. Clark, and S. B. Freedman. The interrogator: Protocol security analysis. *IEEE Transactions on Software Engineering*, 1987.
- [62] John C. Mitchell, Vitaly Shmatikov, and Ulrich Stern. Finite-state analysis of ssl. In *DIMACS Workshop on Design and Formal Verification of Security Protocols*, 1997.
- [63] S. Mocas and T. Schubert. Formal analysis of ip layer security. In *Proceedings of DIMACS Workshop on Design and Formal Verification of Security Protocols*, September 1997.
- [64] J. Myers and M. Rose. RFC 1725: Post office protocol — version 3, November 1994. Obsoleted by RFC1939, STD0053 [65, 66]. Obsoletes RFC1460 [84]. Status: DRAFT STANDARD.
- [65] J. Myers and M. Rose. RFC 1939: Post Office Protocol — version 3, May 1996.
- [66] J. Myers and M. Rose. STD 53: Post Office Protocol — Version 3, May 1996. See also RFC1939 [65]. Obsoletes RFC1725 [64].
- [67] J. Myers and M. Rose. RFC 2821: Simple mail transfer protocol, April 2001.
- [68] Ataru N. Nakagawa, Toshimi Sawada, and Kokichi Futatsugi. CafeOBJ user’s manual – ver 1.4.2 –. <http://www.1dl.jaist.ac.jp/cafeobj/doc/>, 1999.
- [69] R. M. Needham and M. D. Schroeder. Using encryption for authentication in large networks of computers. *Communications of ACM*, 21(12):993–999, 1978.
- [70] C. Newman. RFC 2595: using tls with imap, pop3 and acap, June 1999.
- [71] US National Bureau of Standards. Data encryption standard. *Federal Information Pro-*

- cessing Standard (FIPS) Publication, 46, January 1977.*
- [72] K. Ogata and K. Futatsugi. Formally modeling and verifying Ricart&Agrawala distributed mutual exclusion algorithm. In *2nd Asia-Pacific Conference on Quality Software*, pages 357–366. IEEE CS Press, 2001.
  - [73] K. Ogata and K. Futatsugi. Modeling and verification of distributed real-time systems based on CafeOBJ. In *16th IEEE International Conference on Automated Software Engineering*, pages 185–192. IEEE CS Press, 2001.
  - [74] K. Ogata and K. Futatsugi. Formal analysis of Suzuki&Kasami distributed mutual exclusion algorithm. In *5th International Conference on Formal Methods for Open Object-Based Distributed Systems*, pages 181–195. Kluwer Academic Publishers, 2002.
  - [75] Kazuhiro Ogata and Kokichi Futatsugi. Formal verification of the horn-preneel micropayment protocol. In *Proc. of the 4th Int'l Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI 2003)*, LNCS 2575, pages 238–252. Springer, 2003.
  - [76] H. Orman. RFC 2412: The OAKLEY Key Determination Protocol, November 1998. Status: INFORMATIONAL.
  - [77] Lawrence C. Paulson. Inductive analysis of the internet protocol TLS. Technical report, Computer Laboratory, University of Cambridge, 1997. Technical Report440.
  - [78] R. Pereira and R. Adams. RFC 2451: The ESP CBC-Mode cipher algorithms, November 1998. Status: PROPOSED STANDARD.
  - [79] J. Postel. RFC 788: Simple mail transfer protocol, November 1981. Obsoleted by RFC0821 [80]. Obsoletes RFC0780 [90]. Status: UNKNOWN. Not online.
  - [80] J. Postel. RFC 821: Simple mail transfer protocol, August 1982. See also STD0010 [81]. Obsoletes RFC0788 [79]. Status: STANDARD.
  - [81] Jonathan B. Postel. STD 10: Simple mail transfer protocol, August 1982. See also RFC0821, RFC1869 [80, 45]. Obsoleted by RFC2821 [67]. Obsoletes RFC788, RFC780, RFC772 [79, 90, 89].
  - [82] R. Rivest. RFC 1321: The MD5 message-digest algorithm, April 1992. Status: INFORMATIONAL.
  - [83] R. Rivest. RFC 2268: A description of the RC2(r) encryption algorithm, January 1998. Status: INFORMATIONAL.
  - [84] M. Rose. RFC 1460: Post office protocol — version 3, June 1993. Obsoleted by RFC1725 [64]. Obsoletes RFC1225 [86]. Status: DRAFT STANDARD.
  - [85] M. T. Rose. RFC 1081: Post office protocol: Version 3, November 1988. Obsoleted by RFC1225 [86]. Status: UNKNOWN.
  - [86] M. T. Rose. RFC 1225: Post office protocol: Version 3, May 1991. Obsoleted by RFC1460 [84]. Obsoletes RFC1081 [85]. Status: DRAFT STANDARD.
  - [87] A. Shacham, B. Monsour, R. Pereira, and M. Thomas. RFC 3173: ip payload compression protocol (ipcomp), September 2001.

- [88] H. Shacham and D. Boneh. Fast-track session establishment for tls. In *Proceedings of Internet Society's*, 2002.
- [89] S. Sluizer and J. Postel. RFC 772: Mail transfer protocol, September 1980. Obsoleted by RFC0780 [90]. Status: UNKNOWN. Not online.
- [90] S. Sluizer and J. Postel. RFC 780: Mail transfer protocol, May 1981. Obsoleted by RFC0788 [79]. Obsoletes RFC0772 [89]. Status: UNKNOWN. Not online.
- [91] William Stallings. *Cryptography and Network Security*. Prentice Hall, New Jersey, 1998.
- [92] Stephen Thomas. *SSL and TLS Essentials*. Wiley Computer Publisher, New York, 2000.
- [93] W Tuchman. Hellman presents no shortcut solutions to des. *IEEE Spectrum*, 16(7):40–41, January 1979.
- [94] David Wagner and Bruce Schneier. Analysis of the ssl 3.0 protocol. In *Second USENIX Workshop on Electronic Commerce*, pages 29–40, 1996 (1997 revised).
- [95] T. Ylonen and Ed. D. Moffat, Editor. ID: ssh authentication protocol, September 2002.
- [96] T. Ylonen and Ed. D. Moffat, Editor. ID: ssh connection protocol, October 2003.
- [97] T. Ylonen and Ed. D. Moffat, Editor. ID: ssh protocol architecture, October 2003.
- [98] T. Ylonen and Ed. D. Moffat, Editor. ID: ssh transport layer protocol, October 2003.
- [99] 馬場 達也. マスタリング *IPsec*. オライリー・ジャパン, 2001.